

Università degli Studi di Salerno

Dipartimento di Informatica

Dottorato di Ricerca in Informatica
Curriculum Internet of Things and Smart Technologies
XXXVIII Ciclo

PH.D. THESIS

Software Sustainability in the ML Era: Empirical Insights, Trade-offs, and LLM Methodologies

VINCENZO DE MARTINO

SUPERVISOR: **PROF. FABIO PALOMBA**

PHD PROGRAM DIRECTOR: **PROF. ANDREA DE LUCIA**

A.A 2024/2025

The final version of the thesis has been revised based on the reviews made by Prof. Dr. Marc Oriol Hilari and Prof. Dr. Daniel Feitosa.

The doctoral committee was composed of:

Prof. Carmine Gravino	University of Salerno, Italy
Prof. Panagiota Chatzipetrou	Örebro University School of Business, Sweden
Prof. Marc Oriol	Universitat Politècnica de Catalunya, Spain

First Release, 18 September 2025. Revised on 23 November 2025.

"If you want something you've never had, [...]
you must be willing to do something you've never done".
Unknown

ACKNOWLEDGMENTS

By now, I have become someone who writes a lot, really a lot, as can also be seen in this thesis. I hope, however, to be at least a little briefer and more direct in the acknowledgments, even though it is not simple to contain everything I feel inside.

It has been three years since I began my doctoral journey, and so many things have happened that I still struggle to believe. If I managed to experience them and bring them to completion, it was thanks to the people who surrounded me and accompanied me along this path.

My first thanks go to my family, my everything, the people I love more than anything else. I would never have come this far without the help and support of my mother **Loredana**, who created me, raised me, and continues to support me every single day of my life. She is my number one fan, tolerates all my daily complaints, and always manages to show me things from a different perspective. Thank you, Mom, for all that you have done and continue to do. This career is also thanks to you and the sacrifices you made. I know it, and I appreciate it every single day. Thank you to my sister **Angela**, always present and always ready to help me at any time. In recent years, we have grown very close, and our bond has become stronger. Your support has been essential. Thank you to my father **Domenico**, who, with his lightheartedness and irony, always manages to make me laugh and ease even the most complicated situations. Thanks to my grandparents **Anna** and **Antonio**, with whom I grew up and who taught me to look at the world from a different perspective. Without your kindness and care, life would be much duller. All of you are an integral part of who I am, even if sometimes we are physically apart. There are no words or "thank yous" big enough to repay what you have done for me: I owe you everything.

A special thanks goes to the PhD coordinator **Andrea De Lucia**, a person who always had a ready answer, even when it meant telling you uncom-

fortable but useful truths. Thank you, Andrea, for teaching us to be precise, meticulous, and rigorous. Thank you also for pushing us to take on the teaching experience, which was very important for my growth.

A huge thanks to my advisor **Fabio**, a person who has taught me so much over these years and who has truly shaped me in a significant way, pushing me to do things I would never have thought of doing before, and who shares with me many ambitions and dreams. I still remember when I rejected the idea of a PhD several times, but you saw something in me that I couldn't see in myself. For this reason, you did everything to accompany me along this winding path that has now led to this doctoral thesis. Thanks to you, I have become more independent and grown as a person, and you managed to calm me down whenever reviews arrived, always turning my "this reviewer hasn't even read the paper" or worse into the proper response for the reviewers. You supported me in all my countless ideas, always trying to find a way to show me the right path, and I have always appreciated this way of yours, you are an excellent guide. I hope that, in my own way, I have managed to change you and improve your life as much as you have mine. In these three years, we have shared so many moments, truly so many, that I will never forget in my life, and I hope to follow in your footsteps in my future career.

To the **Sesalab**, a special place that was much more than just a laboratory. It gave me friendships, collaborations, and unforgettable moments. A heartfelt thanks to **Dario Di Nucci**, a true friend, always there, always with advice or a joke at hand. Even in the worst moments, you managed to make me smile and gave me the strength to move forward. To **Fabiano**, whom I have known forever and without whom I probably would never have started this path. You have been a guide and constant support. Thanks to **Filomena**, who helped me so much over these years; to **Carmine**, who always brought lightness and good humor; to **Gemma**, for making these days easier. To my colleagues of the 38th cycle, **Gilberto** and **Dario**, travel companions in a thousand experiences. To **Giusy**, always ready to support me and help me grow. To **Giammaria**, who makes the lab a better place with his presence. And to **Stefano**, **Valeria**, **Manuel**, **Giulia**, **Emanuele**,

and **Francesco**, who in the early years made me immediately feel part of something special. Thanks also to all the other members of the lab – you are truly many – but each of you left something important in me: **Luana**, **Gabriele**, **Alexandra**, **Viviana**, **Gianmario**, **Antonio DP**, **Antonio T**, **Gerardo**, **Alfonso**, **Matteo**, **Alessandra**.

Thanks also to the **GESSI Group** in Barcelona, who welcomed me as if I were one of them. With you, I learned so much, faced new challenges, and even learned to teach in Spanish. Special thanks to **Silverio** and to all the others – **Xavier**, **Marc**, **Santiago**, **Joel**, **Alexandra**, **Pol**, **Quim**, **Francisco**, **Matias** – for making my experience in Spain so rich and unforgettable.

A thought also goes to my lifelong friends, those who are here and will always be. To **Vincenzo F.**, a special friend, an ever-present person who makes every day better. Even though we only see each other a few times a year, those moments always have a big impact. To **Giuseppe**, for being my personal lawyer, you are a great friend, and I hope to be like you too. To **Mario**, a constant presence and an enormous shoulder to lean on. To **Pierpaolo**, **Carlo**, **Giacomo**, **Roberto**, **Agostino**, **Fabrizio**, **Francesco "Ciccio B."**, **Francesco "Ciccio R."**, **Gianpaolo**, **Marco**, **Gregorio**, **Delfina**, **Maya**, **Giuliana**, **Marianna**: you have been and still are a fundamental group in my life. To **Federica**, who is much more than a colleague, a friend who endures my days and whom I thank deeply.

To the friends I met in Spain: **Daniele**, my roommate, **Alfredo**, **Step (Señora)**, **Monica**, **Vincenzo P.**, and all the others who made this period of my life so special.

Finally, to **Pilar**, who entered my life when I least expected it, but with her sweetness and constant support, has managed to make my days brighter. Thank you for always being there and helping me grow every day. I hope you will remain by my side for a long time.

Thanks also to the people I have not included in this list, but who have contributed to my personal growth and accompanied me throughout these years. At the end of this journey, I realize that this thesis is not only mine. It is the result of all the bonds, affections, advice, and support I have received. Each of you has played a role, big or small, and without you, nothing would

have been possible. This thesis is also yours, and my “grazie” “thank you” “gracias” “merci”, is forever.

ABSTRACT

Machine Learning (ML) is increasingly embedded in critical software systems, enabling powerful applications across diverse domains. Yet, the engineering of ML-enabled systems raises new challenges that are not solely functional but revolve around non-functional requirements (NFRs), with sustainability emerging as one of the most pressing. Despite its importance, sustainability is still poorly conceptualized, inconsistently evaluated, and rarely supported by automated tools, leaving a gap between high-level principles and actionable engineering practices.

This dissertation addresses these gaps through an integrated empirical investigation guided by four research questions. First, it examines how sustainability is conceptualized across research and practice. The findings reveal a misalignment: academia tends to frame sustainability through abstract environmental principles, whereas practitioners adopt a pragmatic view grounded in efficiency, maintainability, and ethical responsibility. Second, explore how sustainability can be evaluated in relation to trade-offs with other NFRs. Empirical benchmarks demonstrate that improvements in fairness and accuracy often lead to increased energy consumption and reduced maintainability, underscoring the need to evaluate sustainability as a multidimensional quality.

To scale the study of sustainability practices, the thesis introduces *MARK*, a rule-based classifier that automatically distinguishes ML repositories as producers, consumers, or hybrids. Validated on thousands of projects, MARK reduces manual effort and enables reproducible dataset construction. Building on this dataset, large language models (LLMs) are then employed to mine sustainability practices, with a focus on green architectural tactics. LLMs not only confirm known tactics but also uncover previously undocumented ones, showing their potential to capture implicit sustainability signals across heterogeneous artifacts, though with risks of variability, hallucinations, and computational overhead.

To ensure the rigorous and sustainable use of LLMs in repository mining, the thesis proposes *PRIMES 2.0*, a methodological framework that structures the use of LLMs into stages covering prompt design, empirical validation, model comparison, and trade-off evaluation. This framework advances reproducibility, rigor, and sustainability awareness, moving beyond ad hoc experimentation toward systematic adoption in empirical software engineering.

Overall, the dissertation contributes to bridging the gap between principles and practice in sustainable ML engineering. It clarifies the conceptualization of sustainability, quantifies its trade-offs with other NFRs, introduces automated tools to scale repository mining, and establishes methodological foundations for LLM-based studies. In doing so, it provides both theoretical insights and practical support for building ML-enabled systems that are not only effective and fair, but also sustainable, reliable, and socially responsible.

CONTENTS

List of Figures xvi
List of Tables xxi

PART I: Introduction and Background

1 Introduction 3
1.1 Context and Motivation 3
1.2 Research Statement 5
1.3 Research Contributions 9
1.3.1 Publicly Available Tools and Replication Packages 11

2 A Systematic Analysis of NFRs in ML-Enabled Systems: State of the Art, Practice, and Challenges 13
2.1 Classification and Challenges of Non-Functional Requirements in ML-enabled systems: A Systematic Literature Review 13
2.2 Research Method 15
2.2.1 Research Objectives and Questions 17
2.2.2 Execution of the Research Methods 20
2.3 Analysis of the Results 24
2.3.1 **RQ**$_1$ - What are the non-functional requirements of ML-enabled software considered by researchers? . 25
2.3.2 **RQ**$_2$ - What are the challenges for ML-enabled systems specifically relating to non-functional requirements? 35
2.4 Discussion and Implications 68

3 Sustainability from the Practitioner's Perspective 75
3.1 Introduction 75
3.2 Research Process Overview 78
3.2.1 First Phase: Qualitative Study 80
3.2.2 Second Phase: Quantitative Survey 87

3.2.3 Connecting the Dots Between Qualitative and Quantitative Findings 94
3.3 Analysis of the results 95
3.3.1 RQ_1 — Sustainability Perception 95
3.3.2 RQ_2 — Sustainability Approaches 99
3.3.3 RQ_3 — Sustainability Challenges 110
3.4 Discussion and Implications of the Study 118
3.4.1 Positioning the findings within the current state of the art 118
3.4.2 Using the Sustainable Approaches Elicited to Optimize Non-Functional Requirements 123
3.4.3 Breaking the barriers to the adoption of sustainability approaches 124

PART II: Empirical Foundations for Sustainable ML System Design

4 Examining the Impact of Bias Mitigation Algorithms on the Sustainability of ML-enabled Systems: A Benchmark Study. 129
4.1 Introduction 129
4.2 Empirical Study Design 133
4.2.1 Objects of the study 136
4.2.2 Subjects of the Study 138
4.2.3 Empirical Study Variables 139
4.2.4 Experimental Hypotheses, Execution, and Analysis 144
4.3 Analysis of the Results 149
4.3.1 **RQ_1** - How do bias mitigation algorithms impact social sustainability regarding multiple protected attributes? 150
4.3.2 **RQ_2** - How do bias mitigation algorithms impact environmental sustainability regarding multiple protected attributes? 155
4.3.3 **RQ_3** - How do bias mitigation algorithms impact economic sustainability regarding multiple protected attributes? 159

4.4 Discussion and Implications 166
4.4.1 Discussion of the Findings 166
4.4.2 Implications of the Study 170
5 Into the ML-Universe: An Improved Classification and Characterization of Machine-Learning Projects. 177
5.1 Introduction 177
5.2 MARK: Machine learning Automated Rule-based Classification Kit 182
5.2.1 Dataset Specification 182
5.2.2 Definition of Knowledge Base 183
5.2.3 Classification Rules 186
5.2.4 Tool-Supported MARK Classification 190
5.3 Research Questions and Methodical Overview 190
5.4 RQ_1. Empirical Study: On the Performance of MARK 192
5.4.1 Data Collection 192
5.4.2 Data Analysis 194
5.4.3 Analysis of the Results 196
5.5 RQ_2. Empirical Study: On the Usefulness of MARK 206
5.5.1 Data Collection 207
5.5.2 Data Analysis 209
5.5.3 Analysis of the Results 210
5.6 Discussion and Implications 215

PART III: Leveraging Large Language Models to Support Sustainable ML Engineering

6 Sustainable Practices in ML-enabled Systems: An LLM-based Repository Mining Study 225
6.1 Introduction 225
6.2 Research Goal and Questions 228
6.3 Research Method 230
6.3.1 Sampling and Dataset Description 230
6.3.2 Selection of Architectural Green Tactics 231
6.3.3 Extraction of Architectural Green Tactics 233
6.3.4 Model Validation 237

6.3.5 Data Analysis 241
6.4 Analysis of the Results 243
6.4.1 **RQ_1.** Adoption of Green Tactics in Practice 243
6.4.2 **RQ_2.** Elicitation of Additional Green Tactics 246
6.5 Discussion and Implications 253
7 A Methodological Framework for LLM-Based Mining of Software Repositories 259
7.1 Introduction 259
7.2 Research Process Overview 263
7.2.1 Rapid Review 265
7.2.2 Questionnaire Survey 272
7.3 Analysis of the Results 278
7.3.1 Context of the Analyzed Studies: Research Topics and Platforms 278
7.3.2 RQ_1: What approaches are employed when using large language models for repository mining? . . . 280
7.3.3 RQ_2: What are the current threats to validity faced by researchers when they use large language models for repository mining? 298
7.4 Discussions and Implications 314
7.4.1 Understanding Emerging Approaches in LLM-based MSR 315
7.4.2 Revisiting Technical and Methodological Threats To Validity in MSR 316
7.4.3 Implications for the Mining Software Repositories Community 319
7.4.4 PRIMES 2.0: An Empirical-Based Methodological Framework for LLM-Based Mining of Software Repositories 321

PART IV: Synthesis, Threats and Conclusions

8 Summary of the Main Findings 327

8.1 RQ_A: How is sustainability conceptualized for researchers and practitioners within the broader landscape of non-functional requirements in ML-enabled systems? 327
8.2 RQ_B: How can we evaluate sustainability in ML-enabled systems, accounting for trade-offs between different non-functional requirements? 329
8.3 RQ_C: How can automated tools and LLMs support the mining and identification of sustainability practices in ML-enabled repositories? 330
8.4 RQ_D: How can LLMs be effectively and sustainably leveraged in repository mining scenarios, considering the perspective of researchers interested in mining ML repositories? 331
9 Threats to Validity 333
9.1 Common Threats to Validity 333
9.2 Quantitative-Specific Threats 336
9.3 Qualitative-Specific Threats 337
10 Conclusion 339

List of Publications 343

Bibliography 347

LIST OF FIGURES

Figure 2.1 Overview of the process for selecting papers. . . . 19
Figure 3.1 Overview of the research process. 82
Figure 3.2 Survey participants distribution by continent. . . 90
Figure 3.3 Sustainability rate for survey respondents. 96
Figure 3.4 Sustainability importance for survey respondents. 97
Figure 3.5 Word cloud generating analyzing survey participants responses for the first research question. . 97
Figure 4.1 Overview of the research process. 133
Figure 4.2 Execution process. 146
Figure 4.3 Impact of Fairness Algorithms on *Mean Difference* with different datasets. The *algorithms* are Reweighing (RW), Meta Fair Classifier (MF), Gerry Fair Classifier (GF), Exponentiated Gradient Reduction (EG), PrejudiceRemover (PR), Grid Search Reduction (GS). A green dot • indicates a statistically significant improvement against the baseline, a red dot • indicates a deterioration instead. Finally, a gray dot • indicates that there is no statistically significant difference with the baseline. 151
Figure 4.4 Impact of Fairness Algorithms on *Equal Opportunity Difference* with different datasets. The *algorithms* are Reweighing (RW), Meta Fair Classifier (MF), Gerry Fair Classifier (GF), Exponentiated Gradient Reduction (EG), PrejudiceRemover (PR), Grid Search Reduction (GS). A green dot • indicates a statistically significant improvement against the baseline, a red dot • indicates a deterioration instead. Finally, a gray dot • indicates that there is no statistically significant difference with the baseline. 152

Figure 4.5 Impact of Fairness Algorithms on *Average Odds Difference* with different datasets. The *algorithms* are Reweighing (RW), Meta Fair Classifier (MF), Gerry Fair Classifier (GF), Exponentiated Gradient Reduction (EG), PrejudiceRemover (PR), Grid Search Reduction (GS). A green dot • indicates a statistically significant improvement against the baseline, a red dot • indicates a deterioration instead. Finally, a gray dot • indicates that there is no statistically significant difference with the baseline. 154

Figure 4.6 Impact of Fairness Algorithms on *Energy Consumption (Joule)* with different datasets. The *algorithms* are Reweighing (RW), Meta Fair Classifier (MF), Gerry Fair Classifier (GF), Exponentiated Gradient Reduction (EG), PrejudiceRemover (PR), Grid Search Reduction (GS). A green dot • indicates a statistically significant improvement against the baseline, a red dot • indicates a deterioration instead. Finally, a gray dot • indicates that there is no statistically significant difference with the baseline. 156

Figure 4.7 Impact of Fairness Algorithms on *Carbon Emission (g/CO2)* with different datasets. The *algorithms* are Reweighing (RW), Meta Fair Classifier (MF), Gerry Fair Classifier (GF), Exponentiated Gradient Reduction (EG), PrejudiceRemover (PR), Grid Search Reduction (GS). A green dot • indicates a statistically significant improvement against the baseline, a red dot • indicates a deterioration instead. Finally, a gray dot • indicates that there is no statistically significant difference with the baseline. 157

Figure 4.8 Impact of Fairness Algorithms on *Accuracy* with different datasets. The *algorithms* are Reweighing (RW), Meta Fair Classifier (MF), Gerry Fair Classifier (GF), Exponentiated Gradient Reduction (EG), PrejudiceRemover (PR), Grid Search Reduction (GS). A green dot • indicates a statistically significant improvement against the baseline, a red dot • indicates a deterioration instead. Finally, a gray dot • indicates that there is no statistically significant difference with the baseline. 160

Figure 4.9 Impact of Fairness Algorithms on *F1 score* with different datasets. The *algorithms* are Reweighing (RW), Meta Fair Classifier (MF), Gerry Fair Classifier (GF), Exponentiated Gradient Reduction (EG), PrejudiceRemover (PR), Grid Search Reduction (GS).A green dot • indicates a statistically significant improvement against the baseline, a red dot • indicates a deterioration instead. Finally, a gray dot • indicates that there is no statistically significant difference with the baseline. 161

Figure 4.10 Impact of Fairness Algorithms on *Training Time (seconds)* with different datasets. The *algorithms* are Reweighing (RW), Meta Fair Classifier (MF), Gerry Fair Classifier (GF), Exponentiated Gradient Reduction (EG), PrejudiceRemover (PR), Grid Search Reduction (GS).A green dot • indicates a statistically significant improvement against the baseline, a red dot • indicates a deterioration instead. Finally, a gray dot • indicates that there is no statistically significant difference with the baseline. 163

Figure 4.11 Impact of Fairness Algorithms on *Storage Weight (KB)* with different datasets. The *algorithms* are Reweighing (RW), Meta Fair Classifier (MF), Gerry Fair Classifier (GF), Exponentiated Gradient Reduction (EG), PrejudiceRemover (PR), Grid Search Reduction (GS). A green dot • indicates a statistically significant improvement against the baseline, a red dot • indicates a deterioration instead. Finally, a gray dot • indicates that there is no statistically significant difference with the baseline. 164

Figure 4.12 Energy Consumption Vs. Cumulative Bias for each dataset. The *algorithms* are Reweighing (RW), Meta Fair Classifier (MF), Gerry Fair Classifier (GF), Exponentiated Gradient Reduction (EG), PrejudiceRemover (PR), Grid Search Reduction (GS). The numerical values represent cumulative bias (purple), where lower values are better, and energy consumption (green), where lower values are better. 170

Figure 4.13 Cumulative Bias Vs. Storage Weight for each dataset. The *algorithms* are Reweighing (RW), Meta Fair Classifier (MF), Gerry Fair Classifier (GF), Exponentiated Gradient Reduction (EG), PrejudiceRemover (PR), Grid Search Reduction (GS). The numerical values represent cumulative bias (purple), where lower values are better, and storage weight (red), where lower values are better. 172

Figure 4.14 Cumulative Bias Vs. Energy Consumption Vs. Accuracy for each dataset. The *algorithms* are Reweighing (RW), Meta Fair Classifier (MF), Gerry Fair Classifier (GF), Exponentiated Gradient Reduction (EG), PrejudiceRemover (PR), Grid Search Reduction (GS). The numerical values represent cumulative bias (purple), where lower values are better; energy consumption (green), where lower values are better; and accuracy (blue), where higher values are better. 173
Figure 5.1 MARK: Overview of the Design. 181
Figure 5.2 Overview of the Knowledge Base Construction and Refinement. 182
Figure 5.3 Top 10 ML Libraries used in *ML-Model Producers'* projects. 211
Figure 5.4 Top 10 ML Libraries used in *ML-Model Consumers'* projects. 211
Figure 5.5 Comparison of the top 10 ML libraries used by ML-Model Producers' and ML-Model Consumers'. . . 211
Figure 5.6 Distribution of the categories of ML projects among the selected metrics. The categories are ML-Model Producer (MP), ML-Model Consumer (MC), ML-Library & Toolkit (LT), ML-Model Producers & Consumers (MPC). 213
Figure 6.1 Overview of the research process. 228
Figure 6.2 The prompt designed to extract architecture green tactics from software repositories. 238
Figure 6.3 Frequency of Green Tactics in ML Projects. 244

Figure 6.4 Catalog of 39 Green Architectural Tactics for ML-Enabled Systems, adopted from [182] and enhanced with use frequency of all tactics and the elicitation of nine new tactics. The symbol '*' indicates a new tactic found by our LLM-based mechanism. Colors indicate frequency using quartiles on 168 ML projects. 254
Figure 7.1 Overview of the Research Process. 262
Figure 7.2 Papers published over time. 270
Figure 7.3 Papers venues. 270
Figure 7.4 Frequency of software repositories and platforms analyzed in the rapid review (N=15 papers) and survey (N=22 respondents). 284
Figure 7.5 PRIMES 2.0: An Empirical-Based Methodological Framework for LLM-Based Mining of Software Repositories. 321

LIST OF TABLES

Table 1.1 The research studies discussed in this thesis and how they contributed to addressing the problems here faced. 10

Table 2.1 Non-Functional Requirements in ML-enabled systems. The *'Meaning'* column reports whether the non-functional requirements are *'new'* (N), i.e., they are specific to ML-enabled systems, are *'altered'* (A), i.e., their meaning is different with respect to the definition of the non-functional requirements in non-ML-enabled systems, or are the *'same'* (S), i.e., their meaning is the same as the non-functional requirements in non-ML-enabled systems. 27
Table 2.2 Challenges Related to Non-Functional Requirements in ML-Enabled Systems. 36
Table 3.1 Structure and examples of questions of semi-structured interviews. 81
Table 3.2 Information of the interview study participants. . 86
Table 3.3 Categorization of Sustainable Approaches 100
Table 3.4 Frequency of sustainability tool adoption by survey participants. 108
Table 3.5 Categorization of Sustainable Challenges 111
Table 3.6 Frequency of sustainability challenges in ML-enabled systems according to the survey participants. 117
Table 3.7 Mapping of Sustainability Approaches to ML-enabled Systems Stages and NFRs. Sustainability Definition (Def). 122
Table 4.1 Datasets used in the experimentation. 137
Table 4.2 Variables of the study 139
Table 5.1 Statistical Metrics. 193
Table 5.2 Confusion Matrix for *'ML-Model Producer'* . . . 195
Table 5.3 Confusion Matrix for *'ML-Model Consumer'* . . . 195
Table 5.4 Statistical Metrics for *'ML-Model Producers'* and *'ML-Model Consumers'* 197
Table 5.5 Causes of Misclassification and their Frequency. . 199

Table 5.6 Statistical Metrics for *'ML-Model Producers'* and *'ML-Model Consumers'* with different Knowledge Base and Rules (R) configurations of MARK. The symbol 🔒 indicates essential rules for classification. 200
Table 5.7 Confusion Matrix for *'ML-Model Producer & Consumer'* 206
Table 5.8 Performance Metrics for *'ML-Model Producer & Consumer'* 206
Table 5.9 Distribution of Projects by Category. 207
Table 5.10 Metrics selected for analyzing community engagement, activity, and collaboration aspects of ML projects. 209
Table 5.11 Kruskal-Wallis H and Mann-Whitney U tests. Number(#), ML-Model Producer (MP), ML-Model Consumer (MC), ML-Library & Toolkit (LT), ML-Model Producers & Consumers (MPC). 213
Table 6.1 Average (AVG) and Median (Med) of each group of projects. 232
Table 6.2 Green architectural tactics for ML-Enabled Systems selected in the scope of the work and the Large Language Model (LLM) detection rationale. 234
Table 6.3 New Green Tactics for ML-Enabled Systems. (*) Indicate energy efficiency is a secondary quality assurance (QA) 246
Table 7.1 Publication Venues. 266
Table 7.2 Inclusion and exclusion criteria used in the selection process. 268
Table 7.3 Summary of the stages and methodological approaches of LLM4 MSR research. Categories marked with (*) are newly introduced in this version; those marked with (†) represent revisions of categories from the original PRIMES framework [80]. 281

Table 7.4 Summary of Threats, Mitigation Strategies, and Employed Techniques. 299

PART I

INTRODUCTION AND BACKGROUND

1

INTRODUCTION

1.1 CONTEXT AND MOTIVATION

Machine Learning (ML) is playing an increasingly central role across theory, experimentation, and simulation domains [262, 290]. Its integration into everyday decision-making processes is growing, with both companies and individuals depending on ML models to support informed decisions [401] or automate tasks that would otherwise require substantial human effort [285]. At the same time, ML-enabled systems, software systems with functionalities enabled by at least one AI component, are being adopted in a wide range of domains, with recent applications demonstrating impressive levels of accuracy and efficiency [234, 246, 260]. Given these trends, the presence of ML-enabled systems is projected to grow even further across sectors in the near future [283, 390].

Additionally, the advent of Large Language Models (LLMs) has taken on a central role in software engineering, reshaping established practices in software development, maintenance, and comprehension [103, 135, 166]. Recent work by Hou et al. [166] has systematically mapped how LLMs are reshaping software engineering, providing a comprehensive taxonomy of tasks supported by LLMs across the development lifecycle, from code generation [93, 374] and comprehension [252] to testing [275], and maintenance [274]. Their findings highlight the breadth of LLM applicability and their disruptive potential in automating and augmenting main software engineering practices.

Despite this momentum, the widespread adoption of such systems faces ongoing challenges. These are often not tied to functional capabilities, but rather to non-functional attributes[156, 255]. Non-functional requirements (NFRs) are conditions used to assess system qualities beyond its

specific behaviors [33]. In ML-enabled systems, these attributes—such as robustness, trustworthiness, and sustainability play a critical role in determining overall system quality [146, 161, 228]. Consequently, the software engineering community, particularly within the Software Engineering for Artificial Intelligence (SE4AI) field, has devoted growing attention to understanding and supporting the management of NFRs in ML-enabled systems [25, 144, 191]. This line of research is further encouraged by institutional support: for instance, the European Union recently introduced the Artificial Intelligence Act,[1] which prioritizes advancements related to the non-functional aspects of AI-based systems.

While the management of NFRs in ML-enabled systems has garnered increasing attention, significant challenges remain, particularly when such systems are deployed at scale. The training and inference phases of ML models require vast amounts of data and computational power [304]. Although these demands do not necessarily hinder adoption, they raise critical concerns about the sustainability of ML-enabled systems [347, 352]. In this context, sustainability, commonly defined as "meeting the needs of the present without compromising the ability of future generations to meet their own"[35], emerges as a key cross-cutting requirement for both developers and end users[298, 336]. As further analyzed in the scope of this thesis (see Chapter 2), sustainability has increasingly been recognized as a central requirement in ML-enabled systems, encompassing social, environmental, and economic dimensions.

Software sustainability is a multidimensional construct encompassing environmental, economic, and social aspects [41, 206, 348]. From an environmental perspective, the emphasis lies in reducing the carbon footprint of software systems, particularly relevant for resource-intensive ML workloads. Economic sustainability addresses the long-term viability and cost-effectiveness of maintaining and evolving software. Meanwhile, social sustainability includes considerations such as fairness, accessibility, and the ethical impact of software decisions. Crucially, sustainability is not a

[1] The European Union Artificial Intelligence Act: https://artificialintelligenceact.eu.

fixed attribute but rather an emergent property resulting from the dynamic interaction among these dimensions.

This holistic perspective is articulated in the Karlskrona Manifesto for Sustainability Design [22], which encourages the software engineering community to adopt a systemic and forward-looking mindset. More recently, McGuire et al. [242] emphasized that sustainability should be treated as a stratified and multisystemic property, one that evolves over time and must be continuously reassessed in practice. These evolving perspectives highlight the need for empirical investigations that uncover how sustainability is operationalized and analyzed within real-world ML-enabled systems, and how trade-offs between sustainability and other NFRs are managed during the software lifecycle.

> ⚠ *This thesis addresses a critical gap in the intersection between machine learning and software engineering by investigating the sustainability of ML-enabled systems through empirical methods. It aims to advance our understanding of sustainability as a software quality attribute, highlighting the principles, practices, challenges and trade-offs that practitioners must consider to ensure the long-term maintainability and evolvability of these systems. In doing so, the thesis also examines how the emerging use of large language models can facilitate software engineering practices.*

1.2 RESEARCH STATEMENT

Over the last decades, the software engineering community has investigated the topic discussed in this thesis with considerable attention. Despite the high-quality contributions in the sustainability field and the use of artificial intelligence to solve software engineering activities, there are still unexplored aspects worthy of investigation.

The main limitations and critical aspects that were identified are the following:

While the SE4AI research community has produced a significant body of knowledge concerning the management of NFRs in ML-enabled systems, a comprehensive and systematic synthesis of this knowledge is still lacking [144–146]. In particular, there is limited understanding of which NFRs most critically impact ML-enabled systems and what practical challenges and trade-offs arise when addressing them in real-world scenarios.

A first important challenge concerns the conceptualization of sustainability within the broader context of NFRs. Although NFRs have long played a central role in traditional software engineering [41, 206, 348], their adaptation to ML-based systems is often considered in isolation from the individual dimensions of sustainability. Existing taxonomies and classifications tend to emphasize isolated properties, such as fairness or robustness[144, 146]. Moreover, the perspectives of practitioners are not always fully captured in academic investigations [143], making it harder to assess how research insights align with real-world needs and practices. Clarifying how sustainability is framed, both in research and in practice, is thus essential to support its integration as a meaningful and actionable requirement.

RGap$_1$: ⚠ Lack of clear concepts and alignment with researchers and practitioners in defining NFRs and sustainability requirements.

A second challenge is the limited exploration of the trade-offs that arise when addressing sustainability concerns. Many studies have focused on improving specific NFRs, such as fairness, but often evaluate these interventions in isolation or only consider accuracy metrics [64, 164]. For example, while bias mitigation algorithms can enhance fairness, they may decrease overall accuracy. Previous studies have discussed the definition of innovative techniques capable of balancing equity objectives with other sustainability principles, thus paving the way for research in this specific area [56, 299]. Although these trade-offs are recognized, they are not consistently assessed through multi-dimensional evaluations. Since sustainability is inherently multifaceted, it is essential to better understand how different NFRs interact and how improvements in one area may affect others.

RGap$_2$: ⚠ Lack of multi-dimensional evaluations of sustainable solutions and their trade-offs.

A significant gap exists in the development of automated tools and techniques that assist researchers and practitioners in identifying and analyzing sustainability practices in ML projects. While open-source repositories offer valuable resources such as code, configurations, and usage patterns, analyzing these artifacts, especially on a large scale, is often challenging and can require considerable effort, obtaining a final dataset that is not generalizable [40]. To effectively address this gap, it is necessary to understand the differences among various types of ML projects, as sustainability concerns must be approached in distinct ways depending on the project's nature (e.g., training-focused versus inference-focused systems) [233]. Although there have been some initial efforts to classify ML projects [136, 251], these attempts are limited in granularity and do not consistently account for distinctions such as model producers (train ML models) and model consumers (use and deploy ML models). To foster a more nuanced understanding of project types and their implications for sustainability, there is a need for automated classification mechanisms that can serve as a foundation for empirical research and practical applications. Finally, the growing adoption of LLM models for software repository extraction and analysis is transforming the way researchers explore code, issues, documentation, and developer practices. LLMs enable new forms of large-scale semantic analysis that go beyond traditional keyword- or pattern-based techniques, supporting activities such as classification, summarization, and code and text extraction. While green practices [182, 350] aimed at improving energy efficiency are described in the literature, there is a lack of understanding of how these are actually adopted by developers.

RGap$_3$: ⚠ Lack of automated support for classifying ML projects

RGap$_4$: ⚠ Lack of analysis of sustainable practices in open-source repositories.

Furthermore, as the adoption of LLMs becomes more widespread, there is a growing need to understand how these models can be applied to

repository mining in a sustainable and effective way. While LLMs offer unprecedented opportunities for large-scale activities, it is not yet clear how to optimize their use and to ensure efficiency, reproducibility, and methodological soundness. This uncertainty is particularly critical because unsystematic or inefficient use of LLMs may lead to wasted resources and threaten the reproducibility of findings. The absence of empirically grounded guidelines covering aspects such as prompt design, validation strategies, and reproducibility represents a significant gap, preventing researchers from mitigating methodological threats while effectively utilizing LLMs for repository analysis.

RGap$_5$: ⚠ Lack of support for empirical methodological frameworks for LLM-based mining repository.

To address the research gaps identified above, this dissertation is guided by the following four overarching research questions:

> 🔍 **RQ$_A$:** *How is sustainability conceptualized for researchers and practitioners within the broader landscape of non-functional requirements in ML-enabled systems ?*

This question addresses the lack of consensus around sustainability as a software quality concern, and seeks to bridge the gap between academic definitions and real-world perspectives (see RGap$_1$).

> 🔍 **RQ$_B$:** *How can we evaluate sustainability in ML-enabled systems, accounting for trade-offs between different non-functional requirements?*

This question focuses on understanding the interactions between multiple NFRs (e.g., fairness, energy efficiency) when (see RGap$_2$).

> **RQ$_C$:** *How can automated tools and LLMs support the mining and identification of sustainability practices in ML-enabled repositories?*

This question investigates how large-scale software artifacts can be analyzed to uncover sustainability-relevant insights, with a focus on project classification and sustainability signal extraction (see RGap$_3$ and RGap$_4$).

> **RQ$_D$:** *How can LLMs be effectively and sustainably leveraged in repository mining scenarios, considering the perspective of researchers interested in mining ML repositories?*

This question focuses on the design of empirically grounded frameworks to guide the effective use of LLMs in repository mining studies (see RGap$_5$).

The empirical experiments conducted in this thesis adhere to the best practices established in Empirical Software Engineering research. Specifically, the thesis utilizes the *Goal-Question-Metric* paradigm [39] to identify and evaluate products, processes, and resources.

For each study, the forthcoming sections will detail the objectives, research questions, and the instruments utilized to conduct the analysis. To ensure rigor in our investigations, I adhered to the *ACM/SIGSOFT Empirical Standards* [281].

1.3 RESEARCH CONTRIBUTIONS

This thesis is structured into four main chapters, which together address the theme of sustainability in ML-enabled systems through an integrated approach combining theory, practice, and automation.

- Part I provides the conceptual foundations, offering a comparative overview of the knowledge acquired from both scientific literature and professional practice, with the aim of building a solid and systematic understanding of the problem. Part I addresses the **RQ$_1$**

Table 1.1: The research studies discussed in this thesis and how they contributed to addressing the problems here faced.

ID	Study Name	Authors	Contribution	Venue & Quartile	Ref
J1	Classification and Challenges of Non-Functional Requirements in ML-enabled systems: A Systematic Literature Review	**De Martino V.** and Palomba F.	A Systematic Literature Review on Non-Functional Requirements in ML-enabled systems and the related challenges	IST 2025 (Q1)	Chapter 2
J2	Sustainability of Machine Learning-Enabled Systems: The Machine Learning Practitioner's Perspective	**De Martino V.**, Lambiase S., Pecorelli F., Van Den Heuvel W. J., Ferrucci F., and Palomba F.	A mixed-method study on the perspectives of practitioners on sustainability, the approaches adopted, and sustainability challenges	TOSEM 2025 (Q1)	Chapter 3
J3	Examining the Impact of Bias Mitigation Algorithms on the Sustainability of ML-enabled Systems: A Benchmark Study	**De Martino V.**, Voria G., Troiano C., Catolino G., and Palomba F.	A benchmark study assessing how bias-mitigation algorithms affect social, environmental, and economic sustainability dimensions in ML-enabled systems	JSS 2025 (Q1)	Chapter 4
J4	Into the ML-universe: An Improved Classification and Characterization of Machine-Learning Projects	**De Martino V.**, Recupito G., Giordano G., Di Nucci D., Ferrucci F., and Palomba F.	A large-scale empirical study presenting a classifier to distinguish different ML projects	JSS 2025 (Q1)	Chapter 5
J5	Sustainable Practices in ML-enabled Systems: An LLM-based Repository Mining Study	**De Martino V.**, Martínez-Fernández S., and Palomba F.	A mixed-method study using LLMs to identify and extract green architectural tactics from open-source ML projects	EMSE (Major Revision)	Chapter 6
J6	Toward a Methodological Framework for LLM-Based Mining of Software Repositories	**De Martino V.**, Castaño J., Palomba F., Franch X., and Martínez-Fernández S.	A mixed-method study combining a rapid review and a survey to analyze approaches and threats in LLM-based mining studies, developing the PRIMES 2.0 framework for methodological rigor.	EMSE (Under Revision)	Chapter 7

- Part II explores the role of sustainability in relation to other NFRs that often compete with each other. The analysis highlights the importance of adopting a trade-off-aware perspective and introduces a classification tool of ML projects into different types. This distinction, implemented through the MARK tool, enables a more targeted empirical analysis to identify sustainable practices. Part II addresses the **RQ$_2$** and **RQ$_3$**
- Part III of the thesis proposes an LLM-based mechanism to address the challenges identified in the empirical analysis of the sustainability of ML-enabled systems. From a practical standpoint, it demon-

strates how LLMs can be utilized to identify sustainable practices in open-source projects, surpassing existing theoretical taxonomies and shedding light on undocumented tactics employed in real-world development. From a methodological perspective, this discussion examines the approaches and threats to validity associated with using LLMs in mining studies. It introduces the PRIMES 2.0 framework, designed to enhance the validity and reproducibility of such studies. In this way, LLMs are presented as objects of study and support for both empirical research and the implementation of sustainable solutions. Part III addresses the **RQ$_3$** and **RQ$_4$**.

- Finally, Part IV brings together concluding discussions, threats to validity faced during these chapters, and reflections, summarizing the insights developed throughout the thesis and for future directions.

Table 1.1 provides an overview of the scientific contributions included in the thesis, specifying for each study the type of publication (*"J"* for articles in international journals), the title, the main contribution, and the chapter in which the study is discussed.

1.3.1 *Publicly Available Tools and Replication Packages*

All the raw data, tools, datasets, and scripts are publicly available for each study by including a replication package. This includes raw data, reusable datasets, and reproducible scripts, which are provided through online appendices. The replication packages of Chapter 2 [238], Chapter 3 [81], Chapter 4 [85], Chapter 5 [239], Chapter 6 [237], Chapter 7 [236]. Additionally, pre-prints of the published articles are made available online.

2

A SYSTEMATIC ANALYSIS OF NFRS IN ML-ENABLED SYSTEMS: STATE OF THE ART, PRACTICE, AND CHALLENGES

This Chapter provides the necessary background information helpful to understand the next chapters and summarizes the state-of-the-art, giving an overview of closely connecting studies on sustainability and non-functional requirements in ML-enabled systems, the sustainability from the practitioner's perspective, and the foundation on mining software repository.

2.1 CLASSIFICATION AND CHALLENGES OF NON-FUNCTIONAL REQUIREMENTS IN ML-ENABLED SYSTEMS: A SYSTEMATIC LITERATURE REVIEW

A non-functional requirement (NFR) is defined as a condition that specifies a criterion that may be used to judge the operation of a system rather than specific behaviors [33]. In the context of ML-enabled systems, non-functional attributes may affect the overall level of reliability, trustworthiness, and sustainability of these systems [146, 161, 228]. It is therefore not surprising that the software engineering research community—and specifically the software engineering for artificial intelligence (SE4AI) research branch—has been investing notable efforts in understanding non-functional requirements of ML-enabled systems, other than proposing methods and instruments to support practitioners when dealing with them [25, 144, 191]. This effort is also stimulated by government and funding agencies, which are more and more willing to invest in the matter, e.g.,

the European Union has recently approved the *Artificial Intelligence Act*,[1] which aims at promoting research on themes connected to the improvement of non-functional attributes of artificial intelligence-based software systems.

Recent advances in the field of SE4AI contributed to the development of a consistent body of knowledge with respect to the management of multiple non-functional requirements, including fairness [34, 109, 385], security [147, 225], privacy [222, 313], and more [203]. While recognizing the relevant advances made over the last years, this research identifies a notable key limitation.

> ⚠ *Despite the current, extensive body of knowledge produced by the SE4AI research community regarding the management of non-functional requirements in ML-enabled systems, there is still a lack of a comprehensive, systematic synthesis of the current knowledge on the* ***non-functional requirements impacting ML-enabled systems*** *and the* ***challenges encountered in addressing them within the context of these systems.***

An improved understanding of these aspects may have critical implications for researchers and practitioners. First, researchers might learn more about the current state of the art, possibly identifying neglected research angles that would be worth further investigating. At the same time, practitioners may have a comprehensive overview of the instruments that researchers have been providing to support the analysis and optimization of non-functional requirements, possibly accelerating the technological transfer of academic prototypes to industry.

As highlighted in **Gap #1**, there is still a lack of clear concepts and alignment between researchers and practitioners in defining NFRs and sustainability requirements for ML-enabled systems. To address this gap, this chapter presents a systematic literature review (SLR) on non-functional

[1] The European Union Artificial Intelligence Act: https://artificialintelligenceact.eu.

requirements of ML-enabled systems [2]. In particular, I focus on traditional ML and deep learning (DL) systems but do not address large language models (LLM) or generative artificial intelligence systems integration. By focusing on traditional ML and DL systems, this study aims to provide a comprehensive synthesis of the non-functional requirements related to these technologies, which remain critical in a wide range of applications. This work follows well-established guidelines [200, 365] and additional search criteria based on seed set identification [249] to comprehensively synthesize existing research. From an initial set composed of over 3,000 hits, and after applying multiple snowballing rounds and additional data collection procedures, I ended up analyzing more than 94,000 research results. Through the application of exclusion/inclusion criteria and quality assessment, I finally selected 130 papers. Additionally, I provide a new catalog consisting of 26 software engineering challenges proposed by researchers to assist practitioners in optimizing ML-enabled systems specifically in relation to non-functional requirements during the software life cycle. These challenges are not about the NFRs themselves but instead arise from the complexities faced by ML-enabled systems when striving to meet specific non-functional requirements. They occur at various stages of the software life cycle, including design, development, implementation, and operation, and are shaped by the socio-technical context and the inherent trade-offs between competing NFRs. I conclude this chapter by elaborating on the implications of the results, along with the actionable items that readers of this work may (re-)use to analyze further the problem of non-functional requirements in machine learning-intensive software systems.

2.2 RESEARCH METHOD

The *goal* of the study was to survey the current research on non-functional requirements of machine learning-enabled systems, with the *purpose* of

[2] This chapter is largely based on De Martino and Palomba [83]. For reasons of space, only the main results and implications are reported here; full methodological details and extended analyses can be found in the original publication.

providing researchers with actionable items and insights that they can exploit to further improve the support provided to machine learning engineers in the *context* of machine learning and deep learning-enabled systems.

The *perspective* is of the broad community of researchers working on software engineering for artificial intelligence (SE4AI). At first, fresh Ph.D. students may be particularly interested in the results since they may be looking for neglected or emerging research areas to investigate in the context of their doctoral studies. Senior researchers may be looking for research opportunities to advance the current state of the art or additional opportunities for research proposals and industrial collaborations. By senior researchers, I mean all academics who may be interested in contributing to advancing the current knowledge on non-functional requirements of ML-enabled systems, hence extending to anybody who may help advance the state of the art in the field. Finally, practitioners - which I identify as professional software engineers, machine learning engineers, project managers, and tool vendors - may be interested in understanding the challenges to face to deal with non-functional requirements, as these may possibly raise obstacles to technological transfer that should be carefully considered in practice.

I designed a systematic literature review (SLR) following the guidelines by Kitchenham et al. [199]. The initial use of a search string focusing on terms such as “NFR” or “non-functional requirement” aimed to let the types of NFRs emerge naturally from the data, indeed following a bottom-up approach. To mitigate threats due to incompleteness of the search, I boosted the search process by means of two additional steps. These steps incorporated a top-down approach, which leveraged implicit understanding of NFRs to guide the identification and selection of relevant studies. This approach ensured the use of broader conceptual insights to complement the systematic search process, addressing potential gaps and improving the overall comprehensiveness of the review. First, I systematically screened the research articles published in top-tier software engineering and artificial intelligence venues to identify seed papers to

further process [367]. Second, I integrated the snowballing procedure [366], i.e., the iterative scanning of the incoming and outcoming references of the primary studies done to identify additional relevant sources of information. The systematic literature review can be considered *"hybrid"* [249], as it combines traditional search strategies with additional search steps and snowballing. In terms of reporting, I followed the *ACM/SIGSOFT Empirical Standards*[3] and, in particular, the *"General Standard"* and *"Systematic Reviews"* guidelines.

2.2.1 *Research Objectives and Questions*

I defined two research questions (**RQ**s): $\mathbf{RQ}_1$ aimed at systematically classifying the non-functional requirements identified by researchers, as well as providing insights into the application domains considered by researchers. By 'application domain', I referred to the specific application areas or contexts in which a non-functional requirement has been investigated, such as healthcare, finance, automotive, and others. Additionally, a domain can also be general, encompassing non-functional requirements that are broadly applicable across any kind of domain without being tied to a specific context. More specifically, I asked:

> **$\mathbf{RQ}_1$ — NFRs of ML-enabled systems**
>
> *What are the non-functional requirements of ML-enabled software considered by researchers and in which domains were they addressed?*

From a *scientific* perspective, by addressing this research question I aimed to contribute to classifying the non-functional requirements of ML-enabled systems [144]. While previous efforts in this respect have been conducted, there is still not a systematic classification - this represents a key contribution to the work. From a *practical* perspective, the classification of non-functional requirements serves three key purposes. In the

[3] The *ACM/SIGSOFT Empirical Standards*: https://github.com/acmsigsoft/EmpiricalStandards.

first place, a clear classification allows to aggregate the current knowledge on the matter, enabling more targeted research and development efforts. Researchers can indeed use the classification to identify specific areas that need further investigation, ensuring that their efforts are focused and aligned with industry needs. Second, the classification provides an instrument to increase the community awareness of the multifaceted aspects affecting the design of ML-enabled systems: researchers and practitioners may use the classification to have a comprehensive overview of the non-functional requirements that should be preserved, potentially identifying areas where trade-offs might be necessary. Finally, as for practitioners, understanding the various non-functional requirements and their classifications aids in the design and implementation of ML-enabled systems. It ensures that all relevant aspects are considered, leading to the development of more robust, reliable, and efficient systems.

RQ$_2$ aimed to aggregate and synthesize the body of knowledge on the challenges faced by ML-enabled systems when addressing non-functional requirements, informing researchers and practitioners on the aspects to further consider while developing these systems.

> **RQ$_2$—NFRs Challenges**
>
> *What are the challenges for ML-enabled systems specifically relating to non-functional requirements?*

From a *scientific* perspective, addressing this research question advances the body of knowledge by systematically aggregating the challenges for ML-enabled systems specifically relating to non-functional requirements, hence proposing a super-set of the challenges identified in previous work. This classification provides a foundational understanding that can guide future research efforts, ensuring that subsequent studies build upon a well-defined framework. It also highlights gaps in the existing literature, pointing researchers towards areas that require further investigation and potentially leading to the development of new theories, models, and methodologies specific to ML-enabled systems.

From a *practical* perspective, understanding the challenges for ML-enabled systems specifically relating to non-functional requirements equips practitioners with crucial insights that can enhance the design, development, and maintenance of these systems. By identifying common challenges, practitioners can develop strategies and best practices to mitigate them, leading to more robust, reliable, and efficient ML-enabled software. Additionally, this knowledge helps in setting realistic expectations and priorities during the software development lifecycle, ensuring that critical non-functional requirements are adequately addressed, thereby improving overall system quality and user satisfaction.

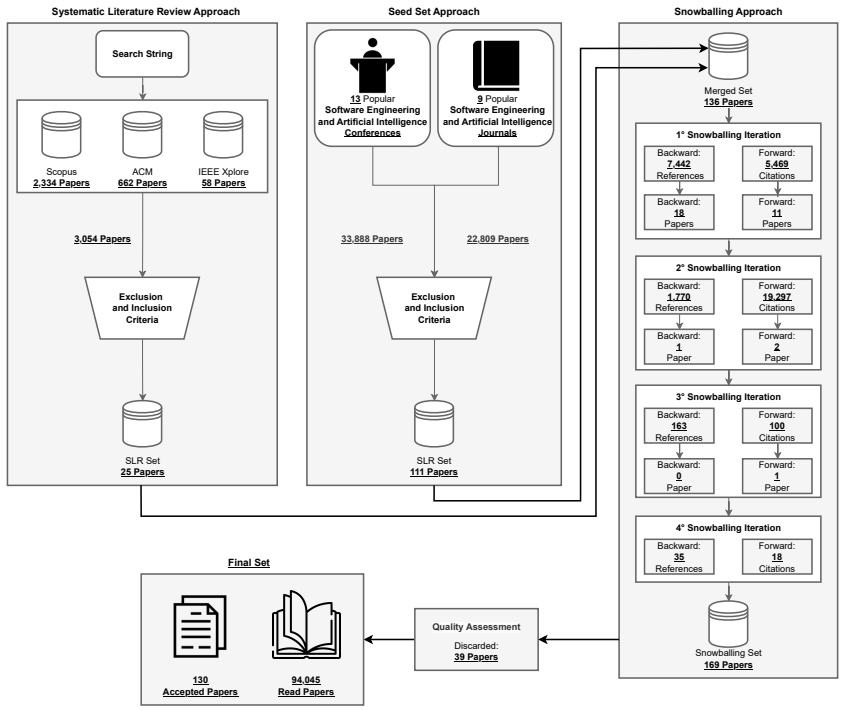


Figure 2.1: Overview of the process for selecting papers.

2.2.2 *Execution of the Research Methods*

Figure 2.1 presents the outcomes of the research methods. The systematic literature review commenced on June 26, 2024, yielding a total of 3,054 hits, predominantly from *Scopus* (2,334), with fewer from *ACM Digital Library* and *IEEEXplore* (662 and 58, respectively).

Upon execution of the search string and the seed search, I imported the resulting articles in a shared EXCEL sheet. At first, the duplicated articles (coming as a result of the search string execution against different databases) were removed, with only one instance retained. Through the meta-data provided by the databases, I could automatically filter out the papers not written in English, short papers, papers not published/peer-reviewed, papers that did not fall into themes of computer science and computer engineering, and papers published before 2012: this process reduced the number of papers to 1,916. Next, each remaining paper was manually scanned to apply the additional exclusion criteria. I verified that each paper was actually accessible, meaning it was available for full-text review. If a paper was not accessible, it was excluded.

For the remaining papers, I analyzed (1) the venue where the paper was published, (2) the article type, and (3) the topic addressed. Regarding the venue, I excluded workshop papers. For the article type, I excluded vision papers, systematic literature reviews, and literature surveys by examining the title, keywords, and, when necessary, the abstract. To verify the scope of the article, I relied on the title and keywords, and read the abstract, introduction, and conclusions when additional clarification was needed. This scanning also allowed us to identify journal articles that extended conference papers, leading to the exclusion of the conference proceedings. In addition, I excluded papers based on whether they addressed any non-functional requirement (**EC10**). As explained in [83], papers were analyzed with respect to the extent to which they discussed quality attributes in relation to the system's operation, performance, or non-functional goals. As the reader may expect, attributes such as classification accuracy and performance were widely discussed in the considered literature; however,

not all papers explicitly or implicitly contextualized these attributes as non-functional requirements. According to this analysis, many papers address these attributes solely in the context of functional goals, such as improving prediction accuracy or optimizing model performance for a specific task, without considering their role as overarching system-level qualities or linking them to the broader concept of NFRs. An example is the paper by Suwonchoochit and Senivongse [327]. The primary focus of the paper is on improving prediction accuracy through algorithmic optimization without addressing accuracy as a broader system-level quality or linking it to non-functional requirements. Specifically, the paper employed various ML algorithms, with Extra Trees and TF-IDF emerging as top-performing models to classify user-reported database-related issues on Stack Overflow, but it does not consider accuracy in terms of trade-offs with other system qualities (e.g., fairness or robustness) or its implications for system deployment and operation. Consequently, while accuracy was a central theme, it was not treated as a system-level non-functional requirement. This lack of contextualization within the broader framework of non-functional requirements led to its exclusion based on **EC10**. On the contrary, let us consider the paper by Biswas and Rajan [27]. This paper was not excluded in this phase - it is actually one of those finally included in this work. The paper explicitly framed accuracy as a system-level quality and linked it to broader non-functional requirements. Indeed, the paper explores fairness in ML models sourced from Kaggle. It aims to evaluate bias in these models, investigate its causes, and analyze the effectiveness and trade-offs of various bias mitigation techniques. By presenting accuracy as a non-functional requirement with system-level implications, the paper was not excluded and led to the following phase, which was concerned with the application of the inclusion criteria.

After completing the exclusion criteria, I proceeded with the inclusion criteria. While the application of these two sets of criteria may be performed simultaneously, I chose to separate them to ensure a thorough and systematic review process. This approach allowed us to focus first on eliminating papers that clearly did not meet the basic requirements before

concentrating on those that potentially fit the study's scope. The inclusion criteria were first assessed by scanning the title, keywords, abstract, introduction, and conclusions of the papers. In the case these pieces of information were not sufficient to establish the suitability of a paper, a full read-through of the entire paper was conducted. The vast majority of the papers analyzed in this stage were removed as a consequence of **IC1** and **IC2**, as they lacked focus on ML-enabled systems or they tangentially mentioned non-functional requirements. According to the findings, most of the papers mentioning terms such as *"Non-Functional Requirement"*, *"Non Functional Requirement"*, and *"NFR"* focused on non-functional requirements of classical software systems, i.e., systems that do not present any machine learning component. Furthermore, most papers addressing specific aspects that could, in principle, be connected to ML-enabled systems, e.g., adversarial sampling, did not explicitly link these aspects to specific non-functional requirements. These papers often lacked the detailed terminology or focused analysis necessary to contextualize their findings within the framework of non-functional requirements in ML-enabled systems. For instance, the paper by Nguyen et al. [257] represents a valid example that explains the impact of **IC1** and **IC2** to assess the relevance of the primary studies considered in the work. The article proposed an adversarial sampling technique to generate training samples for API recommendation systems. However, the paper did not explicitly refer to any specific non-functional requirements in their reporting. Instead, they focused primarily on the technical aspects of the proposed technique without addressing the broader non-functional requirements, such as reliability or security, that this technique could impact. This omission limits the applicability of their findings in the scope of the work, as it prevents a clear understanding of how the proposed technique addresses or impacts specific non-functional requirements critical to ML-enabled systems. As a consequence, it could not be included in the review.

The remaining papers satisfied **IC3**. As for **IC4**, papers that satisfied this criterion either defined a general framework for addressing NFRs across ML-enabled systems or provided a detailed analysis of NFRs within a spe-

cific domain. These papers offered insights into how NFRs were handled, making them relevant to the study. As an example, let's consider the paper by Aminifar et al. [13]. This article provided a general-purpose framework to optimize accuracy and privacy, which would be applicable to any ML-enabled systems, independently from the specific application domain. Similarly, the paper by Hutiri et al. [179] explores the emergence and propagation of biases in on-device ML workflows, emphasizing the impact of design choices made during development. On-device ML, commonly deployed on resource-constrained IoT devices, may impact privacy, energy Consumption, and fairness. Both these papers were included in the study, as they properly reflected the intent of **IC4**. On the contrary, papers that did not satisfy **IC4** typically mentioned NFRs superficially without grounding them in any meaningful context. For instance, a paper that briefly referred to "security" without explaining how it applies to a general or specific domain was excluded.

Overall, the application of the exclusion and inclusion criteria led to the exclusion of 3,054 papers, hence leading to 25 primary studies that were employed in the context of the snowballing process. This latter step was conducted by scanning the titles of the papers with ingoing or outgoing references to the primary studies; the papers that were deemed potentially relevant were then subject to the inclusion criteria to establish their suitability for the study.

The 25 papers passing the inclusion criteria were then subjected to quality assessment, following the procedure described in [83]. As an outcome, 11 papers were finally considered for inclusion in the study: these papers provided detailed discussions, comprehensive analyses, and significant insights into non-functional requirements in ML-enabled systems, thereby contributing meaningfully to the understanding of the topic.

At the same time, the limited amount of relevant papers, due to the *inherent variability in terminology* used in relevant studies, convinced us to augment the search process with additional steps. I, therefore, conducted the (1) seed set search described in [83], which led to the identification of 111 additional primary studies; and (2) the snowballing search described

in [83], which contributed with 136 primary studies more. To be included, the papers coming from these two additional search steps had to satisfy both the exclusion and inclusion criteria, with key discriminators being **IC1** and **IC2**. Specifically, 99.8% of the papers were excluded because they did not explicitly link to non-functional requirements within the title, keywords, abstract, introduction, and conclusions. This stringent application of the criteria was required to consider a manageable amount of resources and ensured that only papers with a clear focus on non-functional requirements in ML-enabled systems were considered, thereby maintaining the relevance and quality of the primary studies included in the review.

The set of 169 papers was then analyzed with a quality assessment process on the full text. The majority of these papers were excluded due to not meeting the thresholds for **Q1** and **Q2**. While many papers scored well on **Q3**, discussing the domain-specific or general-purpose applicability of non-functional requirements, they lacked either a clear and detailed definition of NFRs or robust examples and empirical data to substantiate their claims. The application of these criteria led to the exclusion of 39 articles from the initial set. Summing up the primary studies identified by using the three different methods, the hybrid systematic literature review considered a total of 130 papers, which were finally used to address the research questions.

2.3 ANALYSIS OF THE RESULTS

In this section, I report quantitative and qualitative insights coming from the data extraction and analysis phase where I address the specific research questions of the study. Additional bibliometric data and information considered in this research are in the original paper [83].

2.3.1 RQ_1 - *What are the non-functional requirements of ML-enabled software considered by researchers?*

The first goal of this work was to classify the non-functional requirements of machine learning-enabled systems. In the following section, I report the data synthesis process and the results achieved.

Data Analysis and Synthesis. Based on the data collected through the data extraction form, I addressed $\mathbf{RQ}_1$ by applying a three-step systematic classification exercise through which I (i) elicited the non-functional attributes that they aimed to address, (ii) characterized the non-functional requirements, and (iii) grouped them according to two main classification criteria. In particular, the three steps were conducted as follows:

- *Extraction of Non-Functional Requirements.* I analyzed the primary studies to identify and extract non-functional requirements explicitly mentioned in the papers. This step involved thorough reading and annotation of the relevant sections discussing the non-functional requirements. Whenever possible, i.e., when considering the papers that received a quality score of "Yes", the extraction was relatively straightforward as these resources explicitly referred to the non-functional requirements considered. In the other cases, relevant terms connected to non-functional aspects were extracted and elaborated for consistency. For example, the primary study [149] reported terms like *"memory problems"* and *"battery drain"*, which I used to interpret the context of the study and mapped them to the non-functional requirement named 'energy consumption'.

- *Naming and Describing Non-Functional Requirements.* Each extracted non-functional requirement was assigned a unique name and a detailed description. In this stage, possible naming inconsistencies that emerged in the previous step were solved. The description was intended to capture the essence and scope of the requirement as discussed in the primary studies.

- *Mapping to Reference Class.* The non-functional requirements were then mapped to high-level reference classes based on two classification criteria: (1) the non-functional requirements must have a significant impact on the high-level reference class; (2) non-functional requirements sharing similar purposes or characteristics must be grouped together under a common reference class. For example, non-functional requirements like privacy and security were grouped under the 'Resiliency' class because they both contribute to the system's ability to handle unexpected events and threats.

The names assigned to the classes, the description of each non-functional requirement, and the classification exercise as a whole, were informed by different sources of the existing body of knowledge, i.e., by the primary studies collected in this thesis, other resources of the state of the art not directly related to the research goals, and the *Systems & Software Quality Requirements & Evaluation* (ISO/EIC 25010).

I (1) analyzed the primary studies to elicit the non-functional attributes considered and (2) provided a preliminary version of the taxonomy. This preliminary taxonomy aimed to homogenize definitions and classifications by leveraging existing knowledge and favoring well-established class names whenever possible, thus providing a more comprehensive and usable taxonomy. The preliminary taxonomy was later subject to evaluation. The second author joined the process at this stage, and a discussion was opened on the produced taxonomy. The two authors discussed the consistency of the classification, as well as the names and descriptions assigned. Modifications were made where necessary—this occurred in four cases, where some non-functional requirements were grouped differently, and the names assigned to the classes were adjusted.

Addressing RQ$_1$. In the first place, as a result of this procedure I could identify a total amount of 31 non-functional requirements.Interestingly, in this systematic exercise could identify not only non-functional requirements that emerged already in previous studies [144, 146, 161], but also additional categories that were not previously pointed out, i.e., *transferability*, *cost*, *accountability*, *energy consumption*, *sustainability*, *capacity*,

stability, *behavioral*, *imperceptibility*, *efficiency*, *availability* and *adaptability*.

Table 2.1: Non-Functional Requirements in ML-enabled systems. The *'Meaning'* column reports whether the non-functional requirements are *'new'* (N), i.e., they are specific to ML-enabled systems, are *'altered'* (A), i.e., their meaning is different with respect to the definition of the non-functional requirements in non-ML-enabled systems, or are the *'same'* (S), i.e., their meaning is the same as the non-functional requirements in non-ML-enabled systems.

Cluster	NFRs	Meaning	Definition
Accuracy	Accuracy	N	The degree to which a model's predictions match the actual values.
Efficiency	Performance	S	The ability of a system to perform actions within defined time or throughput bounds.
	Capacity	A	The amount of space required to store the model and any associated data.
	Stability	N	Degree to which the output of a model varies as a consequence of perturbations to its input.
	Scalability	A	The capability to handle increased workloads by adding resources while maintaining or enhancing model performance.
Maintainability	Replaceability	N	The degree to which a model can be replaced or substituted with another model without significant changes to the system.
	Retrainability	N	The degree to which a model can be retrained on new data without significant performance loss.
	Reproducibility	N	The degree to which a model's results can be reproduced by others using different software or hardware.
	Transferability	N	The degree to which a model trained on one data set can be applied to another with similar characteristics.
	Reusability	A	The degree to which a model can be reused in different applications or contexts.
	Adaptability	A	The ability of the model to adapt to changing requirements or environments.
	Traceability	S	The ability to trace work items across the development lifecycle.
Resiliency	Security	A	The degree to which a model and its associated data are protected against unauthorized access, modification, or theft.
	Safety	A	The degree to which a model and its outcomes are safe for humans and the environment.
	Privacy	A	The degree to which a model and its associated data protect individuals' privacy rights and comply with data protection regulations.
	Robustness	A	The ability of a model to maintain its performance when faced with uncertainties or adversarial conditions.
	Reliability	A	Degree to which a model is resilient to errors and to variations of the surrounding environment.
	Behavioral	A	The degree to which a model's outcomes align with requirements and expectations.
	Flexibility	A	The degree to which a model can adapt to input data or environment changes without significant performance degradation.
	Availability	A	The degree to which a model is operational and accessible when needed, without significant downtime or interruption.
Sustainability	Fairness	N	The degree to which a model produces unbiased predictions and decision-making outcomes across different groups of individuals.
	Ethics	N	The degree to which a model mitigates potential societal risk.
	Accountability	N	The degree to which individuals or organizations are held responsible for the actions of the model and its outcomes.
	Cost	A	The overall economic means required to develop and maintain an ML-enabled system.
	Energy Consumption	A	The amount of energy required for training and inference of the model and its impact on the systems.
Usability	Interpretability	N	The degree to which a model's predictions and decision-making process can be explained in terms of causality or human-understandable concepts.
	Imperceptibility	S	The system's ability to produce outputs that are indistinguishable from what a human would produce in the same scenario.
	Explainability	A	The degree to which a model's predictions can be explained and understood by humans.
	Transparency	A	The degree to which a model's inner workings and decision-making process can be understood and evaluated by humans.

The classification exercise led to the identification of six main classes of non-functional requirements: Table 2.1 reports, for each class, the set of non-functional requirements belonging to the class, along with their definition and meaning. The column 'Meaning' reports three symbols explaining whether each non-functional requirement can be considered

(1) new with respect to traditional non-functional requirements previously investigated ('New'), (2) not new but with altered meaning ('Altered'), or (3) not new but with the same meaning ('Same'). More specifically, I identified the following classes:

ACCURACY. The first class contained only the accuracy requirement, namely the degree to which a model's predictions match the actual values. This represents a new category of non-functional requirement that is specific to ML-enabled systems. I considered accuracy as a non-functional property for two main reasons. In the first place, multiple papers included in this systematic literature review, e.g., [144, 146, 161], described accuracy as a non-functional requirement, defining it as the number of correctly predicted data points out of all the data points. As this work aims at synthesizing the current knowledge available on the matter, I preferred to be conservative and considered accuracy as reported in the state of the art. In the second place, the definition is in line with the concept of non-functional requirement, i.e., *"a condition that specifies a criterion that may be used to judge the operation of a system rather than specific behaviors"* [33]: as a matter of fact, accuracy does not define constraints on the specific functionality that a system should enable, but rather question how the system performs in terms of correctness of the predictions made. In other terms, accuracy represents an attribute that can be used to judge the operation of a system rather than its specific behavior, i.e., it is by definition a non-functional requirement. The class was made isolated because of two main reasons: (1) accuracy represents the key feature to optimize by ML-enabled systems; (2) the non-functional requirement cannot be conceptually compared to any other, being "unique".

EFFICIENCY. This was concerned with the overall performance and effectiveness of ML-enabled systems. The ISO/IEC 25010 standard includes performance efficiency as one of its main characteristics, encompassing attributes such as time behavior, resource utilization, and capacity. As such, I exploited the same reasoning to cluster non-functional require-

ments such as *performance*, *capacity*, *stability*, and *scalability*, under the *"Efficiency"* class. While the definition of *performance* is the same as the one available for non-ML-enabled systems, I classified a novel requirement, *stability*, and altered the meaning of *capacity* and *scalability*. More specifically, *stability* is a new non-functional requirement for ML-enabled systems due to the dynamic nature of machine learning models, which can undergo continuous updates and changes in response to new data. This dynamic aspect requires systems to maintain consistent performance and reliability over time, even as models evolve. The meanings of *capacity* and *scalability* are different for ML-enabled systems compared to traditional systems. In the context of ML-enabled systems, *capacity* not only refers to the system's ability to handle large volumes of data and numerous concurrent users but also encompasses the system's capability to manage the computational demands of inference tasks in terms of storage and deployment. This includes the ability to process complex models efficiently and to scale resources dynamically as computational requirements fluctuate. Similarly, *scalability* for ML-enabled systems involves more than just the traditional scaling of resources to accommodate increased load. It also includes the ability to scale machine learning models themselves, such as adjusting model complexity or deploying multiple models to handle different tasks. Scalability in ML-enabled systems must account for the need to distribute models across multiple nodes and ensure that inference can be performed efficiently across distributed environments. These additional layers of complexity necessitate a redefined understanding of capacity and scalability within the realm of ML-enabled systems.

MAINTAINABILITY. Maintainability is chosen as a cluster to represent *replaceability*, *retrainability*, *reproducibility*, *transferability*, *reusability*, *traceability* and *adaptability*: it emphasizes the ability of a system to adapt and evolve over time. All these non-functional requirements are critical to ensure that the system can be maintained and updated as needed and that it can be easily adapted to new use cases or environments. All the non-functional requirements in this cluster, with the only

exception of *traceability*, assume a different meaning with respect to non-ML-enabled systems. *Reusability* in ML-enabled systems involves the reuse of machine learning models and components in different systems. This includes the ability to transfer knowledge from one model to another, leveraging pre-trained models to reduce development time and improve performance in new applications. *Adaptability* refers to the system's ability to dynamically adjust to new data and changing conditions. In ML-enabled systems, this often means the ability to update models with new data to improve accuracy and relevance over time, as well as the capacity to integrate with new technologies and frameworks as they emerge. *Replaceability* is a new requirement that addresses the need to seamlessly replace or upgrade machine learning models and components without disrupting the overall system functionality. This is crucial in ML-enabled systems where models may need frequent updates or replacements to stay effective. *Retrainability* is another new requirement, emphasizing the system's capability to retrain models efficiently as new data becomes available. This ensures that the system remains accurate and up-to-date, reflecting the latest information and trends. *Reproducibility* is critical in ML-enabled systems to ensure that experiments and model training processes can be reliably repeated with the same results. This is essential for validating model performance and for iterative development. Finally, *transferability* refers to the system's ability to apply models and knowledge from one domain or task to another. This involves leveraging transfer learning techniques to adapt pre-trained models for new tasks, improving development efficiency and performance. By prioritizing maintainability, designers and developers can create reliable, efficient, adaptable, flexible, and sustainable systems over the long term.

RESILIENCY. Resilience represents *robustness*, *reliability*, *behavioral*, *flexibility*, *security*, *safety*, *privacy*, and *availability*, as it emphasizes the ability of a system to adapt and recover from adverse events while maintaining its functionality and performance. Robustness, reliability, and behavioral flexibility are essential to ensure the system can operate

in various conditions and remain responsive to changing circumstances. Security, reliability, and confidentiality are also critical to ensure the system protects users and stakeholders from harm, including threats to their physical safety, personal information, and data confidentiality. In this cluster, the definition of all the non-functional requirements required to be adapted for ML-enabled systems. Specifically, *robustness* in ML-enabled systems refers to the system's ability to handle diverse and potentially unforeseen data inputs and scenarios without failing. This involves ensuring that models can generalize well to new, unseen data and are resistant to noise and adversarial inputs. *Reliability* is crucial for ML-enabled systems to maintain consistent performance over time. This includes the dependability of the model predictions and the ability to maintain performance despite variations in data quality or environmental conditions. *Behavioral* refers to the degree to which a model's outcomes align with the predefined requirements and expectations. This involves rigorous validation and testing to confirm that the model works as intended by the initial requirements. *Flexibility* refers to the system's capacity to adapt its behavior dynamically in response to changing environments and data patterns. For ML-enabled systems, this means models must be capable of adjusting their outputs on new inputs to remain effective. *Security* in the context of ML-enabled systems involves protecting the integrity of the model and data from malicious attacks. This includes safeguarding against attacks designed to manipulate model results and ensuring that the model or associated data is not stolen by attackers. *Safety* is a non-functional requirement that ensures the system does not cause harm to users or the environment. For ML-enabled systems, this involves ensuring that model decisions do not result in unsafe conditions, particularly in critical applications such as autonomous driving or healthcare. *Privacy* is critical in ML-enabled systems to protect users' personal information. This involves implementing techniques like differential privacy to ensure that individual data points cannot be reverse-engineered from the model outputs. Last but not least, *availability* ensures that the ML-enabled system remains

accessible and operational, even under high load or during adverse conditions. This involves designing models and systems that can handle scalability challenges and remain responsive during peak usage times. By prioritizing resilience, designers and developers can build reliable and secure ML-enabled systems that can withstand disruptions and threats while maintaining their core functions and services. This altered understanding of the non-functional requirements within the resilience cluster highlights the unique challenges posed by integrating machine learning into complex systems.

SUSTAINABILITY. Sustainability is chosen as a cluster to represent economic, social, and environmental aspects. It ensures that resources are used efficiently and effectively, reducing the environmental impact, *energy consumption* and financial *costs* associated with producing the model and making inferences. Additionally, sustainability is chosen to represent *fairness*, *ethics*, and *accountability*, as these aspects are critical for ensuring that the system benefits all stakeholders, including marginalized and vulnerable communities, and operates in a way that aligns with ethical principles and values. The rationale to classify social and ethical concerns under the *"Sustainability"* category comes from the increasing recognition that social sustainability is currently experiencing. For instance, the United Nations included the reduction of inequalities among the 17 objectives for sustainable development.[4] Researchers have also been arguing that fairness and ethical concerns should be considered as a form of sustainability. McGuire et al. [242] advocated that social sustainability refers to multiple dimensions, including pro-social vs. anti-social affordances. On a similar note, various other researchers [22, 89, 206] argued to consider social and ethical aspects as sustainability properties. Overall, sustainability can be considered a guiding principle for designing and developing efficient, cost-effective but also equitable, ethical, and accountable systems. When comparing the definition of these non-functional requirements to those of non-

[4]The United Nations Goals for Sustainable Development: https://sdgs.un.org/goals.

ML-enabled systems, it is worth pointing out that some were altered to better fit the context of ML-enabled systems, while others represent new aspects specific to ML-enabled systems. In particular, *energy consumption* and *costs* assume a slightly different meaning when considered in this context. Indeed, the former refers not only to the energy used during the operation of the system but also to the significant resources required during the inference of machine learning models. This includes the computational power needed to make model predictions, which can have a considerable environmental impact. As for the latter, it includes both the financial costs associated with developing, deploying, and maintaining ML models, and the long-term operational costs. This encompasses costs related to data storage, processing power, and the need for specialized hardware. The other non-functional requirements are instead new. *Fairness* is a new requirement that addresses the need to ensure that ML-enabled systems operate without bias, providing equitable outcomes for all users. This includes implementing strategies to mitigate algorithmic bias and ensuring diverse and representative training data. *Ethics* refers to the principles guiding the design and deployment of ML-enabled systems to ensure they operate in a morally responsible manner. This includes considerations of the potential societal impacts of deploying these technologies. Finally, *accountability* involves ensuring that the actions and decisions made by ML-enabled systems can be traced and audited. This includes implementing mechanisms for tracking data provenance, model decisions, and providing transparency in how models operate and make decisions. By incorporating these redefined and new non-functional requirements under the sustainability cluster, designers and developers can create ML-enabled systems that are not only efficient and cost-effective but also socially and ethically responsible. This holistic approach ensures that the systems developed are sustainable in the broadest sense, benefiting the environment, economy, and society at large.

USABILITY. This cluster has non-functional requirements with unique definitions because of the black-box nature of ML-enabled systems - it

is similar to the one defined by Habibullah et al. [144]. More particularly, this class was chosen as the cluster to represent *interpretability*, *imperceptibility*, *explainability*, and *transparency*: it is concerned with ensuring that a system is easy to use and understand for end users, reducing the cognitive load required to interpret the model's behavior, and making it more transparent and explainable. In this cluster, some non-functional requirements are redefined to better fit the context of ML-enabled systems, while others retain their original meaning, and new requirements are introduced. More particularly, *imperceptibility* retained its original meaning, referring to the system's ability to integrate seamlessly into the user's workflow without causing disruptions or requiring significant changes to existing processes. *Interpretability* is a new requirement that refers to the degree to which a human can understand the cause of a decision made by a model. This involves making the model's workings understandable to users, which is crucial given the complex and often opaque nature of machine learning algorithms. *Transparency* is a non-functional requirement that, while traditionally associated with characteristics of systems and processes in non-ML contexts, in ML-enabled systems focuses on providing clear insights into how the model functions, what data it uses, and the logic behind its decision-making process. This includes documenting the model's design, training data, and any preprocessing steps, ensuring users can trace and understand the model's operations. As for *explainability*, its meaning is altered to involve providing users with understandable and actionable explanations of the model's outputs. This goes beyond mere transparency by ensuring that users can comprehend and trust the decisions made by the system and understand the reasoning behind those decisions. By incorporating these specific non-functional requirements under the usability class, I address the unique challenges posed by the black-box nature of ML-enabled systems. This ensures that such systems are not only effective in their operations but also user-friendly, transparent, and trustworthy, ultimately enhancing their overall usability and acceptance among end users.

Researchers can use the taxonomy built in the context of the work to have a comprehensive mapping of the relevant non-functional requirements to optimize when developing ML-enabled systems, other than understanding the quality and efficiency aspects practitioners should focus on when releasing machine learning models.

> **RQ_1 — NFRs of ML-enabled systems**
>
> *As a result of RQ_1, I could identify a total of 31 relevant non-functional attributes, classified according to six main categories. I also identified the most frequent ML tasks and ML domains where these non-functional attributes have been investigated. The resulting taxonomy maps the current knowledge on the matter and provides insights into the research areas that may be worth exploring in the future.*

2.3.2 RQ_2 - *What are the challenges for ML-enabled systems specifically relating to non-functional requirements?*

The second research question focuses on identifying the challenges faced by ML-enabled systems specifically relating to non-functional requirements. In the following, I summarize the data analysis process and results that address RQ_2.

Data Analysis and Synthesis. Similarly to the previous research question, I addressed RQ_2 by means of a three-step systematic classification exercise that aimed at (i) extracting the challenges for ML-enabled systems specifically relating to non-functional requirements, as reported in the primary studies, (ii) synthesizing them by grouping similar challenges, and (iii) refining the groups initially produced. In this respect, it is worth remarking that I deliberately did not include all challenges faced by ML-enabled systems but focused solely on those that directly impact the achievement, management, or optimization of specific NFRs. These challenges were derived exclusively from the primary studies selected as part of the systematic literature review, ensuring that all included challenges

Table 2.2: Challenges Related to Non-Functional Requirements in ML-Enabled Systems.

Cluster	Challenge	References
Efficiency	E.1 - Dealing with High Latency	[59–61, 220, 227, 279]
	E.2 - Dealing with Space	[149, 179, 216, 220, 227, 303, 362, 403]
	E.3 - Managing Simulation for Inference	[149, 179, 216, 303]
Maintainability	M.1 - Increasing Model Reproducibility	[55]
	M.2 - Increasing Model Decomposition and Reuse	[55, 181, 266, 267, 279, 289]
Resiliency	R.1 - Resilience to Adversarial Attacks	[30, 100, 174, 379, 382, 383]
	R.2 - Exploiting the Sensitivity of Adversarial Attacks	[16, 30, 44, 113, 116, 141, 379]
	R.3 - Security Verification of Pre-Trained Models	[176, 177, 218, 223, 396]
	R.4 - Resilience to Intellectual Property Theft	[131, 177]
	R.5 - Optimal Post-Deployment Simulation	[102, 183, 213, 217, 230, 240, 395]
	R.6 - Preserving Privacy in Machine Learning-Enabled Systems	[13, 310]
Sustainability	S.1 - Dealing with Algorithmic Discrimination	[27, 50, 63, 140, 331, 370, 388]
	S.2 - Model Accountability	[37, 338, 357]
	S.3 - Fairness Analytics	[58, 64, 184, 273, 310, 317, 326, 338, 386, 391, 398]
	S.4 - Improving Sustainability Benchmarks	[37, 378]
	S.5 - Reducing Energy Cost	[59, 114, 149, 150, 216, 220, 224, 315, 340, 342, 362]
	S.6 - Increasing the Practitioner's Awareness of Sustainability	[114, 149, 303]
Usability	Usability	[8, 69, 92, 94, 142, 151, 170, 173, 220, 261, 404]
Cross-cutting challenges	C.1 - Dealing with Internal Errors	[101, 110, 149, 169, 215, 220]
	C.2 - Diagnosing the Internal Behavior of Models	[5, 95, 97, 153, 172, 229, 333, 371, 384, 389, 397]
	C.3 - Context-based Trade-off Identification	[43, 55, 63, 112, 310]
	C.4 - Prioritizing and Balancing Non-Functional Requirements	[58, 63, 94, 114, 149, 179, 184, 326, 342, 388, 391, 403]
	C.5 - Integration and Interaction of ML Models	[8, 111, 272, 321, 363]
	C.6 - Improving Practitioners' Knowledge	[77, 145, 273, 315]
	C.7 - Software Analytics for Non-Functional Requirement Assessment	[59, 150, 151, 259, 376]
	C.8 - Improving the Generalizability of Existing Automated Approaches	[51, 112, 163, 335]

are traceable to evidence provided in the reviewed studies. Consequently, challenges that are not discussed in these papers, even if potentially relevant to NFRs, could not be included. The three steps were conducted as follows:

- *Data Extraction.* For each primary study, I extracted information about the nature of the challenges for ML-enabled systems specifically relating to NFRs, their underlying causes, potential impacts, and any specific examples or case studies provided. In this respect, it is worth remarking that only challenges explicitly linked to NFRs were included in this analysis. While the primary studies may discuss a broader range of issues, this focus remained on extracting and analyzing challenges directly associated with NFRs. For example, in studies addressing algorithmic discrimination, I analyzed challenges such as mitigating disparate impacts or ensuring equitable model performance—both of which are closely tied to the NFRs of fairness and accuracy. Conversely, challenges mentioned in the papers but

unrelated to NFRs, such as purely technical optimizations without a clear NFR connection, were excluded. Similarly, I excluded generic ML system challenges that do not intersect with NFRs. For instance, issues like hyperparameter tuning or general convergence optimization were excluded unless the primary studies explicitly linked these issues to their impact on a specific NFR.

- *Synthesis of Challenges.* I synthesized the extracted data by grouping similar challenges together and identifying common themes. This process involved analyzing the context in which each challenge was presented and understanding the broader implications of the challenges described for ML-enabled systems.

- *Validation and Refinement.* The synthesized challenges were initially extracted by me. Subsequently, these challenges underwent a review and validation process through discussions with the second author. These discussions aimed to ensure consistency and accuracy in the categorization and description of the challenges. During this stage, only minor modifications were made to the names and descriptions to enhance clarity and coherence.

Addressing RQ$_2$. I identified 26 software engineering challenges for this research community summarized in Table 2.2. First and foremost, it is worth pointing out that most of the challenges identified pertained to the use/integration of neural networks within more complex software systems. This was somehow expected, as deep learning has become the most widely employed form of machine learning to empower the capabilities of traditional software systems [118]. In the second place, accuracy represents a cross-cutting concern, i.e., all the challenges identified have implications for the accuracy of ML-enabled systems. For this reason, I preferred to discuss accuracy while presenting the challenges connected to the other classes of NFRs identified in **RQ$_1$**. The list of challenges is presented below.

EFFICIENCY (E): In terms of the overall performance and effectiveness of ML-enabled systems, I could elicit three major challenges, which

are connected to internal errors, high latency, and memory issues. Specifically:

- **Challenge E.1 - Dealing with High Latency [59–61, 220, 227, 279].** This challenge is connected to *performance* requirements, hence affecting the efficiency of ML-enabled systems.

  Nature of the Challenge. Dealing with high latency in ML-enabled systems, particularly in neural networks, is a significant challenge. This latency can arise from various design choices and optimization strategies that aim to improve accuracy but may inadvertently increase processing time.

  Possible Underlying Causes. Multiple design choices related to *hyper-parameter tuning*, *model optimization*, and *model architecture* can affect the overall efficiency of neural networks. These decisions, while sometimes leading to increased accuracy, can result in higher latency. Additionally, the computational efficiency of DNN systems is very sensitive to even slightly different inputs. Changes to the inputs may result in significantly higher computational demand, deteriorating the overall models' efficiency.

  Potential Impacts: High latency can severely impact the usability and performance of ML-enabled systems, leading to slower response times and decreased user satisfaction. It can also affect the real-time processing capabilities of the system, making it unsuitable for applications requiring quick decision-making.

  Target Research: Researchers in the fields of *software analytics*, *software architecture*, and *software quality and optimization* can help address this challenge by developing analytical and code quality tools aimed at finding the right compromise between accuracy and efficiency. This includes conducting preliminary studies to identify the best configurations for specific use cases and optimizing the computational efficiency of neural networks to handle varying input conditions effectively.

- **Challenge E.2 - Dealing with Space [149, 179, 216, 220, 227, 303, 362, 403]**. This challenge affects both *performance* and *capacity* of ML-enabled systems.

  Nature of the Challenge. A critical challenge identified is the space required by machine learning models, particularly neural network-based solutions. These models can be very large, leading to significant memory issues.

  Possible Underlying Causes. The large size of neural network models is particularly relevant in contexts like edge computing and IoT. Deploying these large models on such devices often results in memory constraints that can affect overall system performance.

  Potential Impacts: Memory constraints caused by large models can lead to degraded performance, reduced responsiveness, and the inability to deploy models on resource-limited devices such as those used in edge computing and IoT domains.

  Target Research: Designers should apply policies and strategies to build reduced models before deploying them on devices to mitigate memory issues and improve overall performance. This involves leveraging techniques in the emerging field of *tiny machine learning* [221], which combines hardware, algorithms, and software to support on-device sensor data analytics at low power. While some initial efforts have been made in the artificial intelligence research community (e.g., [91, 312]), there is a notable lack of knowledge and research from the software engineering perspective. Further research is needed to develop efficient and scalable solutions to this challenge, ensuring that ML-enabled systems can operate effectively in resource-constrained domains.

- **Challenge E.3 - Managing Simulation for Inference [149, 179, 216, 303].** This challenge impacts *performance* requirements of ML-enabled systems.

Nature of the Challenge. The selection of devices for deploying an ML model can significantly impact its *performance*, particularly for edge models used in ML inference, where *latency* and *power consumption* are critical factors. Traditional methods and existing benchmarks often fail to provide a comprehensive and reproducible evaluation of these non-functional requirements across various edge platforms and ML models.

Possible Underlying Causes. This challenge is further complicated by the diversity of edge hardware and associated software toolkits, which necessitate benchmarks that can handle different configurations and optimizations. The lack of standardized benchmarks that accommodate the wide range of model and platform combinations makes it difficult to evaluate performance consistently.

Potential Impacts: These challenges affect ML-enabled systems by making it difficult to evaluate a wide range of model and platform combinations. This complicates developers' ability to make informed decisions for specific deployment scenarios, potentially leading to suboptimal performance, increased latency, or excessive power consumption, especially on edge or IoT devices with hardware limitations.

Target Research: Researchers in the fields of *software analysis* and *software quality* should develop tools that simulate model efficiency before deployment in physical domains. This is especially crucial for edge or IoT devices, where balancing competing objectives, such as detection accuracy, frame processing rate, and energy consumption, is particularly challenging. By focusing on creating robust simulation tools, researchers can help developers better evaluate and optimize ML models for specific hardware configurations, ultimately improving the deployment and performance of ML-enabled systems in resource-constrained domains.

🔍 MAINTAINABILITY (M): When it turns to the ability of ML-enabled systems to be adapted and evolved over time, I could elicit two main

challenges connected to model reproducibility and model decomposition and reuse.

- **Challenge M.1 - Increasing Model Reproducibility [55].** It revolves around the clusters of *"Accuracy"* and *"Maintainability"*, with *accuracy* and *reproducibility* impacting ML-enabled systems:

  Nature of the Challenge. ML-enabled systems based on deep learning models are known to be complex and difficult to reproduce accurately. Ensuring that these models can be reliably reproduced is a significant challenge in the field.

  Possible Underlying Causes. Reproducing deep learning models can be challenging because of randomness, which can affect the behavior of the algorithms used. Additionally, hardware nondeterminism, such as that present in graphics processing units (GPUs), can further complicate the reproducibility of deep learning models.

  Potential Impacts: The inability to accurately reproduce deep learning models can lead to inconsistent results, making it difficult to validate and verify model performance. This inconsistency undermines the trust in model outputs and hinders the ability to build upon previous research or deploy models in production environments with confidence.

  Target Research: Researchers in the filed of *software quality* and *software testing* should focus on designing mechanisms that support the *verification* of deep learning models. This can be achieved, for instance, by injecting elements of randomness and non-determinism to observe how the system behaves under those conditions. By understanding and controlling for these variables, researchers can improve the reproducibility of deep learning models, ensuring that they produce consistent and reliable results across different runs and hardware configurations. This will enhance the reliability and trustworthiness of ML-enabled systems in practical applications.

- **Challenge M.2 - Increasing Model Decomposition and Reuse [55, 181, 266, 267, 279, 289].** This challenge affects non-functional requirements such as *replaceability*, *adaptability*, and *reproducibility* of ML-enabled systems:

  Nature of the Challenge. When building and improving ML-enabled systems based on deep learning models, there is the need to reuse parts of previously constructed models, which is essential for maintaining and evolving models efficiently.

  Possible Underlying Causes. This challenge arises when attempting to replace potentially defective parts with others or when selectively reusing components of a model. Identifying the parts responsible for each output class or module in the models can be difficult due to the intricate and often opaque nature of deep learning architectures. Additionally, the properties of randomness and non-determinism in both software and hardware components further complicate this process.

  Potential Impacts: The inability to effectively decompose and reuse model components can lead to inefficiencies in model development and maintenance. It can also result in increased costs and longer development times, as engineers may need to rebuild models from scratch rather than reusing existing components. Furthermore, it hampers the flexibility to quickly adapt models to new tasks or improve specific functionalities.

  Target Research: Researchers should focus on developing methods for component *verification* and identifying reusable components through *program slicing* and *refactoring*. These approaches can help ML engineers accurately extract parts of models for reuse and replace defective or outdated components efficiently. By improving the understanding of the sources of variability and non-determinism, and enhancing reproducibility, it becomes easier to decompose models into reusable units. This not only increases the efficiency of model development and maintenance but also

ensures that ML-enabled systems can be more easily and reliably updated and adapted to meet new requirements.

RESILIENCY (R): Improving machine learning models' universal robustness and security presents key issues [394]. This was the class of non-functional requirements where I identified more challenges (7); these were connected to various aspects such as adversarial attacks, theft of model and intellectual property, attack and defense of models, repairing internal models, post-deploy issues, preserving privacy, and vulnerability transferability.

- **Challenge R.1 - Resilience to Adversarial Attacks [30, 100, 174, 379, 382, 383].** This challenge impacts *"Accuracy"* and *"Resiliency"*, particularly concerning *accuracy*, *robustness*, and *security* for ML-enabled systems:

  Nature of the Challenge. Machine learning models, particularly deep neural networks, are vulnerable to adversarial attacks. These attacks manipulate input data to cause the neural network to produce incorrect outputs, significantly impacting the model's *accuracy* and overall *robustness*.

  Possible Underlying Causes. Different types of adversarial attacks can be targeted or untargeted and can be carried out through various techniques such as *adversarial examples* [328], *adversarial perturbation* [20], *poisoned data* [171], and *backdoor samples* [139]. These attacks exploit vulnerabilities in the model, leading to incorrect predictions and reducing the model's reliability.

  Potential Impacts: Adversarial attacks can severely degrade the performance and reliability of ML-enabled systems. They can cause models to make erroneous decisions, which in critical applications such as autonomous driving, healthcare, or security systems, could lead to catastrophic consequences. This challenge emphasizes the need for robust defenses and improved resilience to maintain system integrity under attack.

Target Research: Researchers in the field of *software testing* and *software security* should focus on developing instruments that can design perturbations to generate adversarial examples for different machine learning models, especially in computer vision problems. These tools would help practitioners understand how models can be affected by vulnerabilities and enhance methods to defend against adversarial attacks. Additionally, the focus should be on the *verification and validation* of ML-enabled systems. While the artificial intelligence community has targeted the *security* perspective by developing new algorithms and techniques [211, 360], complementary software engineering research could propose novel verification and validation approaches. These approaches would improve the strategies and policies used by practitioners to test for security, ultimately enhancing system robustness and reliability against adversarial attacks.

- **Challenge R.2 - Exploiting the Sensitivity of Adversarial Attacks [16, 30, 44, 113, 116, 141, 379].** This challenge focuses on non-functional aspects related to *security*, *robustness*, and *availability* of ML-enabled systems:

  Nature of the Challenge. A complementary perspective to dealing with adversarial attacks is understanding the sensitivity of certain adversarial attacks to input distortions and how this sensitivity can be exploited to enhance the *security*, *robustness*, and *availability* of machine learning models.

  Possible Underlying Causes. In computer vision applications, attacks made through non-additive noise or geometric morphing can cause significant security concerns. These attacks are sensitive to input distortions, which means that automatic correction instruments could potentially distort the malicious inputs, thereby mitigating their effects on the machine learning model.

  Potential Impacts: The ability to exploit the sensitivity of adversarial attacks to distortions can enhance the security and robustness

of ML models, making them more resistant to adversarial manipulation. This can reduce the risk of incorrect model outputs due to malicious attacks and improve the reliability and availability of ML-enabled systems.

Target Research: Researchers in the broad areas of *software testing* and *software security* should focus on developing techniques that can handle perturbations not only occurring naturally in the physical environment but also those generated maliciously by adversarial attacks. These perturbations are typically stealthy and undetectable, reducing the *flexibility* and *behavioral robustness* of deep neural networks. Characterizing these perturbations is a crucial first step for researchers in *empirical software engineering* and *automatic program repair*. By understanding the nature of these adversarial perturbations, researchers can develop methods to automatically correct or distort the malicious inputs, thereby enhancing the overall security and robustness of machine learning models against adversarial attacks.

- **Challenge R.3 - Security Verification of Pre-Trained Models [176, 177, 218, 223, 396].** This challenge revolves around the non-functional related to *security* and *privacy* of ML-enabled systems:

  Nature of the Challenge. Pre-trained models can be highly beneficial as they reduce the computational burden of training complex deep learning models through transferability. However, they are susceptible to security vulnerabilities due to dataset displacement and the potential for malicious code or exploitation.

  Possible Underlying Causes. The displacement of the dataset used for pre-training and the potential inclusion of malicious code during pre-training can introduce security risks in pre-trained models. These vulnerabilities can compromise the overall *security* of the ML-enabled system.

  Potential Impacts: Using pre-trained models without proper security verification can lead to the deployment of models that are

vulnerable to attacks, potentially resulting in compromised system integrity and performance. This can have severe consequences, especially in critical applications where security is paramount.

Target Research: Research in the area mainly targets *software analytics*, *software quality*, and *verification and validation* researchers. At first, research should focus on developing software analytics instruments that can provide practitioners with security insights and best practices for selecting the most appropriate pre-trained models to avoid security concerns. Additionally, the definition of design practices for developing and retraining security-aware fine-tuned models is crucial. Existing research highlights the need for tools and methodologies that can assess the security risks of pre-trained models and ensure their reliability. Software engineering researchers may contribute significantly to addressing these challenges by developing and implementing security verification processes for pre-trained models, thus enhancing their safety and robustness in practical applications.

- **Challenge R.4 - Resilience to Intellectual Property Theft [131, 177].** This challenge mostly affects *privacy*, *robustness* and *safety* of ML-enabled systems:

  Nature of the Challenge. ML-enabled systems based on neural networks are vulnerable to data breaches and unauthorized access to sensitive data due to training attacks. These attacks can involve malicious manipulation of training data [354], undermining the *robustness*, *safety*, and *privacy* of the system.

  Possible Underlying Causes. Training attacks, which target the data used to train machine learning models, can lead to unauthorized access and intellectual property theft. These attacks exploit weaknesses in the security protocols of ML-enabled systems, particularly during the training phase.

  Potential Impacts: Such vulnerabilities can result in the theft of proprietary algorithms and sensitive data, causing significant fi-

nancial and reputational damage. It can also lead to the unauthorized use or modification of models, compromising their integrity and effectiveness.

Target Research: According to existing research, there is limited knowledge on securing machine learning models against these types of attacks. Greater attention should be paid to strengthening security protocols. While this issue primarily concerns researchers in the field of networks and security, the software engineering research community can also contribute by defining security best practices that protect models from intellectual property theft. This is particularly crucial for edge devices and IoT systems, where the risk of unauthorized access is higher [372]. By developing robust security measures and protocols, software engineers can help ensure that machine learning models are safeguarded against intellectual property theft, thereby enhancing the overall resilience and trustworthiness of ML-enabled systems.

- **Challenge R.5 - Optimal Post-Deployment Simulation [102, 183, 213, 217, 230, 240, 395].** This challenge impacts both *robustness* and *safety* of ML-enabled systems:

  Nature of the Challenge. While some threats to resiliency might be managed during development, additional challenges arise at the deployment stage. Machine learning models might not work as expected when deployed, which can deteriorate the *robustness*, *safety*, and *reliability* of the system.

  Possible Underlying Causes. The primary issue is the difficulty in understanding how the model will perform in real-world scenarios. Factors such as diverse environmental conditions, varied input data, and unforeseen operational challenges can impact the model's performance post-deployment. This challenge has been noted in specific contexts, such as audio applications, but it can affect any kind of ML solution.

Potential Impacts: Unexpected performance issues post-deployment can lead to failures in critical applications, potentially causing significant harm or financial loss. It can also reduce user trust and system reliability, making it crucial to address these challenges to ensure the system functions correctly in real-world conditions.

Target Research: Research has identified post-deployment simulations as a complementary instrument to traditional software testing [292]. These simulations involve the definition of agent-based models that can (i) simulate the environment where the ML-enabled system would operate and (ii) verify the system against a wide range of simulated inputs. Such simulations can help in understanding and predicting the model's behavior in real-case scenarios, thus enhancing its *robustness*, *safety*, and *reliability*. Researchers in the fields of *verification and validation* and *machine learning for software engineering* should focus on developing and refining these simulation techniques to address post-deployment challenges effectively. By doing so, they can contribute to creating more resilient and reliable ML-enabled systems that perform consistently in real-world applications.

- **Challenge R.6 - Preserving Privacy in Machine Learning-Enabled Systems [13, 310].** Non-functional requirements related to *security* and *privacy* for ML-enabled systems are impacted by this challenge:

  Nature of the Challenge. The challenge concerns the development of privacy-preserving deep neural networks and machine learning systems. This involves creating algorithms that can operate securely on distributed data without compromising sensitive information about the data subjects, while allowing users to delete their data at any time.

  Possible Underlying Causes. Ensuring privacy in ML-enabled systems requires strategies that protect private data during training

and inference processes. The challenge lies in developing methods that allow models to learn from data without directly accessing or storing sensitive information, which includes data encryption, anonymization, federated learning, and differential privacy techniques.

Potential Impacts: Failing to preserve privacy can lead to data breaches and unauthorized access to sensitive information, undermining user trust and violating data protection regulations. Privacy-preserving methods are crucial for maintaining the *privacy*, *security*, and protection of personal data while ensuring the *accuracy* of the models.

Target Research: This challenge targets multiple software engineering fields, from *software quality* to *software architecture*. Researchers need to focus on deepening the current knowledge of data encryption, anonymization techniques, federated learning, and differential privacy. These strategies are essential for training models without sacrificing *accuracy* while ensuring the *privacy*, *security*, and protection of private data. By advancing these techniques, the software engineering community can contribute to developing robust privacy-preserving ML-enabled systems that maintain user trust and comply with data protection standards.

SUSTAINABILITY (S): As for the challenges in this category, I could observe that most of the available literature focused on ethics and fairness, somehow neglecting other perspectives of sustainability. More particularly, I identified six main challenges connected to algorithmic discrimination, model accountability, fairness metrics, low-quality datasets, energy cost, energy and performance aware trade-offs, and generalizability of existing solutions.

- **Challenge S.1 - Dealing with Algorithmic Discrimination [27, 50, 63, 140, 331, 370, 388].** This challenge affects *"Accuracy"* and *"Sustainability"* and in particular *accuracy* and *fairness* of ML-enabled systems:

Nature of the Challenge. Improving *performance*, *accuracy*, and *fairness* simultaneously remains a significant challenge for researchers. Existing methods for generating discriminatory instances often fail to produce realistic samples. The primary concern is studying and mitigating the impact of disparate results, offensive labeling, and uneven algorithmic error rates in data-driven applications.

Possible Underlying Causes. The root causes of algorithmic discrimination include biases present in training data, model design, and deployment contexts. Discrimination can occur during the pre-processing phase, where data is prepared for model training, leading to biased outcomes that favor certain groups over others.

Potential Impacts: Algorithmic discrimination can lead to unfair and biased outcomes, reducing the trustworthiness and ethical integrity of ML systems. This can have severe societal impacts, particularly in applications like hiring, lending, and law enforcement, where biased algorithms can perpetuate and amplify existing inequalities.

Target Research: The primary studies highlighted the importance of addressing algorithmic discrimination, especially during the pre-processing phase, to improve the trade-off between *fairness* and *performance* in ML software. The software engineering research community has made some strides in this area (e.g., [5, 338]), but further research is necessary. Focus areas include *software analytics* and *verification and validation*, which can provide tools and methodologies to detect and mitigate biases, ensuring more equitable and fair ML-enabled systems. By advancing these research areas, the community can develop robust methods to address algorithmic discrimination effectively, contributing to the ethical deployment of ML technologies.

- **Challenge S.2 - Model Accountability [37, 338, 357].** This challenge mostly affects *"Accuracy"* and *"Sustainability"*, particularly

touching non-functional requirements concerning *accuracy*, *fairness*, *accountability*, and *ethics* of ML-enabled systems:

Nature of the Challenge. The term *'accountability'* refers to the model's ability to provide clear explanations of its decisions, its transparency in how it has been trained and makes decisions, and its ability to allow users to provide feedback or challenge its decisions. Ensuring model accountability involves making the decision-making process understandable and transparent, which is key for ethics and fairness.

Possible Underlying Causes. The underlying causes of the lack of model accountability include complex and opaque model architectures, insufficient documentation of training processes, and the absence of mechanisms for users to query or challenge decisions. These issues are compounded by the rapid development and deployment cycles of ML systems, which often prioritize performance over transparency.

Potential Impacts: Without accountability, ML-enabled systems can make decisions that are difficult to understand or explain, leading to mistrust and potential misuse. This can have significant ethical implications, especially in critical areas such as healthcare, criminal justice, and finance, where decisions can significantly impact individuals' lives.

Target Research: The primary studies considered in this work emphasized the importance of focusing more effort on model accountability to ensure the ethical and fair deployment of ML systems. The research should prioritize the development of *verification and validation* instruments that allow practitioners to verify how accountable their systems actually are. For instance, decision-making frameworks can serve as ideal tools to assess the level of *accountability* of machine learning models. By focusing on these verification and validation processes, researchers and practitioners can ensure that models are not only accurate but also transparent

and responsive to user feedback, thus fostering trust and ethical integrity in ML-enabled systems.

- **Challenge S.3 - Fairness Analytics [58, 64, 184, 273, 310, 317, 326, 338, 386, 391, 398].** This challenge impacts on both *"Accuracy"* and *"Sustainability"*, and in particular non-functional requirements concerning *accuracy*, *fairness*, and *accountability* of ML-enabled systems:

  Nature of the Challenge. Fairness analytics involves defining metric toolkits that support practitioners in diagnosing fairness during the development of ML-enabled systems. The challenge lies in creating tools that help identify and measure fairness-related issues early in the development process. However, fairness should not be considered a non-functional requirement in isolation but rather as an integral part of trade-off analyses.

  Possible Underlying Causes. The primary cause of this challenge is the inherent complexity of balancing fairness with other competing requirements such as accuracy, privacy, and accountability. Existing tools and metrics often lack the capability to provide a comprehensive analysis that considers these trade-offs simultaneously. The dynamic and context-dependent nature of fairness further complicates the development of universal metrics and toolkits.

  Potential Impacts: If fairness is not adequately addressed, ML-enabled systems can perpetuate biases and inequalities, leading to unfair and discriminatory outcomes. This can erode trust in the system, reduce its acceptance and reliability, and cause harm to individuals and communities.

  Target Research: The literature calls for novel metrics that provide insights into the trade-offs between *fairness* and other critical non-functional requirements such as *accuracy*, *privacy*, and *accountability*, as well as combinations of these factors. The research

highlights the need for diagnosis and multi-objective optimization of data-driven applications across various stages of development, from requirements engineering to verification and validation. To address these challenges, researchers should focus on developing comprehensive metric toolkits that can diagnose fairness issues and provide multi-objective optimization solutions. These toolkits should be integrated into the development workflow to ensure that fairness is considered alongside other non-functional requirements from the outset. By doing so, practitioners can make informed decisions that balance fairness with other critical system attributes, leading to more equitable and trustworthy ML-enabled systems.

- **Challenge S.4 - Improving Sustainability Benchmarks [37, 378].** Such a challenge may impact various aspects, including *accuracy*, *fairness*, and *accountability* of ML-enabled systems:

  Nature of the Challenge. Improving sustainability benchmarks involves creating high-quality datasets that assist in developing sustainable ML applications. The challenge is to develop datasets that enable comprehensive cross-domain and cross-sectional analysis while maintaining high standards of fairness and accountability.

  Possible Underlying Causes. The primary cause of this challenge is the prevalence of low-quality datasets that can exacerbate issues related to discrimination, fairness, and accountability. Low-quality datasets often fail to represent diverse populations adequately, leading to biased training and evaluation of machine learning models. Additionally, the need for specialized datasets in certain application domains, such as image processing, further complicates the creation of comprehensive benchmarks.

  Potential Impacts: Low-quality benchmark datasets can lead to the development of ML systems that are biased and unreliable. This can negatively impact the fairness and accountability of these systems, leading to discriminatory outcomes and reducing trust

in the technology. Furthermore, high-quality benchmark datasets may be essential to ensure that ML models are trained and evaluated in a way that is fair, accurate, and representative of diverse populations.

Target Research: Current research has highlighted the need for improved benchmark datasets to create sustainable ML applications. Low-quality datasets are more likely to discriminate against individuals in the dataset and worsen fairness and accountability attributes. Therefore, there is a pressing need for high-quality datasets that enable automated pose, illumination, and expression (PIE) analysis in image processing applications. Additionally, datasets that present large and diverse information on minorities are crucial for more effective training of ML systems. Researchers should focus on developing these high-quality datasets by ensuring they cover diverse populations and various domains comprehensively. This involves collaborative efforts across different fields and leveraging advanced data collection and annotation techniques to create benchmarks that truly reflect the complexities and nuances of real-world data. By improving the quality and diversity of benchmark datasets, the research community can enhance the sustainability, fairness, and accountability of ML-enabled systems, ultimately leading to more robust and trustworthy applications.

- **Challenge S.5 - Reducing Energy Cost [59, 114, 149, 150, 216, 220, 224, 315, 340, 342, 362].** As the name of the challenge suggests, it impacts both *energy consumption* and *cost* of ML-enabled systems:

  Nature of the Challenge. Reducing the energy and computational costs of deep neural networks is a critical challenge. These systems often require substantial energy and financial resources, leading to increased CO2 emissions and memory consumption during both the training and post-release phases.

  Possible Underlying Causes. Deep neural networks are inherently resource-intensive due to their complex architectures and large

training datasets. The significant computational power needed for training these models results in high energy consumption. Additionally, post-deployment phases, especially on edge and IoT devices, exacerbate energy consumption issues due to the need for continuous processing and inference.

Potential Impacts: High energy and computational costs can limit the scalability and deployment of ML-enabled systems, particularly in edge and IoT environments where resources are constrained. This not only increases operational costs but also has a significant environmental impact due to higher CO2 emissions. Moreover, it can lead to increased memory consumption, affecting their efficiency and sustainability.

Target Research: This challenge may involve researchers in the board fields of *empirical software engineering*, *software quality and architecture*, and *machine learning engineering*. The former may be involved in assessing the impact of different AI containerization strategies on energy consumption and memory utilization. By empirically evaluating these strategies, it is possible to identify methods that minimize energy usage while maintaining performance. Researchers in software quality and architecture may focus on balancing the trade-offs between training accuracy and post-deployment energy consumption. This includes optimizing model architectures and developing energy-efficient training techniques. Finally, machine learning engineering researchers may be specifically looking at the rising field of tiny machine learning, which combines hardware, algorithms, and software to support on-device sensor data analytics at low power. This field aims to create efficient ML models that can operate on resource-constrained devices without compromising performance.

- **Challenge S.6 - Increasing the Practitioner's Awareness of Sustainability [114, 149, 303].** This challenge affects non-functional

requirements concerning *energy consumption* and *cost* of ML-enabled systems:

Nature of the Challenge. Increasing practitioners' awareness regarding sustainability concerns is a critical challenge identified in this literature review. This issue is especially relevant to energy consumption, which is a significant concern in today's environmentally conscious world. The challenge lies in making developers aware of the trade-offs between performance and sustainability.

Possible Underlying Causes. Many practitioners may prioritize performance and accuracy over sustainability due to a lack of awareness or tools that highlight the environmental impact of their decisions. Additionally, existing development practices and tools may not emphasize the importance of sustainability, leading to its neglect during the software development lifecycle.

Potential Impacts: If practitioners remain unaware of sustainability concerns, ML-enabled systems may continue to be developed with high energy consumption and resource usage, negatively impacting the environment. This can lead to inefficiencies and higher operational costs, reducing the overall sustainability of ML projects.

Target Research: Addressing this challenge would require the effort of multiple communities. *Empirical software engineering* researchers may work toward understanding practitioners' needs and practices. By tailoring automated approaches to fit these needs, researchers may develop tools and methods that highlight the sustainability impacts of various development decisions. This could include developing frameworks that provide feedback on energy consumption and resource usage during the development process. At the same time, *software quality*, *software analytics*, and *machine learning engineering* researchers may target the design of automated approaches able to improve the scalability of models without compromising their overall accuracy. In this respect,

approaches such as quantization [194] and knowledge distillation [307] can help reduce the energy footprint of ML models, making them more sustainable. Also, researchers in *software engineering education* may play a role in helping to create a common understanding of sustainability practices in software engineering.

🔍 USABILITY (U): Usability is the non-functional requirement with fewer challenges for ML-enabled systems. Only a few works targeted this matter, suggesting that further research might be worthwhile. In this context, non-functional requirements such as *explainability* and *imperceptibility*, along with the need for visualizing the outcomes of both shallow and deep learning models, pose significant challenges when it comes to assessing quality assurance throughout the software lifecycle. For instance, the lack of *interpretability* increases the effort required to estimate the soundness of ML-enabled systems. Current research neglected this research angle, especially when considering models operating in critical and evolving scenarios. As such, this findings suggest the urgent need for further research on usability concerns [8, 69, 92, 94, 142, 151, 170, 173, 220, 261, 404]. Ensuring usability of AI algorithms and their decisions can be a challenge for users and regulators, bringing distrust of users and stakeholders [77, 261, 277, 293]. This further suggests the investigation of domain-specific approaches and methods.

🔍 CROSS-CUTTING CHALLENGES (C): Besides the challenges for ML-enabled systems specifically relating to individual classes of non-functional requirements, I also identified a set of cross-cutting challenges associated with the management and assessment of these requirements. It is worth remarking that the list of associated NFRs for each cross-cutting challenge reflects the specific NFRs that were most frequently and explicitly linked to the challenge in the primary studies included in this systematic literature review. This means that the associations presented are not exhaustive and do not exclude the possibility that a given challenge may also apply to other NFRs. Instead, the associ-

ations highlight the NFRs that were directly discussed in the context of each challenge within the reviewed papers.

- **Challenge C.1 - Dealing with Internal Errors [101, 110, 149, 169, 215, 220].** This challenge mostly affects *"Efficiency"*, *"Sustainability"*, and *"Resiliency"*, and in particular non-functional requirements concerning *robustness*, *accuracy*, *performance*, *energy consumption* and *cost* of ML-enabled systems:

  Nature of the Challenge. Machine learning models and, in particular, neural networks can produce erroneous outputs due to internal errors caused by *incorrect parameters*, *incorrect weight values*, *uncovered root causes*, and *incorrect manual labeling*.

  Possible Underlying Causes. Internal errors can stem from several factors, such as misconfigured parameters, incorrect weight values assigned during the training process, overlooked root causes in the data or model structure, and errors introduced during manual labeling of training data.

  Potential Impacts: These internal errors may significantly impact the performance of ML-enabled systems, leading to reduced accuracy, slower inference times, and increased energy consumption. For example, incorrect weight values or parameters can cause the network to converge slowly or get stuck in suboptimal solutions, resulting in poor performance.

  Target Research: Dealing with internal errors has two main aspects: tracking errors in deep neural network models and addressing them. As a consequence, researchers in the fields of *bug localization*, *software analytics*, and *program repair* may help address the elicited challenges. For instance, *bug localization* researchers may work toward the development and refinement of novel techniques to effectively identify and localize bugs within neural network models by leveraging techniques such as fault injection, automated debugging, or static analysis. Research in *software analytics* may

revolve around the definition of advanced monitoring frameworks that track a wide range of performance metrics, including accuracy, latency, throughput, and resource utilization. Finally, *program repair* researchers may provide automated instruments that suggest and implement fixes by leveraging AI algorithms.

- **Challenge C.2 - Diagnosing the Internal Behavior of Models [5, 95, 97, 153, 172, 229, 333, 371, 384, 389, 397].** This challenge affects multiple clusters, including *"Accuracy"*, *"Resiliency"*, and *"Usability"*. It particularly impacts *accuracy*, *robustness*, *safety*, and *transparency* of ML-enabled systems:

  Nature of the Challenge. Understanding the root cause of anomalous behaviors of deep neural networks and how to fix them represents a critical aspect to further elaborate.

  Possible Underlying Causes. Evaluating the *reliability* implications of small sets of weights assigned to the network can support practitioners in better understanding how they work. Inconsistent weight assignments, faulty training data, or model architecture flaws can lead to unpredictable or erroneous model behaviors.

  Potential Impacts. Failure to diagnose and address these issues can lead to significant misclassifications, security concerns, and overall reduced model *robustness* and *accuracy*. Anomalous behaviors can undermine the trustworthiness and reliability of the ML-enabled system, leading to potential failures in real-world applications.

  Target Research. Addressing this challenge involves the development of methods for exploration, *interpretability*, and *explainability* to diagnose failures and misclassifications. Researchers in *fault localization* can contribute by identifying precise locations of faults within the model. *Explainable AI* researchers can focus on creating methods that make the decision-making process of models transparent, aiding in the identification of the root causes of failures. *Software metrics* researchers can develop novel metrics that go beyond *accuracy* to provide actionable insights into

the model's *robustness* and reliability. Finally, *automatic program repair* researchers can design tools to automatically fix identified issues within the models, ensuring their proper functioning and enhancing overall system reliability.

- **Challenge C.3 - Context-based Trade-off Identification [43, 55, 63, 112, 310].** According to this analysis, this challenge affects *"Accuracy"*, *"Resiliency"*, *"Sustainability"*, *"Maintainability"*, and *"Efficiency"*, and in particular *accuracy*, *replaceability*, *security*, *fairness*, and *performance* of ML-enabled systems; however, it is important to note that the challenge may also extend to other NFRs beyond those for which I observed explicit evidence in the reviewed studies:

  Nature of the Challenge. Identifying a trade-off between multiple non-functional requirements, particularly those influenced by specific contextual factors (e.g., regulatory requirements, ethical considerations, and operational constraints), represents a key socio-technical challenge. This step involves exploring and diagnosing which NFRs are most relevant for the deployment environment.

  Possible Underlying Causes. ML-enabled systems must always preserve *accuracy*, but other non-functional requirements such as *robustness*, *security*, or *fairness* may be more or less relevant based on the specific context where the system is deployed. The underlying causes include the socio-technical influences of the deployment environment, such as domain-specific regulations, user expectations, and ethical considerations, which dictate the importance of different NFRs and their trade-offs.

  Potential Impacts. The failure to appropriately identify non-functional requirements based on contextual needs can lead to suboptimal system performance and may even result in critical failures. For instance, a driving system that does not identify and understand specific non-functional requirements such as *robustness* and *security* might be prone to failures or attacks, while a

healthcare system that does not emphasize *fairness* might result in biased treatment outcomes. This challenge is particularly relevant for system analysts, domain experts, and requirements engineers who need to explore and document these trade-offs as an early step in system design.

Target Research. This challenge emphasizes the need for novel approaches to identify non-functional requirements in specific contexts. Researchers in *software analytics* can develop methods to (semi-)automatically mine contextual information, providing practitioners with insights into the non-functional requirements that should be identified and highlighted. Additionally, research in *software project management* can focus on novel methods to assess the significance of non-functional requirements, supporting managerial decisions in the design and deployment of ML-enabled systems. By integrating socio-technical analyses and advanced project management techniques, researchers can contribute to more effective trade-off identification and decision-making processes.

- **Challenge C.4 - Prioritizing and Balancing Non-Functional Requirements [58, 63, 94, 114, 149, 179, 184, 326, 342, 388, 391, 403].** This challenge affects *"Accuracy"*, *"Resiliency"*, *"Sustainability"*, *"Maintainability"*, and *"Efficiency"*, and in particular non-functional requirements concerning *accuracy*, *robustness*, *replaceability*, *cost*, and *performance* of ML-enabled systems. Similarly to **C.3**, it is likely that this challenge also impacts other NFRs beyond those explicitly discussed in the reviewed studies. The NFRs listed here reflect those most frequently and explicitly linked to this challenge in the primary studies, and they represent areas where balancing priorities has been particularly emphasized. However, given the broad and interconnected nature of this challenge, it is plausible that its relevance extends to additional NFRs that were not directly observed in this analysis:

Nature of the Challenge. Balancing multiple non-functional requirements represents a key challenge in developing ML-enabled systems. While **C.3** focuses on identifying relevant trade-offs based on context, **C.4** addresses the prioritization and resolution of conflicts between competing NFRs to achieve a balanced system design. This step involves strategic decision-making and technical resolution.

Possible Underlying Causes. The complexity of balancing non-functional requirements arises from the need to address multiple, often conflicting, objectives simultaneously. Different requirements may have varying levels of importance depending on the specific use case and context, and prioritizing them effectively requires a deep understanding of their interdependencies and impacts. Unlike the exploratory nature of **C.3**, **C.4** involves decision-makers such as software architects and project managers, who must consider trade-offs identified earlier and resolve them to achieve a cohesive system design.

Potential Impacts. Inadequate prioritization and balancing of non-functional requirements can lead to suboptimal system performance, increased costs, and potential failures. For example, overemphasizing *performance* at the expense of *security* may result in vulnerabilities while focusing too much on *fairness* might reduce *efficiency*. These imbalances can compromise the overall quality and reliability of ML-enabled systems.

Target Research. The *requirements engineering* research community can address this challenge by proposing guidelines or automated tools to prioritize and balance non-functional requirements. Researchers can develop methods to analyze contextual factors and practitioners' preferences, enabling them to find an optimal balance among correlated and contrasting objectives. This may involve leveraging multi-objective optimization techniques,

decision-making frameworks, and tools that assist stakeholders in navigating complex trade-offs and prioritization challenges.

- **Challenge C.5 - Integration and Interaction of ML Models [8, 111, 272, 321, 363].** This challenge affects *"Accuracy"*, *"Resiliency"*, *"Sustainability"*, *"Maintainability"*, and *"Efficiency"*. Specifically, *accuracy*, *security*, *adaptability*, *cost*, and *performance* of ML-enabled systems:

  Nature of the Challenge. Integrating ML models within complex systems involves synchronizing multiple components, such as traffic light recognition, lane detection, and obstacle perception. Each of these components may utilize several ML models that must function cohesively, hence further complicating the integration.

  Possible Underlying Causes. The dynamic interactions among these ML models create dependencies that are difficult to predict and manage. While individual ML models can undergo rigorous testing, comprehensive testing for the integrated system as a whole is often inadequate. In addition, designing complex architectures requires knowledge of the characteristics of ML models and, perhaps more importantly, of the whole set of non-functional requirements expected to be implemented by the ML-enabled system.

  Potential Impacts. Inadequate integration can lead to unreliability, where the system might fail to perform as expected under certain conditions. This is particularly critical in safety-critical systems, such as autonomous driving, where integration failures can result in dangerous situations and potentially fatal accidents.

  Target Research. Researchers in the fields of *software architecture* and *software testing* can address this challenge by developing methods to ensure seamless integration and interaction of multiple ML models within complex systems. This may involve creating comprehensive testing frameworks that simulate real-world interactions and dependencies among ML models, ensuring that

integrated systems are robust and reliable. Additionally, developing synchronization protocols and automated tools for managing dependencies among ML models can help improve the overall reliability of integrated systems. Finally, software architecture techniques should be made model-aware and non-functional-requirements-aware, meaning that they should incorporate pieces of information that may support software architects in the design of complex infrastructures.

- **Challenge C.6 - Improving Practitioners' Knowledge [77, 145, 273, 315].** This challenge affects the whole set of non-functional requirements of ML-enabled systems:

  Nature of the Challenge. Understanding how to handle non-functional requirements for ML systems compared to traditional software systems can be problematic. ML systems introduce new non-functional requirements and alter the importance and interpretation of existing ones, as pointed out in $\mathbf{RQ}_1$. Practitioners often struggle to adapt existing knowledge and processes to effectively manage these evolving requirements.

  Possible Underlying Causes. Non-functional requirements for ML systems are highly dependent on the application domain and require specific skills for accurate definition and measurement. Additionally, there is a significant lack of awareness and understanding of non-functional requirements among both customers and engineers, leading to insufficient consideration and implementation. Measuring non-functional requirements is also complex, leading to their omission in analyses.

  Potential Impacts. These challenges can negatively impact ML-enabled systems, compromising quality and customer satisfaction, and can lead to regulatory non-compliance. For instance, in autonomous vehicles, non-functional requirements related to safety and robustness are critical to preventing accidents. In health diagnostics, accuracy and interpretability are essential for proper

diagnosis and treatment, while in financial systems, fairness, transparency, and security are vital to avoid biased decisions and legal issues.

Target Research. Researchers in the fields of *software engineering education*, *requirements engineering*, and *empirical software engineering* can address this challenge by developing training programs and educational materials that enhance practitioners' understanding of NFRs specific to ML systems. Creating guidelines and best practices for the accurate definition, measurement, and management of non-functional requirements in various application domains can also help. Additionally, tools and frameworks that assist in the identification and evaluation of non-functional requirements during the software development lifecycle can support practitioners in ensuring comprehensive consideration of these requirements.

- **Challenge C.7 - Software Analytics for Non-Functional Requirement Assessment [59, 150, 151, 259, 376].** This challenge affects the whole set of non-functional requirements of ML-enabled systems:

  Nature of the Challenge. Multiple primary studies advocate the need to empower requirements engineering processes with software analytics instruments able to assess the implications that trade-off choices may have on the development of ML-enabled systems.

  Possible Underlying Causes. Current literature highlights a gap in software metrics that can inform practitioners about how their requirements engineering decisions may impact both the complexity of the development process and the overall quality of the system. There is a lack of strategies to recommend quality assurance mechanisms based on specific trade-offs and predictive analytics tools that can enhance practitioners' capabilities in assessing the impact of trade-off analysis.

Potential Impacts. Without effective software analytics, practitioners may struggle to make informed decisions regarding non-functional requirements, leading to suboptimal trade-offs that could compromise system quality and development efficiency. For example, lacking tools to predict how certain trade-offs will affect the system may result in overlooked quality assurance mechanisms, increasing the risk of defects and reducing system reliability and performance.

Target Research. Researchers in *requirements engineering* and *software analytics* can address this challenge by developing novel software metrics and analytics tools that help practitioners assess the impact of their decisions regarding non-functional requirements. Collaborative efforts could focus on creating predictive analytics instruments that enhance the ability to foresee the consequences of trade-offs throughout the software lifecycle. By doing so, the research community can improve the support available to requirements engineers, enabling them to make more informed and effective decisions.

- **Challenge C.8 - Improving the Generalizability of Existing Automated Approaches [51, 112, 163, 335].** This challenge impacts all the non-functional requirements of ML-enabled systems:

  Nature of the Challenge. The challenge concerns the generalizability of existing approaches that support the sustainability analysis of ML-enabled systems. Current methods often have limitations that hinder their broader application. Limited generalizability often leads to the development of narrowly applicable tools, which increases effort, reduces the ability to reuse components, and hinders the adoption of certain practices in both the short and long term.

  Possible Underlying Causes. Multiple primary studies identified limitations in terms of learning tasks and models support. For instance, recommendation systems like PLUM are restricted to

deep learning models, while other approaches can only handle a limited set of deep neural networks, are built using limited datasets [51], or are not integrated within MLOps pipelines [163].

Potential Impacts. These limitations restrict the usefulness of current tools in practical scenarios, making it difficult for practitioners to generalize their findings across different types of ML models and learning tasks. Consequently, this can lead to inefficiencies and increased effort in model repair, development, and maintenance, ultimately compromising the sustainability and performance of ML-enabled systems.

Target Research. Researchers in the fields *software analytics*, *machine learning engineering*, and *MLOps* are called to the development of more generalizable, accurate, and integrated tools that support practitioners throughout the software lifecycle. This includes creating instruments that can handle a wide range of neural network architectures, utilize diverse and comprehensive datasets, and seamlessly integrate with MLOps pipelines. Such advancements will enhance the applicability and effectiveness of sustainability analysis tools in real-world settings, thereby improving the overall quality and efficiency of ML-enabled systems.

As a final note, it is worth remarking that I observed that some challenges were discussed by more papers, while others have a lower frequency of mention across the literature. In this respect, I cannot speculate on whether this is because some challenges are inherently more important than others. The number of referenced publications discussing a particular challenge may reflect several factors, such as current research trends, the ease of identifying and addressing certain challenges, or even the availability of data and case studies related to specific issues. In other terms, while a higher number of references might suggest that a challenge is widely recognized or frequently encountered, it does not necessarily indicate its overall importance relative to other challenges. More comprehensive evaluations, considering both quantitative and qualitative aspects, may be

necessary to understand the full scope and significance of the challenges associated with non-functional requirements in ML-enabled systems.

> **RQ_2 — NFRs Challenges**
>
> *I elicited 26 software engineering challenges targeting the individual categories of non-functional requirements identified in this systematic synthesis work and the cross-cutting aspects affecting non-functional requirements engineering. I methodically untangled the challenges faced by ML-enabled systems, providing insights into the specific research fields interested in those issues. According to this results, I call for comprehensive analyses and approaches that may assess the impact and implications of managing individual and multiple non-functional requirements.*

2.4 DISCUSSION AND IMPLICATIONS

The results of this study provide a number of additional discussion points and implications, which I discuss further in the following.

IMPACT OF THIS FINDINGS ON RESEARCH AND PRACTICE. These results allowed us to synthesize the currently available knowledge on non-functional requirements of ML-enabled systems, hence contributing to a unified framework that consolidates the state of the art in the field [7, 144, 146, 351]. By leveraging the set of primary studies collected through this hybrid systematic literature review, I could aggregate the findings obtained in the broad field of requirements engineering, hence reporting a comprehensive overview of the multifaceted aspects impacting the trustworthiness of ML-enabled systems. I could specifically classify 31 distinct non-functional requirements, categorized into six main classes. More importantly, with $\mathbf{RQ}_1$, I could provide insights into the meaning of the collected non-functional requirements, finding commonalities and peculiarities with respect to non-functional requirements considered in non-ML-enabled systems. For instance, I observed that certain

requirements like performance efficiency and traceability have similar interpretations in both ML and non-ML contexts. At the same time, unique non-functional requirements, e.g., retrainability, replaceability, or stability, emerged as aspects that characterize ML-enabled systems and that, therefore, require focused investigations by the research community. Additionally, this classification revealed that a number of non-functional requirements, e.g., scalability and security, present specific nuances within ML-enabled environments, highlighting the need for tailored approaches and methodologies to effectively address these requirements. Through this detailed classification and analysis, this work not only enhances the understanding of non-functional requirements in ML-enabled systems but also provides a reference framework for research and practice. For instance, I envision studies that delve deeper into the unique non-functional requirements of ML-enabled systems in an attempt to characterize their impact and criticality in various application domains. In this respect, the results of $\mathbf{RQ}_1$ inform the research community on the application domains targeted by researchers so far, possibly indicating areas that might be worth investigating. Furthermore, I may also envision studies aiming at refining current requirements elicitation and analysis practices to take the peculiarities of ML-enabled systems into account, e.g., by developing guidelines for eliciting sustainability requirements. Last but not least, this classification may serve as a starting point for researchers to devise novel methodologies for assessing, validating, and monitoring non-functional requirements of ML-enabled systems, e.g., by proposing methods to assess retrainability, replaceability, and stability throughout the lifecycle of ML-enabled systems. From the practitioners' perspective, this classification provides a unified body of knowledge to gather information about the qualities to meet when developing trustworthy ML-enabled systems, raising their awareness on the complex set of non-functional requirements that may impact the development.

Implication #1. This classification of non-functional requirements provides a reference framework that comprehensively integrates the current knowledge on non-functional requirements of ML-enabled systems, presenting commonalities with research on non-functional requirements of non-ML-enabled systems and peculiarities that make ML-enabled systems different. I argue that the results to $\mathbf{RQ}_1$ may inspire further research, other than making practitioners aware of the multiple, multi-faceted concerns arising when developing ML-enabled software systems.

When it turns to $\mathbf{RQ}_2$, this work aggregates a set of 26 software engineering challenges proposed in the literature (e.g., [146, 161]), hence presenting a *super-set* of the challenges documented in earlier research. Through the analysis of the primary studies collected in the hybrid systematic literature review, I could classify the unique challenges specific to the individual classes of non-functional requirements identified in $\mathbf{RQ}_1$, but also the cross-cutting challenges that affect the whole set of non-functional requirements of ML-enabled systems. From a scientific perspective, this work addresses one of the major challenges identified by Villamizar et al. [351], which was actually related to the lack of systematic knowledge of non-functional requirements of ML-enabled systems. In addition, this work classifies the documented relations between multiple non-functional requirements, which may be seen as an additional contribution that may stimulate further research on the matter. From a tangible perspective, this work has critical implications for multiple research fields. In response to $\mathbf{RQ}_2$, I could identify the target audience of each software engineering challenge, providing insights that are not only relevant for requirements engineering research but that extend to other sub-communities like software analytics, software testing, and empirical software engineering. This synthesis effort may pave the way for interactions and joint collaborations toward addressing the many specific challenges of non-functional requirements of ML-enabled systems that encompass analysis, measurement, and monitoring from requirements elicitation until testing and post-deployment operations. Furthermore,

the challenges reported by this work may be of interest to professional ML engineers and project managers, who may have an improved understanding of how these challenges may impact the development of real-world systems, other than of the possible obstacles to technological transfer they should preliminary address.

> **Implication #2.** This work could identify several software engineering challenges that affect the management of non-functional requirements of ML-enabled systems. I argue that the challenges discussed in **RQ**$_2$ inform future research efforts and possibly stimulate interactions and collaborations among software engineering researchers studying solutions to support practitioners throughout the development lifecycle. In addition, the challenges reported in this work may inform practitioners of the obstacles to the development of trustworthy ML-enabled systems, leading them to more careful considerations of non-functional requirements in real-world contexts.

In addition to the points discussed above, these findings from **RQ**$_1$ have the potential to significantly influence practical applications by helping to refine standards and guidelines for handling non-functional requirements of machine learning systems. For instance, the detailed definitions could enhance emerging standards such as the *ISO/IEC 25059*,[5] which already outlines a taxonomy of non-functional properties. This could lead to more comprehensive guidelines that better support researchers and practitioners in the field. On the one hand, these standards cover various system and software quality attributes, including performance efficiency, reliability, usability, security, maintainability, and portability. The taxonomy builds on these aspects by introducing ML-specific non-functional requirements such as scalability, retrainability, interpretability, and explainability, which are critical because of the ML-enabled systems' dynamic and data-intensive nature. On the other hand, Habibullah et al. [144] focused mainly on ethics and biases within non-functional requirements for ML systems. The study expands this perspective to include

[5]The *ISO/IEC 25059* standard: https://iso25000.com/index.php/en/iso-25000-standards/iso-25059

other attributes such as energy consumption, sustainability, accountability, and robustness. This comprehensive approach ensures coverage of all critical quality aspects of ML systems, providing a more complete framework for assessing and managing non-functional requirements in these systems. In other terms, the consolidated classification was therefore needed to synthesize the currently emerging knowledge on the specific characteristics and requirements of ML-enabled systems - aspects that are not fully covered by existing standards and classifications. Furthermore, the results complement the insights from Amershi et al.[12], which identified best practices for the actionable development of machine learning systems, including aspects like model debugging and interpretability. This work provides concrete factors that directly relate to these practices, such as the *retrainability* attribute within the *'Maintainability'* category, offering a foundation for developing measurable and monitorable systems through the various stages of machine learning system development. In addition, the findings of $\mathbf{RQ}_2$ provide insights into the research gaps, making the best practices identified by Amershi et al. [12] difficult to actually implement. Also in this case, the systematic synthesis may provide researchers with an improved understanding of the next steps to pursue better support for the development and evolution of ML-enabled systems. Ali et al. [11] emphasized that quality standards should follow the 'ility' procedure based on traditional quality models and require comprehensive quality models for artificial intelligence software components. In this respect, this work could greatly contribute to the creation of new quality standards that would be more robust and tailored to the complexities of AI software, improving its reliability and effectiveness.

Implication #3. these findings complement the current state of the practice in different manners. First, the outcome of $\mathbf{RQ}_1$ may be useful to extend and/or complement emerging standards describing relevant non-functional requirements of machine learning-enabled systems. Second, the results of $\mathbf{RQ}_1$ and $\mathbf{RQ}_2$ can be combined with the pieces of information that emerged from the state of the practice, highlighting the current research gaps that should be filled and the potential opportunities of technological transfer.

ON THE INTER-RELATION AMONG NON-FUNCTIONAL REQUIREMENTS.

Based on the cross-cutting challenges discussed in the context of $\mathbf{RQ}_2$, I could identify the presence of some inter-relations among non-functional requirements, i.e., how the satisfaction or optimization of one non-functional requirement can influence or be influenced by another. These are not only concerned with the relation that accuracy has with other requirements but also with the innate interconnections between multiple non-functional aspects playing a role in developing ML-enabled systems [267, 326, 362]. These results demonstrated that groups within the same cluster are closely related, making it easier to improve others within the same cluster when one is enhanced [37, 44, 69]. Conversely, improving one cluster can lead to the deterioration of non-functional requirements in different clusters [111, 391, 403]. In this respect, researchers in requirements engineering and empirical software engineering might cooperate toward developing novel taxonomies that may map the relations among the non-functional requirements of ML-enabled systems and how they impact each other. these results also called for further research on identifying, managing, and assessing the trade-offs among multiple non-functional requirements.

> **Implication #4.** these findings suggest novel interconnections and research on the multi-objective optimization of non-functional requirements of ML-enabled systems. I call for a new research field focusing on empirically investigating the relations between non-functional requirements and how those relations may inform the development of automated approaches to optimize ML-enabled systems.

A MANAGERIAL VIEWPOINT. As a follow-up discussion, these research questions raise multiple trade-offs to consider when developing ML-enabled systems. This work has implications on the managerial side: the constant and continuous need to measure non-functional requirements and search for trade-offs represent high-level challenges for project managers, who are required to monitor and handle multiple non-functional attributes throughout the evolution of ML-enabled systems. In the first place, the results produced by $\mathbf{RQ}_1$ may allow the reader to understand what the set of non-functional attributes actually takes into account. Secondly, the challenges identified in the context of $\mathbf{RQ}_2$ should not only be considered from a technical perspective but also from a socio-technical and managerial one. This work calls for further research on the managerial strategies that would indicate the most appropriate management policies to apply when handling non-functional requirements.

> **Implication #5.** I call for research on managerial and socio-technical strategies to handle non-functional requirements throughout the evolution of ML-enabled systems. At the same time, these findings point out the need for optimization approaches specifically tailored to software evolution, hence being contextual, dependent, and adaptable.

3

SUSTAINABILITY FROM THE PRACTITIONER'S PERSPECTIVE

3.1 INTRODUCTION

Significant challenges persist when deploying ML-enabled systems at scale. Training and inference processes for ML-enabled systems demand large amounts of data and substantial computational resources [304]. Although these requirements do not prevent the adoption of ML-enabled systems, they pose serious threats to the sustainability of the systems [347, 352]. Sustainable development, defined as *"meeting present needs without compromising the ability of future generations to meet theirs"* [35], is a critical concern in the context of ML-enabled systems. Addressing sustainability has become a pressing issue for both practitioners and users of these systems [298, 336].

Software sustainability is a **multifaceted concept encompassing environmental, social, and economic dimensions** [41, 206, 348]. Environmental sustainability focuses on minimizing the carbon footprint and energy consumption of software systems, while economic sustainability ensures that software remains financially viable throughout its lifecycle. Social sustainability, in turn, refers to the extent to which software systems promote societal well-being, fairness, and inclusiveness across their development and use. This is directly tied to sustainability goals because systems that are discriminatory, opaque, or disrespectful of user rights are less likely to be trusted, adopted, or maintained in the long term. As a result, their societal and operational viability, and therefore their sustainability, might be compromised.

Rather than being a static property, sustainability in software engineering emerges from the dynamic interplay of these interconnected dimen-

sions, as formalized by Becker et al. [22] in the Karlskrona Manifesto. This perspective was also remarked by McGuire et al. [242], who defined sustainability as a *stratified* and *multisystemic* property, advocating for deeper analysis into how **sustainability principles evolve over time and how they are actually adopted in practice**.

Within the domain of ML-enabled systems, sustainability has become an increasingly relevant but still underexplored topic. Researchers have approached these challenges from various angles, including systematic investigations of sustainability concerns [330, 350] and the development of novel methods for integrating sustainable software engineering practices into ML workflows [235, 377]. Parallel efforts have fostered community engagement through thematic workshops, e.g., GREENS[1], which focus on advancing discussions around sustainability-related challenges in ML.

Despite this growing academic interest, **sustainability in AI has been predominantly framed through high-level principles and policy recommendations**, rather than through concrete, actionable practices that ML practitioners can directly implement [154, 168]. For instance, guidelines from organizations like IEEE [154] and Accenture [168] frequently emphasize broad principles such as responsible AI, fairness, and energy efficiency but lack detailed practical methods on how to embed sustainability considerations into real-world ML development pipelines. This disconnection between principles and implementation results in sustainability being treated as an afterthought, often measured post hoc rather than integrated proactively during model design and deployment.

Moreover, the majority of sustainability-focused AI research has centered on *environmental aspects*, emphasizing energy-efficient AI, carbon footprint reduction, and green software engineering practices [182, 350], later analyzed in open-source projects [82]. While these efforts provide valuable insights, **they do not account for the broader spectrum of sustainability dimensions**, i.e., social and economic sustainability. This limited focus creates gaps in the understanding of the trade-offs practitioners face when adopting sustainability-oriented approaches in ML systems. As

[1] https://greensworkshop.github.io

highlighted in **Gap #1**, there is still a lack of clear concepts and alignment between researchers and practitioners in defining NFRs and sustainability requirements for ML-enabled systems.

> *To address this gap, this chapter aims to bridge the gap between high-level sustainability principles and the concrete practices employed by ML practitioners. Specifically, I conduct a mixed-method empirical investigation that systematically examines (1) how ML practitioners perceive sustainability, extending beyond theoretical perspectives to uncover industry-grounded interpretations; (2) the sustainability practices that are actively employed, identifying the approaches practitioners use to integrate sustainability considerations into ML-enabled systems; and (3) the challenges that hinder sustainability adoption in ML workflows, providing empirical evidence to inform the development of actionable frameworks, policies, and engineering solutions.*

By leveraging both qualitative insights from interviews with ML professionals and quantitative findings from a large-scale survey, this study provides a **practitioner-centric perspective on sustainability in ML**. I deliver practical recommendations, best practices, and concrete strategies that can support both researchers and practitioners in fostering sustainability across environmental, social, and economic dimensions. In doing so, this paper offers an empirical characterization of sustainability perceptions in ML, sheds light on the sustainability practices that are effectively applied, and identifies the systemic barriers that hinder their adoption.

> **Data availability statement:** To ensure verifiability and replicability, I have made all data and materials from this study publicly accessible [81]. The replication package includes the complete set of interview transcripts, the coding scheme, the survey questionnaire, as well as both raw and processed survey responses and the analysis scripts. This resource validates the results and encourages future researchers to use the material for further study.

3.2 RESEARCH PROCESS OVERVIEW

The study aimed to explore how machine learning (ML) practitioners involved in developing ML-enabled systems approach sustainability aspects. Specifically, the study focused on (1) understanding how ML practitioners perceive and describe sustainability, (2) revealing the sustainability practices they actively use, and (3) identifying the challenges that hinder the integration of sustainability in ML workflows. To achieve the above-mentioned objective, we delineated three main research questions (one for each objective).

First, we aimed to understand how ML practitioners describe and interpret sustainability in ML-enabled systems. Such a preliminary investigation allowed us to challenge the practitioners' understanding of the concept, potentially uncovering alternative perspectives, misconceptions, or gaps in their reasoning. By analyzing their descriptions, we identified areas of consensus and key points of divergence in how sustainability is perceived within the ML community. Additionally, we could compare these perceptions with the definitions of sustainability proposed by researchers to highlight potential alignments and discrepancies.

> **RQ_1—Sustainability Perception**
>
> *How do machine learning practitioners perceive sustainability in ML-enabled systems?*

The second research question sought to understand the practices—including tools and measures—applied by ML practitioners to address ML sustainability. As reported earlier in this chapter, previous major research efforts have primarily focused on identifying green and environmental practices, creating catalogs of green AI approaches [182, 350]. In the context of this work, we aim to extend the current body of knowledge by considering a more comprehensive definition of sustainability, which includes aspects such as social and economic sustainability [278]. This

knowledge could inspire researchers and practitioners, providing insights for configuring and validating automated approaches to improve system sustainability by reducing design errors. They can also help establish best practices, standards, and guidelines that foster the development of sustainable ML. By sharing these strategies and their impacts, we hope to encourage widespread adoption and discussions within the community.

> **RQ_2—Sustainability Approaches**
>
> *What approaches, metrics, and tools do machine learning practitioners employ in building sustainable ML-enabled systems?*

Finally, the research aimed to highlight ML practitioner's challenges while working on sustainability. Previous research has primarily examined traditional software [67, 138], neglecting the specific challenges associated with ML-enabled systems. This perspective can serve both researchers and practitioners in designing novel solutions and enhancing the current state of practice.

> **RQ_3—Sustainability Challenges**
>
> *What are the challenges faced by machine learning practitioners when building sustainable ML-enabled systems?*

To address the research questions, we applied a *mixed-method* research approach [186] with two main phases. The structure of this chapter was based on a similar study conducted by Al-Subaihin et al. [9], which employed a mixed-method approach using interviews and surveys. The first phase is an explorative *qualitative* study; we conducted semi-structured interviews—following established guidelines [167]—with eight practitioners and analyzed the collected data using *Socio-Technical Grounded Theory (STGT) for data analysis* [157] incorporating techniques such as open coding, constant comparison, and memoing. The focus during this analysis was on uncovering and explaining socio-technical phenomena related to ML practitioners' work practices regarding sustainability [343]. This strategy was adopted based on the consideration that the integration

of sustainability aspects into software development is a complex phenomenon intrinsically linked to human factors. For this reason, a purely quantitative approach would not have been sufficient to capture all relevant aspects. Consequently, a qualitative approach was chosen, as it allows greater flexibility and adaptability throughout the research process, enabling adjustments based on emerging findings [129, 157, 284, 324, 325].

However, the insights from the initial qualitative phase were limited in scope and not necessarily representative of a broader population. To address this limitation, we conducted a second *quantitative* phase using an online survey. This approach allowed us to gather responses from a more diverse sample—spanning various demographics, professional roles, and geographic locations. To conduct this study, we followed the guidelines established by Dillman et al. [90] and Kitchenham [198] for survey studies. The large-scale survey provided quantitative insights that complemented the qualitative findings, thereby enhancing both the robustness and validity of the research. In this way, the survey served as an additional step, ensuring that the conclusions were grounded in both the in-depth qualitative insights and the broader quantitative validation.

For the result reporting, we employed the guidelines by Wohlin et al. [368] and Hoda [157], following the ACM/SIGSOFT Empirical Standards.[2] In the reminder of this section, we describe the overall study design, as depicted in Figure 3.1.

3.2.1 *First Phase: Qualitative Study*

As highlighted at the beginning of this section, the first step in the research process focused on gaining an in-depth understanding of the phenomenon through a qualitative approach. As already mentioned, I employed *STGT for data analysis* method proposed by Hoda [157]. This method emphasizes data collection techniques that yield natural language text, which can be effectively obtained through semi-structured interviews.

[2] Available at: `https://github.com/acmsigsoft/EmpiricalStandards`. Given the nature of the study and the currently available standards, we followed the *"General Standard"*, *"Questionnaire Surveys"*, *"Qualitative Surveys"*, and *"Mixed Methods"* guidelines.

Table 3.1: Structure and examples of questions of semi-structured interviews.

Question	Rationale
Part 1: Introduction and Welcome	
What is your nationality?	Understanding participants' nationality helps contextualize their perspectives on sustainability in ML-enabled systems.
What is your highest academic degree?	Understanding participants' academic background helps contextualize their perspectives on sustainability in ML-enabled systems.
How would you describe your company's business?	Identifying the industry sector allows for the analysis of sustainability approaches in different organizational contexts.
How many employees does your company have?	Company size can influence the adoption of sustainability practices and the availability of resources to implement them.
What is your corporate role?	The role of the participant can affect their involvement in sustainability-related decisions within ML projects.
How many years have you worked in this role?	Experience level provides insight into whether sustainability considerations evolve with industry tenure.
What is your work field?	The specific work domain helps assess how sustainability challenges vary across different ML applications.
Part 2: Sustainability Perception (RQ_1)	
How do you perceive the concept of sustainability? Particularly from a "Social", "Environmental" and "Economic" point of view.	This question helps explore how ML practitioners conceptualize sustainability across multiple dimensions.
Among the sustainability domains, which one (out of the three defined above) do you attach the most value to during the development of an ML project?	Identifying the most valued sustainability dimension provides insights into prioritization biases among ML practitioners.
Can you sort the sustainability dimensions based on your importance?	Understanding the relative importance of sustainability aspects helps contextualize practitioners' decision-making processes. This ranking question was included to mitigate bias by forcing a prioritization decision among dimensions.
Part 3: Sustainability Approaches (RQ_2)	
What tools and metrics do you adopt to improve sustainability aspects?	This question identifies the specific tools—and associated metrics—used to incorporate sustainability into ML development.
What methodologies do you adopt to improve sustainability aspects?	Investigating methodologies provides insight into structured processes for embedding sustainability into ML workflows.
What frameworks do you adopt to improve sustainability aspects?	Understanding the frameworks practitioners rely on can highlight existing best practices and gaps in sustainability integration.
How does the development of ML systems change by adopting the described practices?	Examining changes introduced by sustainability practices allows for evaluating their real-world impact on ML development.
Once you have adopted these sustainable practices, do you notice a decrease in the accuracy or overall quality of the models?	This question investigates potential trade-offs between sustainability and model performance, a key concern in ML adoption.
What are the main trade-offs of these approaches?	Understanding trade-offs allows for the identification of barriers that might prevent practitioners from adopting sustainability measures.
Part 4: Sustainability Challenges (RQ_3)	
What are the main challenges you face when developing sustainable ML systems?	Identifying primary obstacles helps outline key barriers to sustainability adoption.
What are the main challenges you face in integrating these approaches and how successful are they in improving sustainability in the ML systems?	Evaluating challenges in integration helps determine whether sustainability initiatives achieve their intended impact.
Could you tell if you have any additional problems, needs or other issues when dealing with the development of sustainable ML systems?	Exploring additional concerns can uncover previously unaddressed barriers in sustainability adoption.
Could you share some examples of challenges and how they were addressed?	Case studies and practical experiences provide concrete insights into overcoming sustainability-related obstacles.

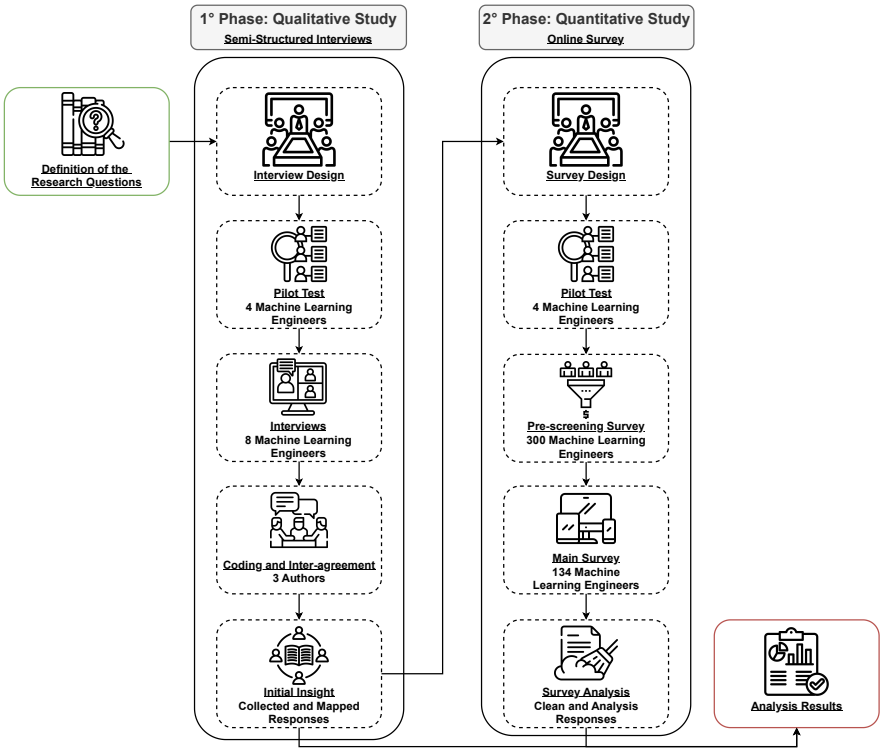


Figure 3.1: Overview of the research process.

The following section provides a detailed description of these methodological aspects.

DESIGN OF THE SEMI-STRUCTURED INTERVIEW The questions for the semi-structured interviews were developed according to the guidelines established by Hove and Anda [167]. I used semi-structured interviews because they blended predefined questions (designed to explore the primary research topic) with open-ended inquiries (intended to uncover unforeseen insights). This qualitative approach is commonly employed to collect in-depth insights, perspectives, and personal anecdotes from participants [74], all while ensuring consistency and comparability across interviews. I initially created the interview design and then refined it collaboratively

with the other authors during recurrent meetings. It is organized into four parts, as illustrated in Table 3.1.

The interviews followed a funnel model, beginning with broad questions and gradually moving toward more specific topics. This approach facilitated a natural conversational flow, allowing participants to speak freely before the interviewer guided the discussion to more focused areas [9]. At the outset, participants were informed about the study's primary focus and objectives, as well as the following conditions: (1) their participation was voluntary; (2) they were free to withdraw from the study at any time; and (3) their responses would be aggregated and used for a scientific publication. Once the participants agreed to these terms, I proceeded with the interview.

Before conducting the interviews, I performed a pilot study with four ML practitioners [245]. This phase was crucial for identifying potential issues in the interview process, such as unclear questions or challenges in maintaining a natural flow of conversation. The pilot participants, each with at least two years of experience in ML, confirmed the quality of the structure and content of the interview. Additionally, I assessed the duration of the pilot interviews to ensure they were appropriately brief.

STRUCTURE OF THE SEMI-STRUCURED INTERVIEW As is well known, semi-structured interviews involve a general set of questions designed to initiate discussion with the participant, which can (and should) be tailored based on the specific context. In particular, the interview approach followed a structured strategy consisting of four key phases. Table 3.1 presents a selection of these questions, generalized across all conducted interviews.

1. **Introduction and Rapport Building:** The participant was welcomed and made comfortable to establish a rapport. The interviewer(s) and the interviewee initially introduced themselves. Afterwards, technical details were provided, and basic demographic information was collected.

2. **Understanding Sustainability:** Immediately afterward, in line with the first research question, the participant was introduced to a definition of sustainability encompassing all relevant dimensions. They were then asked to share and discuss their personal understanding of sustainability through an open-ended question—in this way, I collected unprompted and spontaneous views on the topic. This initial step was purposefully designed to mitigate potential biases such as social desirability or acquiescence [137], as it allowed participants to express their thoughts freely without being influenced by predefined categories or response options. Only after this exploratory question, did I introduce two structured formats: (i) a Likert-scale assessment where participants rated the importance of the three sustainability dimensions (environmental, social, and economic), and (ii) a ranking task where they were asked to order these dimensions based on perceived importance. The combination of these formats enabled both a broad understanding and a prioritization of sustainability concerns.

3. **Sustainability Approaches in Practice:** The third phase focused on the second research question, where participants were asked about the approaches adopted in their workplace to support sustainability. This included inquiries about their awareness and use of tools, as well as whether they considered sustainability-related metrics for evaluation. Additionally, participants were asked whether these approaches had a tangible impact.

4. **Challenges in Sustainable Software Development:** In the fourth phase, the discussion shifted to the final research question, addressing the challenges encountered when attempting to support sustainability or, more broadly, when dealing with this topic in software development. Participants were asked to reflect on the key challenges they typically encounter when addressing sustainability in practice and to share real-world examples of the difficulties they face in integrating sustainability principles into ML-enabled sys-

tems. This approach enabled us to gain practical insights into the constraints, trade-offs, and decision-making processes that shape sustainable ML development.

5. **Conclusion and Final Remarks:** Finally, the interview concluded with a farewell, giving participants the opportunity to ask any remaining questions and receive a recap of key information relevant to them. They were sincerely thanked for their time and effort in contributing to the discussion.

INTERVIEW SAMPLE AND PROCEDURE As for the participant selection process, I opted for a convenience sampling method [15, 148]. This approach involves non-probabilistic sampling, where individuals are chosen based on proximity, availability, or willingness to participate in the research. However, it is worth noting that this method does present limitations regarding the generalizability of findings [148]. For such a reason (other than conducting a survey study described in Section 3.2.2), I identified a set of criteria that the participants had to meet.

- I targeted Machine Learning Practitioners, i. e., ML Engineers, Data Scientists, Artificial Intelligence Engineers, and Software Engineers who primarily focus on producing ML software [31, 245].
- These practitioners were required to be employed at organizations actively involved in ML-enabled system development and demonstrate an interest in sustainability.

Starting from a candidate set of 35 ML practitioners from my network, I conducted interviews in an iterative approach of data collection and analysis, until reaching theoretical saturation [300], i. e., when no new subtopics are identified in different interviews. In the end, 8 interviews were jointly conducted by all the authors of the paper (Table 3.2 reports the participant characteristics). As shown, most participants have a rather long experience with the development of ML-enabled systems and work for medium-large companies having different business goals. As such,

Table 3.2: Information of the interview study participants.

ID	Nationality	Years of Exp.	Role	Company Size	Degree	Work Domains
P1	Italy	4	ML Engineer	0-10	Master's Degree	Medical and Scientific
P2	Brazil	8	ML Engineer	50-200	Master's Degree	Economic and Political
P3	Italy	6	ML Engineer	0-10	Master's Degree	Medical and Scientific, Technical
P4	Italy	1	Data Scientist	Over 500	Master's Degree	Socio-Cultural
P5	Italy	3	Data Engineer	250-500	Bachelor's Degree	Economic and Political
P6	Italy	1	AI Engineer	Over 500	Master's Degree	Economic and Political
P7	Switzerland	5	Software Engineer	Over 500	Master's Degree	Technical
P8	Germany	5	Software Engineer	250-500	Master's Degree	Economic and Political

Notes on the values in the last column: Medical and Scientific: applies ML in psychology, biology, or medical research; Economic and Political: focuses on finance, economy, or politic fields; Technical: focuses on information systems, transportation or urbanistic; Socio-Cultural: uses ML for social trends, education, or sports;

they could provide a diverse set of insights into the sustainability practices employed in different contexts.

The interviews were conducted through *Google Meet*[3] and *Microsoft Teams*,[4] depending on the instrument preferred by the practitioners. On average, the interviews took for 36 minutes, ranging from a minimum of 19 to a maximum of 55 minutes.

DATA ANALYSIS I recorded the interviews and extracted a transcription of them. On such data, I applied *STGT for data analysis* [157]. This approach includes techniques such as open coding, constant comparison, and memoing, which facilitated structured but flexible data analysis.

The transcribed data and collected information were stored in a CSV file for analysis. The coding process began with a first phase of *open coding* and *in-vivo coding*, aimed at identifying and labeling significant segments of text based on participants' information and recurring themes. In this phase, the first three authors independently analyzed separate transcripts to ensure initial diversity in perspectives. In the second phase, *conceptual coding* was conducted to organize the identified codes into broader concepts. The three authors reviewed each other's coding work and met to discuss and reconcile any discrepancies. Disagreements were systematically addressed through discussion, during which authors presented their

[3] https://meet.google.com/

[4] https://www.microsoft.com/microsoft-teams/group-chat-software?rtc=1

interpretations and sought consensus. To ensure rigor, I employed Cohen's Kappa [70] as an inter-rater reliability measure. Each transcript's coding was assessed on a 0–5 scale to validate agreement among the authors, with 0 indicating *Strongly disagree* and 5 *Strongly Agree*. A Cohen's Kappa value of 0.75 or higher was considered acceptable, indicating substantial agreement. Once consensus was reached, the concepts were further grouped into categories, representing higher-level themes. The categorization process was iterative and relied on group discussions to refine the findings, minimizing subjective bias.

To maintain transparency and reproducibility, all details of the coding procedure and resulting categorizations are available in the online appendix [81].

3.2.2 *Second Phase: Quantitative Survey*

As a second step of the methodology, I operationalized a large-scale survey study.

DESIGN OF THE ONLINE SURVEY In designing the survey study, I adhered to the guidelines outlined by Dillman et al. [90] and Kitchenham et al. [198]. Dillman et al. [90] tailored survey design method offers a structured framework for enhancing response rates and data quality. This method emphasizes principles such as customized survey distribution, clear questions, and active respondent engagement. Additionally, Kitchenham et al. [198] empirical research guidelines in software engineering, particularly within the context of surveys, guided the choices regarding questionnaire design, sampling strategies, and data validation. By following these approaches, I aimed to reduce potential bias in participants' responses and ensure that the questions were easy to understand. Moreover, following ethical standards, I provided participants with an informed consent document and deliberately avoided collecting personal information, such as gender, age, or email addresses. The focus was solely on gathering role and company-related information. Notably, the survey in-

corporated attention checks to identify participants who provided careless responses, thus for reliability reasons.

Practically, I designed two questionnaires; the first (pre-screening survey) served as a preliminary screening tool to identify individuals matching the ideal criteria (better discussed in the following sections), while the second (main survey) was the survey. An iterative pilot test approach was employed to (1) evaluate the clarity and quality of the survey and (2) estimate the completion time. The sample group consisted of four individuals with at least one year of experience in the ML working context, and they were different from the individuals in the first phase, but shared similar characteristics. The ML practitioners validated the surveys and provided constructive feedback from various perspectives, which was incorporated to improve the overall quality of the questionnaires before they were distributed online.[5] Additionally, I analyzed the time it took pilot testers to complete the surveys, allowing us to estimate a completion time of 10 minutes.

STRUCTURE OF THE ONLINE SURVEY In terms of structure, the pre-screening survey comprised eight closed-ended questions designed to assess respondents' knowledge and experience regarding the sustainability of ML-enabled systems. Specifically, questions were included to determine their academic background in sustainability and ML. Examples of these questions include: "How familiar are you with the sustainability considerations of machine/deep learning?" and "Have you been involved in any projects where sustainability was a significant factor?". Participants with limited knowledge of sustainability and ML were not selected for the main survey. This pre-screening process ensured that responses were gathered from individuals with relevant expertise.

[5] More specifically, some redundant questions were removed, and typos that were identified were corrected.

Regarding the main survey, it was structured into five sections, aligning with the research objectives and mirroring the structure of the qualitative study. A summary of each section is provided in the following subsection.[6]

1. **Background Information:** The first section collected demographic and professional background information, including the size of the participants' organizations, their education, years of experience, and the types of business systems they have developed. These questions provided an overview of their involvement in the development of ML-enabled systems.

2. **Description and Importance of Sustainability:** The second section, mapped to the first research question, explored participants' understanding, perception, and the perceived importance of sustainability across its various dimensions.

3. **Sustainability Approaches:** The third section, aligned with the second research question, focused on technical aspects. Participants were asked to evaluate the sustainability approaches identified in the interview study, rating their adoption frequency on a five-point Likert scale ranging from *'Never'* to *'Very Often'*. Additionally, they had the opportunity to report other sustainability approaches through a follow-up free-text box. Similar to the interview phase, participants were also asked to report tools and metrics used in conjunction with these approaches to support sustainability.

4. **Challenges in Sustainable ML Development:** The fourth section, corresponding to the third research question, addressed the challenges faced in ensuring the sustainability of ML systems. Participants rated the relevance of challenges identified through interviews on a five-point Likert scale ranging from *'Not Relevant'* to *'Very Relevant'*. They were also given the opportunity to describe additional challenges through a free-text response.

[6] For clarity, the detailed survey structure, including the full list of questions, is available in the online appendix [81].

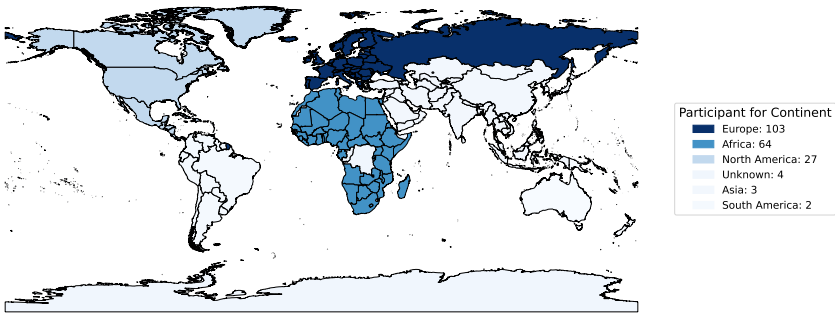


Figure 3.2: Survey participants distribution by continent.

5. **Additional Insights:** In the final section, participants were provided with an open-ended text box to share any additional opinions or insights that could further inform the understanding of sustainability in practice. The survey concluded with a note of appreciation for their time and participation.

SURVEY SAMPLE AND PROCEDURE I selected PROLIFIC[7] as the platform enabling the recruitment of participants and the collection of answers; the platform allows researchers to stipulate specific criteria for participants, i.e., in my case, it allowed limiting participation to ML practitioners. In particular, this is a web-based research tool that has gained wide acceptance among researchers [98, 288] and is recognized as a high-quality recruitment platform when specific criteria—like those I adhered to as outlined—are met [99]. It implements an opt-in strategy [178], which involves participants willingly agreeing to partake in the study. The use of PROLIFIC influenced the design of the study; as such, I followed a set of recommendations provided in the literature [10, 98]:

- A repeated pre-screening method was used to enhance participant selection.

[7] PROLIFIC: https://www.prolific.co/

- The survey incorporated validation questions to verify that the respondents satisfied the selection criteria.
- A layered payment plan was employed, offering a £8 ("good") reward for pre-screening and a £10 ("great") reward for completing the main survey.[8]
- As previously stated, questions designed to check attention were embedded within the questionnaires to verify the reliability of participants.
- I used a tool to detect the presence of AI-generated responses (specifically for open-ended questions) to identify any inappropriate behavior among participants.[9]

I employed a criterion-based purposive sampling approach for participant selection [264]. Participants were chosen based on specific criteria related to their expertise and interest in ML systems. Those who indicated 'not interested', 'never worked with ML,' or 'demonstrated low interest and low knowledge of ML and sustainability', as per their responses in the pre-screening questionnaire, were excluded from the study. Moreover, PROLIFIC provides historical metadata for each participant, such as the number of completed surveys and their approval rate. To guarantee data reliability, I restricted participation to individuals who had completed at least 20 previous tasks and maintained an approval rate of 90% or higher—this was a proxy for ensuring worker quality and engagement. This sampling strategy increased the confidence in having selected individuals with relevant experience, credibility, and genuine interest in the subject matter.

On August 25, 2023, the pre-screening phase began with **300** participants recruited through PROLIFIC. During this stage, five participants did not complete the survey and 44 were removed for failing attention check questions; to maintain the required sample size, PROLIFIC provided 49 substitutes. In addition, 150 participants were excluded from the initial

[8] Prolific rates compensation in four levels—low, fair, good, and great—according to the payments made to participants.

[9] The tool was ZeroGPT: https://www.zerogpt.com (February 2025)

pool due to a lack of experience with ML systems and sustainability knowledge. Two days later, on August 27, 2023, the main survey was submitted to the remaining **150** participants. Of these, six participants were discarded after failing an attention check question, and 10 did not complete the questionnaire, resulting in a final set of **134** valid participants.

The initial data collection involved 134 participants, of which 64 (≈48%) were from Africa, with the remaining participants coming from various other regions. To incorporate a broader range of perspectives and address potential geographic and cultural confounding factors, I decided to repeat the recruitment process. In this second phase, I conducted another pre-screening and full survey, leveraging PROLIFIC's settings to target underrepresented regions and achieve a more balanced distribution of responses.

The second pre-screening was conducted on January 15, 2025, and 130 people were recruited. Of these, 31 did not complete the survey, and 51 were eliminated due to failed attention check questions or lack of experience in ML systems/sustainability knowledge (through PROLIFIC, I got 43 substitutes). The second main survey was submitted on January 17, 2025 (two days later) on a sample of **91** people, of whom 17 did not complete the survey and 5 were discarded because they failed an attention check question.

As a consequence, the final set is composed of **203** valid participants. Figure 3.2 illustrates the continent of each participant; however, four participants chose not to answer. Most of them have a Bachelor's degree (102), followed by a Master's degree (58), a High school diploma or equivalent (34), and a Ph.D. (9). Of the 203 participants, 39.41% define themselves as Software Engineers, 16.26% Data Scientists, 15.76% Machine Learning Engineers, 15.27% Data Engineers, and the remainder held other roles, e. g., ML model Tester and Artificial Intelligence Engineer. Most work in Technical Domains (137) and the Economic and Political domains (60) while the remainder are in Socio-Cultural (49) and Medical and Scientific (32). The distribution of employees across company sizes is as follows: 10.34% (21) of participants work in companies with fewer than 10 em-

ployees; 18.72% (38) of participants are part of companies with a range of 11-50 employees; 28.08% (57) of participants are employed in companies between 51 and 250 employees; 11.33% (23) of participants are employed in organizations between 251 and 500 employees; and, finally, 31.03% (63) of participants are employed in large companies with over 500 employees. Only one (0.49%) participant does not remember the company size. The experience of developers is distributed as follows: 40.39% (82) of developers have less than 2 years of experience, followed by 35.47% (72) with 3-5 years of experience. A smaller number, 10.34% (21), have 6 to 9 years of professional experience. Finally, those with more than 10 years' experience make up 13.79% (28) of the sample. This distribution clarifies the various levels of experience within the group of surveyed developers.

DATA CLEANING AND ANALYSIS Upon reception of the participants' answers, I of the paper went through each of the individual responses to verify their reliability and credibility (using attention check questions, AI detector tools, and manual checking). The goal was to identify potential cases where participants may not have taken the task seriously or lacked sufficient experience to provide meaningful insights. From this analysis, 51 answers were found to be suspicious: therefore, a further discussion with the second and third authors of the paper was opened to decide whether these answers should have been accepted. At the end of the discussion, 14 of them were still considered valid.

Once the data cleaning phase was completed, I analyzed the valid answers in two different manners. When considering close-ended questions, I computed statistics, e.g., frequency analysis and plots, to describe the trends observed in the participants' responses. As for open-ended questions, I analyzed them using *open coding* and studied word frequencies to find patterns. The open coding exercise was conducted by the first three authors of the paper, who jointly collaborated toward the elicitation of the main concepts the answers let emerge. As in the first study, the process involved collaborative discussion and consensus to extract new concepts. Following the mixed-method approach, when a concept overlapped with

one previously identified in the qualitative study, it was not newly created, but rather used to strengthen the existing concept.

3.2.3 *Connecting the Dots Between Qualitative and Quantitative Findings*

The final phase of the study involved integrating and synthesizing the findings from both the qualitative and quantitative research stages. The survey results were systematically combined with the insights gathered from the semi-structured interviews using a qualitative synthesis approach. This phase aimed to align and cross-validate the findings, ensuring that the conclusions were robust, well-substantiated, and reflective of the diverse perspectives within the ML practitioner community. The iterative and collaborative integration process, whose results are detailed in Section 3.3, began with a systematic comparison of key themes and patterns that emerged from the qualitative interviews against the statistical trends identified in the survey responses. Areas of convergence were identified, confirming aspects of sustainability practices that were consistently highlighted across both data sources. For instance, the main challenges identified in the interviews were validated by the prevalence of these concerns in the survey results. Conversely, areas of divergence were critically examined to understand discrepancies. Instances where survey participants reported different perspectives from those expressed in the interviews were analyzed through follow-up discussions among the authors. Such cases were explored in depth, considering contextual factors such as differences in professional roles, company size, and experience levels.

Following this synthesis, the integrated results were further contextualized within the existing body of literature (Section 3.4). A comparative analysis was conducted between the findings of this study and prior related works on sustainability in ML-enabled systems. In particular, a triangulation process was carried out to assess the plausibility of the approaches and challenges reported by ML practitioners. Specifically, for each observation emerging from interviews and surveys, I reviewed the existing literature to identify supporting evidence or comparable results. When

relevant studies were identified, e.g., confirming the sustainability benefits of cloud adoption, model compression, or use of pre-trained models, they were cited directly in the results section. Conversely, in cases where no academic evidence was available, I explicitly acknowledged this absence and framed the practitioner insights as exploratory, suggesting them as potential directions for future empirical research. This process allows readers to clearly distinguish between practices grounded in prior literature and those primarily based on field observations, thereby enhancing both transparency and interpretive depth. The process described above aligned with the principles of Grounded Theory [129, 157, 324, 325], ensuring that the study's contributions extended beyond mere descriptive insights and instead provided an enriched understanding of sustainability in ML practices. The literature comparison helped position the findings within the broader discourse, identifying how the study corroborated previous findings, other than areas where this study advanced the state of knowledge in a non-biased manner.

3.3 ANALYSIS OF THE RESULTS

In this section, we report the results of the analysis on both the interviews and the survey.

3.3.1 RQ_1 — *Sustainability Perception*

The first research question aimed to investigate how ML practitioners perceive sustainability and how they prioritize its different dimensions in terms of importance. To mitigate potential bias in participants' responses, such as social desirability or acquiescence, we adopted a multi-step questioning strategy. First, participants were asked to describe sustainability in their own words, without predefined categories. Only after this, we presented structured questions, including Likert-scale items and a ranking question, to capture both perceived importance and forced prioritization of sustainability dimensions.

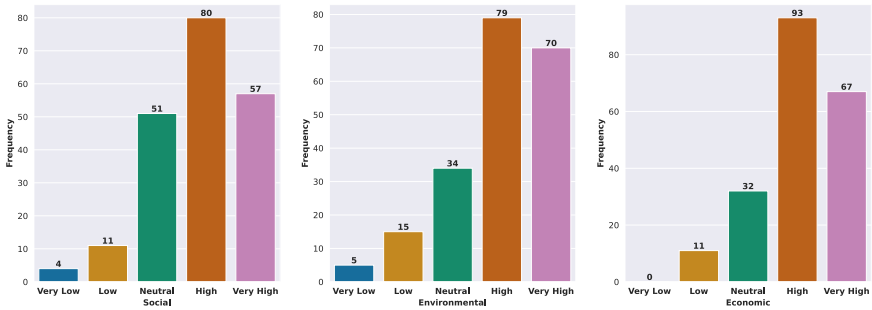


Figure 3.3: Sustainability rate for survey respondents.

From the interviews, P3, P4, P5, and P6 described sustainability in ML software as a trade-off, emphasizing that achieving sustainability often requires a slight reduction in efficiency to minimize negative environmental and societal impacts. Additionally, P3 and P6 highlighted that "good" sustainability entails *"making processes efficient while maintaining the same level of effectiveness"*.

Conversely, P1 and P7 perceived sustainability less as a trade-off and more as an ethical responsibility. For instance, P1 defined sustainability as *"something that causes as little harm as possible to society and the environment"*, while P7 described it as *"the awareness and understanding of the potential consequences that an action may have in the future"*. Unlike the trade-off perspective, these views underscore the moral obligation to integrate sustainability into decision-making, emphasizing the responsibility of those involved in the process. Furthermore, when asked about the relative importance of sustainability dimensions (i. e., social, environmental, and economic), participants consistently ranked the economic dimension as the least important, followed by the social dimension, with the environmental dimension being considered the most critical.

Moving to the survey, the word cloud illustrated in Figure 3.5, was crafted to provide a concise and insightful representation of the predominant terms encapsulating the 184 respondents' perceptions of sustainability. We have extracted the most frequently used terms from each respondent's definition and calculated their respective frequencies. The top three terms,

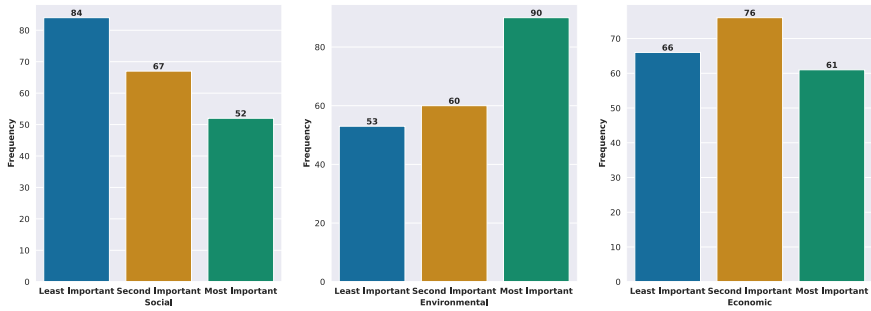


Figure 3.4: Sustainability importance for survey respondents.



Figure 3.5: Word cloud generating analyzing survey participants responses for the first research question.

along with their frequencies, are *Future/Future generations* (34.62%), *Social* (27.07%), and *Resources* (23.33%). Figures 3.3 and 3.4 display the rates and the importance assigned to each sustainability dimension (i.e., social, environmental, and economic) by the survey respondents. As depicted in Figure 3.3, practitioners predominantly perceive all three dimensions as highly or very highly important. In terms of ranking these dimensions, Figure 3.4 reveals that the environmental dimension is considered the most important in the majority of cases (90 out of 203 respondents), followed by the economic dimension, with the social dimension being rated as the least important in 52 cases.

Overall, the findings indicate that ML practitioners in the study perceive sustainability through two primary lenses: as a trade-off involving efficiency reductions to mitigate environmental and societal impact, or as an ethical responsibility emphasizing long-term consequences and minimal harm. While interviewees provided nuanced perspectives on these aspects, survey results further highlighted the prominence of sustainability from practitioners' viewpoints. The word cloud analysis (Figure 3.5) revealed that terms such as *Future/Future generations*, *Social*, and *Resources* were most frequently mentioned, reinforcing the forward-looking and multi-dimensional nature of sustainability concerns. Additionally, quantitative findings (Figures 3.3 and 3.4) demonstrated that practitioners consider all three dimensions—social, environmental, and economic—as important, albeit with varying degrees of emphasis. In the study, while the environmental dimension was ranked as the most critical by the majority, the economic and social dimensions were perceived as secondary, reflecting a broad yet stratified prioritization of sustainability concerns in the ML approach.

> **RQ_1 — Sustainability Perception**
>
> *In the study, ML practitioners perceive sustainability either as a trade-off requiring efficiency reductions or as an ethical responsibility focused on long-term impact. Survey results reinforce this dual view, with key terms like* Future/Future generations, Social, *and* Resources

> *frequently mentioned. While all three sustainability dimensions are considered important, the environmental dimension is ranked highest, followed by the economic and social aspects.*

3.3.2 RQ_2 — *Sustainability Approaches*

The second research question aimed to explore the approaches used to support sustainability in software development, with a particular focus on the tools and metrics adopted by ML practitioners. To further improve the clarity and interpretability of the findings for $\mathbf{RQ_2}$, we grouped the identified sustainability-related approaches into five thematic categories. This categorization was developed through a bottom-up analysis of the approaches' functional goals and their application context. The five resulting categories are: (i) *'Cost and Resource Efficiency'*, (ii) *'Data Management'*, (iii) *'Model Optimization'*, (iv) *'Hardware and Infrastructure'*, and (v) *'Transparency and Monitoring'*. Table 3.3 summarizes the groups with short descriptions and the list of associated approaches. It is worth remarking that the table reports all the approaches emerged in response to $\mathbf{RQ_2}$, including those emerging from both interviews and the survey. In Section 3.3.2.1, we focus on the 16 approaches derived from the interviews (**A.1** to **A.16**). The remaining three approaches (**A.17** to **A.19**), which emerged directly from survey responses, are discussed separately in Section 3.3.2.2 to explicitly preserve the distinction between data sources. Each approach is accompanied by a list of references when supported by scientific literature, as identified through the triangulation process described in Section 3.2.3. When no supporting literature was found, we explicitly acknowledge the lack of evidence and frame the practice as exploratory.

3.3.2.1 *Approaches from the Interviews*

From the interviews, participants described several approaches (A) that influence sustainability. To enrich the discussion of these results, we include

Table 3.3: Categorization of Sustainable Approaches

Category	Description	Approaches
Cost and Resource Efficiency	Approaches targeting reductions in financial cost or resource consumption of ML-enabled systems.	(A.1) Selecting Efficient Libraries and Languages
		(A.2) Optimizing Code to Minimize Resources
		(A.3) Adopt Strategies for Efficient Management of ML Costs
Data Management	Approaches focused on improving data quality and efficiency in the preparation phase of ML-enabled systems.	(A.4) Consider Ethical Data Management
		(A.5) Clean and Optimize Dataset
		(A.6) Consider Train with Synthetic Data
Model Optimization	Approaches that aim to reduce the computational complexity and resource demands of ML-enabled systems.	(A.7) Consider Adopting Pre-trained Models
		(A.8) Consider Streamlining Models for Efficiency and Cost Reduction
		(A.9) Preferring Simplicity in System Design
		(A.10) Consider Delegating Learning to Exploration
		(A.17) Consider Knowledge Distillation
Hardware and Infrastructure	Approaches that improve sustainability by adapting or optimizing the underlying computational environment of ML-enabled systems.	(A.11) Consider Adopting Cloud Technologies
		(A.12) Consider Implementing Offline Predictions
		(A.13) Evaluating the Suitable Model for the Task
		(A.14) Operational Sustainability in AI Devices
		(A.18) Consider New Hardware
Transparency and Monitoring	Approaches aimed at increasing the observability and ethical accountability of ML-enabled systems.	(A.15) Consider Measuring Ethical Aspects
		(A.16) Track Sustainability Metrics
		(A.19) Consider Integrate Model Explainability

selected quotations that illustrate the participants' original phrasing and provide contextual grounding for each approach.

COST AND RESOURCE EFFICIENCY This group comprises three approaches that focus on optimizing code, minimizing unnecessary computation, and selecting efficient execution strategies to lower energy use and financial costs.

(A.1) Selecting Efficient Libraries and Languages [182, 232]: P2, P3, and P7 report the approach to reducing power consumption and costs by carefully selecting efficient libraries and languages. For instance, P1, when analyzing the execution time, reported *"From an economic and environmental point of view, Python is the worst."*. Additionally, P1 described *"Is very simple and full of libraries to use for convenience in the first phase without standing there and wasting too much time on it. For the training part, we use C language."*. Practitioners continually seek ways to reduce resource consumption through model optimization or efficient training techniques.

(A.2) Optimizing Code to Minimize Resources [66]: P1, P2, P3, P5, and P6 minimize expenses and energy usage through strategies such as reducing hardware utilization and preventing the wastage of unused resources. They achieve this by optimizing the quality of their code for both CPU and GPU usage in the case of P1 and P2. P1 highlighted *"It is necessary to optimize the use of the internal memory of the graphics processing units (GPUs), as the amount available is often insufficient."* Additionally, says *"While GPUs are designed to optimize tensor processing, images are processed by the CPU, creating a performance problem and limitations."*.

(A.3) Adopt Strategies for Efficient Management of ML Costs [134, 287]: All interviewees share a common objective of minimizing energy consumption and reducing training time, indirectly leading to decreased development costs. These approaches are consistently implemented

across the board, with some participants like P1 and P4 halting training once a specific accuracy threshold is reached, thereby respecting business constraints and emphasizing the importance of improving data quality rather than continuing to train the model. Additionally, P6 reported that, *"for example, the drones we know fly for 1:45 hours with a 15-watt decision system. If the decision system is at 10 watts, the flight time can increase from 1:45 to 1:52 hours. It means they cover a larger area, so not only do you create a solution that has a smaller impact from an energy standpoint but also a smaller impact on the number of drones to be released because if these fly for 1:45 hours, I need three, maybe if they fly for 2 hours, I can only use two."*

DATA MANAGEMENT This group includes three approaches centered on improving the sustainability of ML systems through better data handling. The practices focus on data preparation, quality assurance, and preprocessing.

(A.4) Consider Ethical Data Management [353, 373]: P5 highlighted the importance of anonymizing sensitive user data before training ML models, using techniques such as MD5 hashing and Hashbytes. This approach is aimed at promoting social sustainability by safeguarding user privacy. P5 further explained: *"The data that we then provide to the business is anonymized, hiding personal information either through encryption or through dummies."*

(A.5) Clean and Optimize Dataset [182, 296]: P1, P2, P5, and P7 emphasized that before training an ML model, there is a need for high-quality data and the need to reduce the size of the dataset. This process is done through the removal of redundant values that increase the use of energy and storage resources to train the models.

(A.6) Consider Train with Synthetic Data [296]: Due to the scarcity of authentic data, P6 optimizes the development of their systems in terms of both time and cost efficiency by employing synthetic data for

training purposes. Once a model achieves a specified level of accuracy, it is deployed into a physical device. After the model is deployed into the system, new real-world data is collected, which is then used for further refinement and improvement of the system through continuous training. This strategy accelerates the initial model-building process and facilitates continuous model performance improvement over time.

MODEL OPTIMIZATION The interview participants mentioned four approaches aimed at improving sustainability by optimizing ML models. These practices involve selecting simpler architectures, reducing model size through compression or pruning, and balancing performance with resource efficiency to limit computational and environmental costs.

(A.7) Consider Adopting Pre-trained Models [182, 350]: P1, P2, P6, P7 and P8 prioritize using pre-trained models as a cost-effective strategy, which reduces development costs and lowers energy consumption than training from scratch.

(A.8) Consider Streamlining Models for Efficiency and Cost Reduction [182, 350]: Sustainability strategies, as mentioned by P3, encourage critical evaluation of patterns. This involves identifying "lazy" neurons in neural networks or unnecessary parts of models that can be eliminated. This approach streamlines models, enhancing efficiency without compromising performance. On this, P3 highlights: *"When we take state-of-the-art open source models, especially academics, tend to have less attention, and maybe the network works, but maybe there are a lot of lazy and useless neurons like that, maybe after the network loading is a little slower."*. Additionally, P6 highlights using quantization and low-rank adapter to reduce the model size, describing *"Good functions that filter the input also go a long way to reducing power consumption by ensuring the same performance as the model grows. However, this thing we have seen fails because the number of the parameters is the real bottleneck more than the input size"*

(A.9) Preferring Simplicity in System Design: Participants P1, P2, P3, and P6 opt for simpler model architectures, recognizing that they are cost-effective and easier to maintain while still delivering effective results. P3 reported *"I always try to think in a simpler system. A simple system consumes fewer resources, which has all the good impact."*. Complexity can lead to resource wastage and challenges in long-term management and maintenance. Sustainability strategies encourage keeping systems simple, reducing resource consumption, and facilitating ongoing maintenance. The triangulation process revealed that this approach has not yet been examined in the scientific literature - hence, it can be considered an emerging practice.

(A.10) Consider Delegating Learning to Exploration [182, 195]: In situations with limited data, participant P3 incorporates reinforcement learning and employ lighter models to reduce cost; P3 said *"If it is possible to use a reinforcement learning approach, we prefer it. In general, we delegate a lot of the learning work to the exploration part, which is always more efficient than the training part, which remains inefficient because, usually, the networks are very small in reinforcement learning models."*.

HARDWARE AND INFRASTRUCTURE This group comprises four approaches focused on enhancing sustainability by acting on the underlying computational environment. These practices involve selecting energy-efficient hardware, optimizing infrastructure configurations, and leveraging cloud or edge resources to reduce resource waste and improve overall system efficiency.

(A.11) Consider Adopting Cloud Technologies [182, 350]: to reduce energy consumption and minimize development costs, participants P2, P3, P4, and P8 rely on the usage of cloud technologies. By doing so, practitioners could benefit from machines optimized for parallel computing and minimize environmental impact. Participant P1 reported *"Currently, we use an in-house machine, which requires a server room with a constant low temperature to handle the heat generated. The associated costs are*

significant in purchasing and maintaining the machine. I recognize that using cloud computing can be beneficial in terms of cost and energy efficiency. Resources in the cloud are used more efficiently than on a dedicated machine, reducing waste and optimizing overall energy consumption".

(A.12) Consider Implementing Offline Predictions: An alternative approach suggested by P1 to reduce costs and energy consumption involves implementing offline predictions on customer PCs. Furthermore, both P1 and P4 prioritize allocating machine resources based on task importance as another strategy for efficiency. According to the triangulation process, this approach has not yet been addressed in the scientific literature. It may therefore be seen as an emerging approach that warrants further investigation by the research community.

(A.13) Evaluating the Suitable Model for the Task [329]: Before embarking on solution development, participants P1, P2, P6, P7, and P8 meticulously evaluate and select the most appropriate model for a specific task. Their focus is on cost reduction and performance enhancement. P4 reports *"If I have a model of 8 GB and another one of 4 GB. Even if I lose two percentage points, whatever, I take the smaller one, so I don't have to get the bigger machine."*. P1 described for the preprocessing phase *"Optimization is a key element in managing time, memory, and cost in the machine learning process, with particular complexity in choosing the architecture and optimizing the processing."*. As emphasized by P1, P2, and P3, sustainability strategies drive the adoption of more resource-efficient development approaches. This is particularly crucial in machine learning, where training models demand substantial time and computing power. Practitioners are motivated to find ways to reduce training time and optimize hardware utilization.

(A.14) Operational Sustainability in AI Devices [48, 66]: P6 highlighted the need to operationalize sustainability in IoT or memory-constrained systems, where input and model sizes are limited. In particular, P6 highlight *if you load a model that goes to memory saturation or the number of inferences required, it will shut down.* This approach improves

the use of limited memory and processing capacity and reduces power consumption within constrained systems.

TRANSPARENCY AND MONITORING This group includes two approaches that aim to enhance the sustainability of ML systems by improving their observability and ethical accountability.

- **(A.15) Consider Measuring Ethical Aspects [106, 353]:** Moreover, participant P3 takes steps to measure equity metrics with the objective of addressing social discrimination and fostering responsible approaches in their endeavors.

- **(A.16) Track Sustainability Metrics [182, 350]:** Interviewees emphasized the importance of integrating sustainability metrics throughout the development lifecycle of ML-enabled systems. Monitoring these metrics offers valuable insights into resource consumption and potential system biases, enabling process optimization, cost savings, and the development/operation of more sustainable and responsible systems.

3.3.2.2 *Approaches from the Online Survey*

Moving to the questionnaire, participants were first asked to indicate whether they had implemented approaches that emerged in the interviews.

The majority of respondents reported using techniques aimed at enhancing computational efficiency: 66.50% (135) indicated that their teams optimize CPU/GPU usage to reduce energy consumption, while 56.65% (115) reported utilizing cloud technologies instead of traditional on-premise solutions. Moreover, model optimization strategies were also widely adopted. Notably, 36.45% (74) of participants focused on reducing model size, and 35.47% (72) actively eliminated unnecessary information, such as underutilized "lazy" neurons. Additionally, 33.50% (68) adopted simpler, lighter architectures, and 30.54% (62) leveraged pre-trained models. Furthermore, 24.63% (50) employed reinforcement learning techniques as part of their optimization efforts. As a final note, other approaches were implemented

to a lesser extent. For instance, 25.12% (51) of respondents measured and addressed fairness issues within their models, and 23.15% (47) reduced training time once the desired accuracy was achieved. Strategies such as training with synthetic data 7.39% (15), reducing dataset size 5.91% (12), and improving the sustainability of devices that integrate AI components 5.91% (12) were adopted even less frequently. Lastly, a small portion of the sample (3.45%, 7) reported that they had not implemented any of the specified methodologies. Moreover, survey participants could also report other approaches. From the analysis of the survey responses, we elicited three additional approaches. These contributions complement the interview findings and map to three different groups in Table 3.3, namely: *'Model Optimization'* (**A.17**), *'Hardware and Infrastructure'* (**A.18**), and (v) *'Transparency and Monitoring'* (**A.19**). For all three approaches, the triangulation process identified relevant supporting literature, which we reference in the corresponding descriptions to substantiate their contribution to sustainable ML development.

MODEL OPTIMIZATION This category now includes one additional approach.

(A.17) Consider Knowledge Distillation [182, 350]: Participants emphasize the need to design ML-enabled systems to balance accuracy and resource utilization. They highlighted the use of knowledge distillation to reduce model size without compromising performance. This approach could reduce model size and computational complexity during both training and inference.

HARDWARE AND INFRASTRUCTURE This category now includes one additional approach.

(A.18) Consider New Hardware [329]: If it is not possible to adopt **(A.11)**, developers emphasize not to use old hardware for too long. Sometimes, it is worth spending money on more efficient technologies and processes to reduce development costs.

TRANSPARENCY AND MONITORING This category now includes one additional approach.

(A.19) Consider Integrate Model Explainability [143]: Developers have emphasized on creating interpretable models, balancing complexity and transparency. Simpler, interpretable architectures may be preferred over "black-box" approaches unless necessary. Additionally, tools such as SHAP and LIME can be used to assess the contribution of individual features to model predictions, uncovering potential biases and areas for improvement. Interpretability increases stakeholder confidence and better meet customer needs, as well as facilitates debugging and refinement of ML-enabled systems.

Table 3.4: Frequency of sustainability tool adoption by survey participants.

Sustainability Tools	Frequency	Sustainability Dimension
Microsoft Fair Learn	**41.38% (84)**	Social
CodeCarbon	**33.99% (69)**	Environmental
IBM 360 degree toolkit	**25.62% (52)**	Social
What-if Tool from Google	**23.65% (48)**	Social
Never Used Tools	**22.17% (45)**	–
AI-Fairness 360	**20.69% (42)**	Social
Running Average Power Limit (RAPL)	**12.81% (26)**	Environmental
Aequitas	**6.90% (14)**	Social
NVIDIA Triton Inference Server	**5.42% (11)**	Environmental
SHapley Additive exPlanations (SHAP)	**0.98% (2)**	Social
Local Interpretable Model-agnostic Explanations (LIME)	**0.49% (1)**	Social
Internal Developed Software	**0.49% (1)**	–
Databricks	**0.49% (1)**	Economic
Microsoft Sustainability Calculator	**0.49% (1)**	Environmental

3.3.2.3 *Sustainability Metrics and Tools*

Other than discussing approaches, we also discussed and collected sustainability metrics and tool used by ML practitioners.

Regarding the metrics, most of the information was derived from the survey study. In the interviews, P2 and P3 mentioned measuring training time, while P6 referred to the use of *Triton tools*, a benchmarking suite designed

to assess model performance (inference time) and energy consumption. In contrast, the survey participants identified a broader range of metrics, particularly within the environmental dimension, including energy consumption, CO_2 emissions, and water usage. Metrics related to the social dimension encompassed employee satisfaction and educational attainment rates. Within the economic dimension, commonly adopted metrics included return on investment (ROI), key performance indicators (KPIs), and cost-benefit analysis measures for the models under evaluation. A comprehensive report detailing all the identified metrics is available in the online supplementary material [81]. It is important to note that ROI was not used to define the concept of economic sustainability in the study, but rather as a practical proxy frequently employed by practitioners to evaluate cost-effectiveness. While ROI captures short- or medium-term financial performance, it does not account for broader economic sustainability objectives, which emphasize long-term viability and resource efficiency.

Moving to the tools, most of the gathered information comes from the survey study, with only P6 (from the interview) mentioning the TRITON TOOLS. Table 3.4 provides an overview of the tool usage's frequency among ML practitioners (according to the survey participants) and the sustainability dimension they impact. The most commonly employed tool is MICROSOFT FAIR LEARN, followed by CODECARBON, GOOGLE'S WHAT-IF TOOL, SHAP, LIME, DATABRICKS, and MICROSOFT SUSTAINABILITY CALCULATOR. In contrast, 22.17% (45) respondents reported never using any tools. Regarding frameworks no interview participants use framework, instead, for survey participants 41.87% (85) never used frameworks. Among those who do, the most prevalent frameworks are IFRS (33.99%), CFD (24.63%), and GRI STANDARDS (19.70%), with only 0.49% of respondents relying on internally developed software and ED Framework (0.49%).

RQ_2 — Sustainability Approaches

We identified 19 sustainable approaches, their effects on the development of ML-enabled systems, and categorized them into five groups. The most frequently used approaches are CPU/GPU optimization to

> *reduce power consumption and the adoption of cloud technologies. We also quantified the frequency with which these sustainable approaches, tools, and frameworks are used. The result is a mapping of ML practitioners' current knowledge of sustainability in the study, providing insights into areas of research worth exploring in the future.*

3.3.3 RQ_3 — *Sustainability Challenges*

We then focused on the challenges encountered by ML practitioners in integrating sustainability into software development. We grouped the 14 identified challenges into four thematic categories through a bottom-up analysis of their content and application context. The resulting categories are: (i) *'Norms, Standards, and Guidelines'*, (ii) *'Organizational and Economic Support'*, (iii) *'Skills and Human Resources'*, and (iv) *'Responsibility and Ethics'*. Table 3.5 summarizes these categories with brief descriptions and the list of associated challenges. Similarly to the organization of the results for $\mathbf{RQ}_2$, the table reports all the challenges that emerged in response to $\mathbf{RQ}_3$, including those from both interviews and the survey. In Section 3.3.3.1, we focus on the 14 challenges derived from the interviews (**C.1** to **C.8**). The remaining six challenges (**A.9** to **A.14**), which emerged from the survey, are discussed separately in Section 3.3.3.2. Also in this case, each challenge is accompanied by a list of references when supported by scientific literature. When no supporting literature was found, we explicitly acknowledge this lack of evidence and frame the challenges as exploratory.

3.3.3.1 *Challenges from the Interviews*

The interviews revealed challenges (C) that practitioners faced when dealing with sustainability.

NORMS, STANDARDS, AND GUIDELINES This group includes two challenges that focus on the lack of standardized metrics and limited integration methods, which can hinder sustainable development.

1

Table 3.5: Categorization of Sustainable Challenges

Category	Description	Challenges
Norms, Standards, and Guidelines	Lack of formal references and criteria to guide sustainable development	(C.1) Absence of Standardized Metrics
		(C.2) Limited Integration Method
		(C.9) Lack of Sustainability Standard
		(C.10) Sustainability Lacking in Design
Organizational and Economic Support	Missing support from institutions, funding, or client demand	(C.3) Lack of Awareness
		(C.4) Lack of Government Incentives
		(C.5) Lack of Customer Demand
		(C.11) Lack of Sufficient Financial Support
Skills and Human Resources	Limited knowledge, training, or frameworks for sustainability	(C.6) Lack of Software Frameworks
		(C.7) Lack of Educational Resources
		(C.12) Lack of Staff Training
		(C.13) Sustainability Slows Down Software Development
Responsibility and Ethics	Concerns about ethical risks and accountability in ML-enabled systems	(C.8) Lack of Ethical Trust in ML
		(C.14) Lack of Model Accountability

(C.1) Absence of Standardized Metrics [17, 143]: P2, P3, and P4 raised concerns about the absence of standardized sustainability metrics, which make it difficult to measure and compare sustainability efforts.

(C.2) Limited Integration Method: P1, P2, P5, and P6 pointed out the scarcity of established methods for integrating sustainability into development processes. This lack of guidance may hinder or slow the adoption of sustainable approaches. According to the triangulation process, this challenge has not yet been founded in the scientific literature.

ORGANIZATIONAL AND ECONOMIC SUPPORT This group includes three challenges that focus on the lack of awareness, government incentives, and customer demand, which can undermine the motivation and capacity of practitioners to adopt sustainable approaches.

(C.3) Lack of Awareness [17]: Participant P2 mentioned a lack of public awareness and publicity surrounding sustainable development efforts. This may contribute to a lack of recognition and motivation for sustainable approaches.

(C.4) Lack of Government Incentives [17]: P2, P3, and P5 highlighted the absence of government incentives—both in terms of economic compensation and regulations—to support sustainable projects. P2 said *"I see that the companies that look for green things do that for branding ... I think government regulation because has very strict environmental laws, and I think this is driving the companies towards getting more concerned with this, those subjects and then if it is mandatory they cannot avoid it. We need a little bit more laws and compliance for the companies to follow, then they start worrying about this stuff."*.

(C.5) Lack of Customer Demand: Participant P3 and P5 noted the lack of incentives from customers to adopt sustainable developments. A lack of demand for sustainability from clients can reduce the motivation for companies to prioritize sustainability. This challenge represents a novel concern for sustainability research.

SKILLS AND HUMAN RESOURCES This category includes two challenges related to the lack of software frameworks and educational resources that can hinder practitioners' ability to integrate sustainability considerations in the absence of structured training or supporting infrastructure.

(C.6) Lack of Software Frameworks: All participant highlighted the absence of dedicated frameworks that are specifically engineered to facilitate the incorporation of sustainability considerations into development processes. Also in this case, such a perspective has not yet been discussed by academic literature.

(C.7) Lack of Educational Resources [330]: Participants P1, P2, and P4 indicated a lack of educational courses focusing on integrating sustainability into software systems. This education gap may hinder the adoption of sustainable approaches by developers.

RESPONSIBILITY AND ETHICS This category includes one challenge focused on the ethical implications associated with ML-enabled systems for accountability and societal impact.

(C.8) Lack of Ethical Trust in ML [192, 269]: Participant P5 described the lack of model trust when it is used and its impact on society; Specifically, P5 reported that: *"Then you also have to think about the organization of virtual figures, for example, politicians." "Through AI, you generate text that is not there, and many people can fall for it, i.e., they are not all able to be able to understand what is true and what is false. And also the damage can be very heavy, I mean already now with deep fakes and very simple Fake News irreparable damage is created. Imagine fake politicians talking and saying certain things. There is a lot of work to be done on that."*

3.3.3.2 *Challenges from the Online Survey*

In the survey, respondents were first presented with a list of challenges identified during the interviews and asked to rate their frequency. Table

3.6 reports these frequencies. As shown, the most significant challenges encountered by ML practitioners were the *'Lack of government incentives'* and the *'Lack of established methods'*, followed closely by the *'Lack of motivation'* and the *'Lack of software frameworks'*. While other challenges were still considered relevant, they were perceived as less critical. A small proportion of respondents did not identify any significant challenges, with only 6.40% (13 participants) reporting no difficulties.

Moreover, participants were invited to supplement the list of identified challenges by suggesting additional potential challenges through an open-ended question. These contributions complement the interview findings and map to four different groups in Table 3.5, namely: *'Norms, Standards, and Guidelines'* (**C.9** and **C.10**), *'Organizational and Economic Suppor'* (**C.11**), *'Skills and Human Re- sources'* (**C.12** and **C.13**), and *'Responsibility and Ethics '* (**C.14**). For five challenges, the triangulation process identified relevant supporting literature, which we reference in the corresponding descriptions to substantiate their contribution to sustainable ML development. However, for one challenge, **C.13**, we were unable to find the corresponding reference, so we considered it new.

NORMS, STANDARDS, AND GUIDELINES This category now includes three additional challenges.

(C.9) Lack of Sustainability Standard [17, 143]: ML practitioners encounter significant challenges in promoting sustainability within their organizations. The absence of established standards, strategies to reduce emissions, and clear direction adds to the difficulties in promoting sustainable approaches.

(C.10) Sustainability Lacking in Design [17]: Respondents pointed out that sustainability is an additional requirement to be considered, influencing the overall design of ML-enabled systems and the solutions that are implemented. However, it is often overlooked during the initial stages of model design, making it complicated to integrate sustainable methods at later stages. Practitioners frequently prioritize achieving high performance and meeting specific goals without adequately addressing

the ecological implications of their decisions. As a result, systems tend to be resource-intensive and have a significant carbon footprint.

ORGANIZATIONAL AND ECONOMIC SUPPORT This category now includes one additional challenge.

(C.11) Lack of Sufficient Financial Support [143]: Respondents highlighted the challenge of inadequate financial support for sustainability projects. There is a widespread belief that sustainability initiatives can be costly, and the scarcity of essential funds and resources perpetuates this perception. Achieving sustainability goals often requires substantial financial investment, but the necessary resources are often insufficient to support these initiatives. While C.10 and C.11 may seem related, we differentiate them based on their nature. C.10 refers to the lack of time within development workflows to incorporate sustainability approaches, whereas C.11 reflects a lack of funding, institutional prioritization, or incentives. Notably, time constraints may exist even in well-funded projects, while some low-budget projects may still integrate sustainability successfully due to high prioritization.

SKILLS AND HUMAN RESOURCES This category now includes two additional challenges.

(C.12) Lack of Staff Training [330]: Some participants pointed out a dearth of staff training dedicated to sustainability. Without adequate training and interest, it becomes challenging to instill sustainable values within an organization and equip employees with the skills to drive meaningful change.

(C.13) Sustainability Slows Down Software Development: On a less positive note, some survey responses indicated the presence of certain disadvantages in adopting sustainability approaches during ML development. One significant challenge is the potential slowdown in development time required to ensure effective sustainability improvements. Consequently, there's a need to ensure that software functions correctly

within the constraints imposed by sustainable approaches. Furthermore, sustainability-focused models may exhibit lower accuracy and complexity, potentially leading to quality issues in their performance and depth. These drawbacks highlight the need for a careful balance between sustainability considerations and maintaining the desired model quality and effectiveness level. This concern seems not to be discussed in literature, highlighting the need for future studies to investigate how sustainability goals can be balanced with development speed and model quality in practice.

RESPONSIBILITY AND ETHICS This category now includes one additional challenge.

(C.14) Lack of Model Accountability [143]: ML-enabled systems typically require large amounts of data, which often includes sensitive information for training purposes. This dependence can lead to challenges in data storage and management, as well as a lack of metrics and government incentives that address these issues. Additionally, the increasing risk of privacy violations and potential data breaches poses significant concerns. When the data is of poor quality or difficult to quantify, it can undermine decision-making processes, highlighting the importance of responsible data management. It is crucial to establish partnerships that prioritize user comfort, security, and privacy. Furthermore, ML-enabled systems that are not held accountable can potentially harm the population.

RQ_3 — Sustainability Challenges

We identified 14 sustainable challenges, quantified their frequency, and categorized them into four categories. The analysis reveals that the most common issues include a lack of government incentives, insufficient methodologies, and a lack of motivation. Many respondents highlighted the necessity for increased awareness and training on sustainability approaches, noting that many ML designs prioritize per-

Table 3.6: Frequency of sustainability challenges in ML-enabled systems according to the survey participants.

Sustainability Challenge	Frequency
Lack of Government Incentives	**36.45% (74)**
Lack of Methods	**34.98% (71)**
Lack of Motivation	**33.00% (67)**
Lack of Software Frameworks	**31.53% (64)**
Lack of Metrics	**30.54% (62)**
Lack of Publicity from Companies	**28.08% (57)**
Lack of Customers Intentives	**28.08% (57)**
Lack of Courses	**26.11% (53)**
Lack of Ethical Trust in ML	**20.20% (41)**
Never had problems	**6.40% (13)**

> *formance over environmental impact or fail to consider sustainability during the initial stages of model design. These findings emphasize the need for a collaborative, multi-stakeholder approach to tackle these challenges, promote best approaches, and ultimately drive the development of more sustainable and ethically responsible ML-enabled systems.*

3.4 DISCUSSION AND IMPLICATIONS OF THE STUDY

This study provides several implications at different levels and for different stakeholders, from financial organizations to sustainable software engineering. In the following, we discuss the major implications we envision for the research community in an attempt to provide software engineering researchers and practitioners with actionable insights that may contribute to further addressing the sustainability of ML-enabled systems.

3.4.1 *Positioning the findings within the current state of the art*

The findings from **RQ**$_1$ shed light on how ML practitioners perceive sustainability, offering a resource for researchers seeking to assess the alignment (or lack thereof) between practitioner views and existing theoretical definitions of sustainability. This comparison reveals both convergence and divergence that can inform future research and guide the development of actionable sustainability frameworks for ML-enabled systems.

In line with prior work [242, 347], practitioners broadly embrace a *socio-technical perspective*, recognizing sustainability as a multifaceted challenge requiring trade-offs between performance, resource efficiency, and ethical responsibility. This aligns with the view of sustainability as a *stratified* and *multisystemic* property spanning technical, organizational, and societal levels [242].

However, a key misalignment emerges in how sustainability is operationalized. While the literature promotes a lifecycle-wide, strategic integra-

tion of sustainability, practitioners often reduce it to a set of short-term, system-level trade-offs, particularly related to energy consumption, computational efficiency, and model size. This suggests that sustainability is frequently treated as an *optimization constraint* rather than a guiding non-functional requirements (NFRs).

This gap points to a broader challenge: to make sustainability actionable, it must be reframed as a measurable NFR, integrated early in the development lifecycle. This need is reinforced by recent industry signals. For instance, Gartner's 2024 press release highlights a growing recognition of software sustainability, predicting that by 2027, 30% of large global companies will explicitly include sustainability among their NFRs—up from less than 10% in 2024 [1]. These results contribute to this evolving landscape by capturing how practitioners conceptualize sustainability in real-world settings. These insights can inform the development of pragmatic guidelines, tools, and processes that translate academic principles into approaches tailored to the constraints and priorities of ML development.

Implication #1. There is a gap between practitioners' focus on operational constraints and academic definitions of sustainability. To bridge this gap, sustainability must be integrated as a practical, measurable NFR throughout the development lifecycle. Industry trends, such as Gartner's recognition of software sustainability, reinforce this need, highlighting a shift toward early adoption of sustainability principles in ML workflows.

Furthermore, the investigation into $\mathbf{RQ}_2$ examined the concrete approaches ML practitioners adopt to build sustainable ML-enabled systems. We identified 19 approaches across the three dimensions of sustainability, grouped into five categories. These approaches primarily focus on strategic hardware decisions, individual awareness, and lightweight software strategies. Most approaches relate to environmental sustainability, with an emphasis on energy-efficient hardware and software configurations. Practitioners frequently mention the use of cloud technologies, CPU/GPU optimization, and software techniques to reduce system complexity and size. Perhaps more importantly, these approaches are often applied with-

out dedicated frameworks or tools, highlighting a lack of instrumented support for sustainability-aware ML development.

These findings align with prior work by Järvenpää et al. [187], whose catalog of green architectural tactics also includes hardware optimization (e.g., T24: Use energy-efficient hardware), energy-aware pruning (T15), quantization-aware training (T18), and efficient data management techniques such as removing redundant data (T2) and applying sampling (T1). These approaches contribute directly to lowering computational and energy costs across the ML lifecycle. At the same time, this study reveals additional, underreported approaches, including (A.11) consider adopting cloud technologies from the project's outset, (A.12) consider implementing offline predictions, (A.9) evaluating the suitable model for the task, and (A.6) consider train with synthetic data. These strategies extend the current body of knowledge and can complement the catalog proposed by Järvenpää et al. [187].

Regarding social sustainability, the results share similarities with the study by Habiba et al. [143], particularly in approaches such as (A.15) consider measuring ethical aspects (aligned with P3: Bias and fairness) and (A.19) consider integrating model explainability: (similar to P11: Internal standards). However, the study introduces a distinct focus on (A.4) consider ethical data management, which emphasizes anonymizing sensitive user data prior to model training. Despite these approaches being recognized, the findings highlight a gap between awareness and adoption. Only 24.12% of survey participants reported actively measuring or addressing fairness in their models. Furthermore, while multiple fairness tools exist, their adoption remains fragmented: 41.38% of practitioners reported using Microsoft FairLearn, 23.65% used Google's What-If Tool, and 20.69% relied on IBM's AI Fairness 360, with smaller shares using SHAP (0.98%) and LIME (0.49%). These findings point to a lack of systematic guidance and tool integration, underscoring the need for more accessible, unified solutions to support social sustainability in ML workflows.

Implication #2. ML practitioners of the study primarily focus on hardware efficiency and software optimization. These approaches align with prior research but also introduce new strategies, such as early adoption of cloud technologies and synthetic data usage. However, sustainability efforts are often applied without formal frameworks or tools, highlighting a gap between intention and implementation. This is particularly evident in social sustainability, where fairness evaluations remain limited, and tool adoption is fragmented.

Finally, the study adds a novel contribution to the literature by surfacing approaches and challenges related to economic sustainability in ML-enabled systems - this is a dimension that, to my knowledge, has not been systematically addressed in prior empirical work. The findings show that sustainability in ML is not purely a technical matter to be solved through algorithmic or hardware optimizations. Instead, it is a socio-technical concern, shaped by industry constraints, evolving regulations, and the availability of tools that support sustainable approaches. Economic sustainability plays a critical role in real-world ML adoption, where development costs, resource allocation, and budget limitations directly influence decision-making. Practitioners must balance performance with cost-efficiency, navigating trade-offs among infrastructure investments, computational needs, and long-term maintainability. Regulatory pressures, like energy usage standards and ethical AI guidelines, further compound these challenges, introducing new costs but also spurring innovation in sustainable approaches. By foregrounding economic considerations, this study complements prior work that has focused predominantly on environmental and social aspects. This expanded lens reinforces the need for integrated strategies that address technical feasibility, regulatory compliance, and financial viability across the ML development lifecycle.

Table 3.7: Mapping of Sustainability Approaches to ML-enabled Systems Stages and NFRs. Sustainability Definition (Def).

Approach	Description	Workflow Stage(s)	NFR(s) Addressed
Def	Integrate Sustainability Definition	Model Requirements	Sustainability Dimensions
A.13	Evaluating the Suitable Model for the Task	Model Requirements, Model Training	Cost, Performance, Energy Consumption
A.9	Preferring Simplicity in System Design	Model Requirements, Model Training, Deployment	Scalability, Energy Consumption
A.4	Consider Ethical Data Management	Data Collection, Data Cleaning	Privacy, Fairness, Accountability
A.5	Clean and Optimize Dataset	Data Collection, Data Cleaning	Capacity, Performance, Accuracy, Cost
A.6	Consider Train with Synthetic Data	Data Collection, Model Training	Cost, Accuracy, Performance
A.11	Consider Adopting Cloud Technologies	Data Collection, Model Training, Deployment	Energy Efficiency, Cost, Scalability
A.1	Selecting Efficient Libraries and Languages	Model Training	Cost, Energy Efficiency, Performance
A.2	Optimizing Code to Minimize Resources	Model Training	Cost, Energy Efficiency, Performance
A.7	Consider Adopting Pre-trained Models	Model Training	Cost, Energy Efficiency, Accuracy, Transferability
A.8	Consider Streamlining Models for Efficiency	Model Training	Cost, Energy Efficiency, Capacity, Scalability
A.10	Consider Delegating Learning to Exploration	Model Training	Cost, Performance
A.17	Consider Knowledge Distillation	Model Training	Capacity, Cost, Energy Efficiency, Performance
A.11	Adopt Strategies for Efficient Management of ML Costs	Model Training, Monitoring	Cost, Energy Efficiency
A.15	Consider Measuring Ethical Aspects	Model Evaluation	Fairness, Accountability
A.19	Consider Integrate Model Explainability	Model Evaluation, Monitoring	Interpretability, Transparency, Explainability
A.12	Consider Implementing Offline Predictions	Deployment	Cost, Energy Efficiency, Performance
A.14	Operational Sustainability in AI Devices	Deployment	Capacity, Energy Efficiency
A.18	Consider New Hardware	Deployment	Cost, Energy Efficiency, Performance
A.16	Track Sustainability Metrics	Model Monitoring	Energy Efficiency, Cost, Fairness, Accuracy

Implication #3. These findings reveal that sustainability in ML-enabled systems is a socio-technical issue influenced by industry constraints, regulations, and financial feasibility. While prior research has focused on environmental and social sustainability, we emphasize the need to incorporate economic sustainability, particularly in reducing development costs and ensuring long-term maintainability.

3.4.2 *Using the Sustainable Approaches Elicited to Optimize Non-Functional Requirements*

To make the contribution more practical, we performed an additional step by mapping the sustainability approaches identified in $\mathbf{RQ}_2$ to both the phases of the ML development workflow and the specific NFRs they address. Following the nine-stage ML workflow proposed by Amershi et al. [12], we systematically positioned each sustainability practice based on when participants reported applying it, from model requirements to deployment and monitoring. This mapping was conducted by I and drew on methodologies used in prior studies [143, 187]. We then linked each approach to relevant NFRs using the framework presented in the previous chapter, analyzing both the motivations and intended impacts described by participants. For example, the approach "(A.6) Consider Training with Synthetic Data" was commonly cited as a way to reduce data acquisition costs and improve training efficiency. Based on participant feedback, we mapped this approach to the "Data Collection" and "Model Training" phases and associated it with the NFRs of cost, accuracy, and performance. The results of this analysis are shown in Table 3.7, which shows how sustainability considerations are distributed across all ML workflow stages. The mapping also highlights the diverse range of NFRs addressed. For instance, energy and cost-related approaches include "(A.11) Consider Adopt Cloud Technologies" and "(A.12) Consider Implement Offline Predictions", while socially oriented approaches, such as "(A.4) Consider Ethical Data Management" and "(A.15) Consider Measure Ethical Aspects", focus on fairness and equity. Other approaches, such as consider knowledge distil-

lation (A.17) and consider integrate model explainability (A.19), contribute to explainability and scalability, underscoring the multifaceted nature of sustainability in ML-enabled systems.

Interestingly, several approaches concentrate around the model training phase, including resource-efficient training, pre-trained model reuse, and library selection. In contrast, relatively few approaches target post-deployment phases like monitoring—despite its importance for tracking sustainability over time. Approaches such as "(A.16) track sustainability metrics" hint at this gap, suggesting opportunities for future work to develop tools and methods that support sustainability beyond initial deployment. Overall, this structured mapping offers a practical framework for understanding how sustainability approaches intersect with both ML workflows and NFRs. It supports researchers and practitioners in systematically integrating sustainability from the outset—moving beyond isolated or reactive efforts toward a more holistic, lifecycle-aware approach.

Implication #4. The mapping confirms that sustainability should be integrated across the entire ML lifecycle. While many approaches focus on optimizing resources during model training, post-deployment monitoring remains underexplored. The variety of addressed NFRs further highlights the multifaceted nature of sustainability in ML-enabled systems.

3.4.3 *Breaking the barriers to the adoption of sustainability approaches*

Despite increasing awareness, the findings reveal several persistent barriers hindering the effective adoption of sustainability in ML-enabled systems. These include the absence of standardized metrics, limited organizational support, and inadequate tool support, contributing to inconsistent approaches across ML workflows. A major challenge is the lack of standardized sustainability metrics and evaluation frameworks. Without clear benchmarks, practitioners struggle to measure and compare the impact of sustainability approaches, leading to fragmented adoption. This

reflects broader challenges in software engineering [17], and highlights the need for domain-specific, actionable indicators to guide practice and foster accountability. Limited motivation and awareness also emerged as key barriers. Performance, cost, and time-to-market often take precedence over sustainability, especially in the absence of organizational incentives or regulatory pressure. Consequently, addressing these issues requires both cultural change and structural incentives, such as policies, certifications, or financial benefits for sustainable development. This represents a *call for research actions* aimed at identifying effective mechanisms to embed sustainability into standard ML development approaches. Further studies are needed to explore how sustainability can be quantified, benchmarked, and integrated without compromising system performance or development efficiency. This includes developing practical assessment frameworks, investigating the trade-offs between sustainability and other quality attributes, and proposing automated tools that assist practitioners in making sustainability-aware decisions throughout the ML lifecycle.

In addition to conceptual challenges, practical constraints like inadequate funding and limited tooling hinder adoption. Many sustainability approaches require added computational resources or specialized hardware, increasing development costs. Practitioners also report a lack of integrated tools for sustainability evaluation and monitoring. Research can help by designing cost-effective, lightweight, and automated tools that integrate into existing ML workflows—ideally as open-source frameworks to lower entry barriers. Promising directions include AI-driven optimization to balance sustainability and performance, and economic studies quantifying long-term cost and energy savings. Finally, training and education gaps remain. Few practitioners have formal instruction in sustainable ML, leading to ad hoc or superficial efforts. Expanding curricula and offering industry training on sustainability principles could help foster best practices from early career stages.

Implication #5. Sustainability adoption in ML-enabled systems is limited by the absence of standardized metrics, integration methods, organizational support, and practical tools. Overcoming these barriers requires clear metrics, automated and cost-effective tools, supportive policies, and incentives. Assessing economic trade-offs and expanding education and training are also key to promoting long-term, sustainable approaches in ML.

PART II

EMPIRICAL FOUNDATIONS FOR SUSTAINABLE ML SYSTEM DESIGN

4

EXAMINING THE IMPACT OF BIAS MITIGATION ALGORITHMS ON THE SUSTAINABILITY OF ML-ENABLED SYSTEMS: A BENCHMARK STUDY.

4.1 INTRODUCTION

Prior research has focused extensively on addressing bias and enhancing fairness through the development of bias mitigation algorithms. These algorithms often target protected attributes—demographic characteristics identified by legal, ethical, and societal frameworks to safeguard marginalized groups from systemic discrimination or inequitable treatment [117, 314]. Examples include pre-processing algorithms like FAIR-SMOTE [50], ensemble-based methods such as MAAT [63], and hybrid approaches like FAIRWAY [51]. These studies have primarily targeted the optimization of the trade-off between fairness and accuracy, demonstrating that bias mitigation can improve equity in ML systems while preserving the overall performance of the models.

Despite these advancements, much of the existing literature evaluates fairness as an *isolated* objective, focusing predominantly on metrics like demographic parity or equalized odds. However, even if these practices have been successful in mitigating discrimination, their broader implications, particularly how fairness techniques might influence non-functional attributes such as environmental and economic sustainability, remain unexplored. More specifically, bias mitigation methods may introduce computational overhead during training and deployment, which can increase energy consumption and carbon emissions, thereby contributing to the environmental footprint of ML models. Similarly, these methods may affect economic factors such as training time and storage requirements, potentially impacting the financial feasibility of deploying fairness-enhanced

systems. These trade-offs, while critical to achieving responsible AI, remain poorly understood in the literature.

As highlighted in **Gap #2**, there is still a Lack of multi-dimensional evaluations of sustainable solutions and their trade-offs. This chapter addresses this gap *by framing fairness within the broader theoretical construct of sustainability*, encompassing its social, environmental, and economic dimensions. Building on the conceptualization of sustainability as a stratified and multi-systemic construct [242], I operationalize fairness metrics as indicators of *social sustainability*, connecting them to societal goals such as equity, inclusiveness, and well-being. This framing allows us to evaluate bias mitigation algorithms not merely as tools for addressing fairness concerns but as components of sustainable software systems that balance equity with other non-functional requirements. Beyond fairness, I extend the evaluation of bias mitigation techniques to include two additional sustainability dimensions. On the one hand, I evaluate *environmental sustainability*, quantified through metrics such as energy consumption and carbon emissions. On the other hand, I expand the discussion on the trade-off between fairness and accuracy by exploring other *economic sustainability* measures such as training time and storage weight. Through the integration of these dimensions, I provide a more comprehensive understanding of the trade-offs involved in adopting bias mitigation techniques.

To the best of my knowledge, only a few recent studies have begun advocating for a broader view of fairness in ML, highlighting the importance of considering its trade-offs with other sustainability dimensions [42, 56], for instance by proposing multi-objective algorithms that attempt to optimize fairness alongside efficiency-related or environmental objectives. Compared to these earlier works, the study offers a large-scale, systematic benchmark that quantitatively examines the impact of bias mitigation algorithms on three key dimensions of sustainability—social, environmental, and economic—hence complementing prior efforts with empirical evidence and providing actionable insights into the practical trade-offs involved in deploying fairness-aware ML systems. I focus on analyzing multi-protected attributes, which involves ensuring fairness across multi-

ple overlapping groups [130], as this type of evaluation represents a more realistic scenario for implementing fairness enhancements in practice [64]. More particularly, the work is routed around three main research questions that drive the exploration:

- **RQ$_1$.** *How do bias mitigation algorithms impact social sustainability regarding multiple protected attributes?*
- **RQ$_2$.** *How do bias mitigation algorithms impact environmental sustainability regarding multiple protected attributes?*
- **RQ$_3$.** *How do bias mitigation algorithms impact economic sustainability regarding multiple protected attributes?*

While **RQ$_1$** corroborates and builds upon existing studies by contextualizing fairness within the social sustainability dimension, **RQ$_2$** and **RQ$_3$** extend the current body of knowledge by assessing the environmental and economic implications of bias mitigation techniques. To achieve these objectives, I implemented a **quantitative benchmark study research approach** [318]. A benchmark study systematically evaluates and compares the performance of specific techniques, tools, or algorithms under controlled and reproducible conditions. This approach is particularly suited to the goals, as it enables an objective analysis of multiple metrics across different dimensions of sustainability. By applying six bias mitigation algorithms— *Reweighing* [188], *Meta Fair Classifier*. [49], *Gerry Fair Classifier* [190], *Exponentiated Gradient Reduction* [4], *PrejudiceRemover* [189], and *Grid Search Reduction* [4]—to four datasets—*Adult* [21], *Statlog* [158], *Mep15* [152], and *Compas* [126]—, I conduct a total of 3,360 experimental runs, where I systematically measure the impact of these bias mitigation algorithms on fairness, energy consumption, carbon emissions, accuracy, training time, and storage requirements. These findings reveal multiple insights into the trade-offs across sustainability dimensions. From a social perspective (**RQ$_1$**), I corroborate previous findings in the field, showing that bias mitigation algorithms significantly improve fairness metrics, with Exponentiated Gradient Reduction (EG) emerging

as the most effective technique. However, the results also indicate that certain algorithms, such as Gerry Fair Classifier (GF) and Grid Search Reduction (GS), may deteriorate fairness when applied to multiple protected attributes, emphasizing the complexity of achieving equitable outcomes in diverse scenarios. For environmental sustainability (**RQ$_2$**), the results demonstrate that algorithms like EG and GS consume significantly more energy and produce higher CO2 emissions, raising concerns about their environmental footprint. In contrast, computationally efficient methods like GF and Meta Fair Classifier (MF) exhibit lower environmental impacts, highlighting the trade-offs between fairness improvements and ecological responsibility. From an economic sustainability perspective (**RQ$_3$**), the application of bias mitigation techniques generally increases training time and storage weight while reducing accuracy compared to baseline models. However, exceptions such as Prejudice Remover (PR) demonstrate competitive accuracy in specific datasets, offering valuable insights into the cost-effectiveness of bias mitigation strategies.

To sum up, the **contributions** of this chapter are the following:

1. I frame fairness within the broader theoretical construct of sustainability, moving beyond the traditional fairness-accuracy trade-offs to explore its intersections with social, environmental, and economic dimensions. This research approach not only highlights the multifaceted impacts of bias mitigation algorithms but also bridges the gap between fairness research and sustainability goals, providing a foundation for the development of more responsible and holistic ML system design;

2. These findings reveal complex trade-offs between fairness improvements, energy consumption, carbon emissions, accuracy, training time, and model scalability. These insights not only confirm the need for multi-dimensional evaluation frameworks but also provide practitioners with a clearer understanding of the sustainability implications of their design decisions;

3. I provide a replication package [85] containing all raw data, scripts, datasets, and experimental setups of the quantitative benchmark study. This resource enables researchers to verify the results, extend the study, and explore new methodologies for assessing sustainability in bias mitigation.

4.2 EMPIRICAL STUDY DESIGN

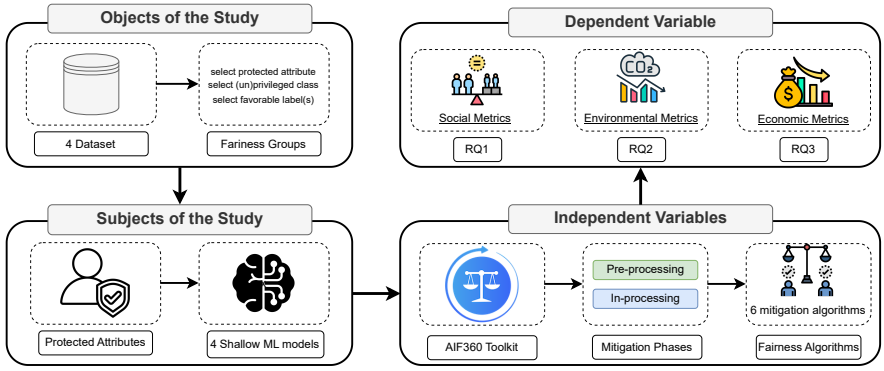


Figure 4.1: Overview of the research process.

To define my research goal, I follow the Goal-Question-Metric (GQM) [39]. The *goal* of the empirical study is to assess the impact of bias mitigation algorithms on the sustainability of ML-enabled systems, with the *purpose* of providing insights into the trade-offs between addressing fairness-related concerns and their corresponding social, environmental, and economic impacts in the *context* of ML-enabled systems. The *perspective* is twofold: practitioners may leverage the outcomes of this study to better evaluate their activities, gaining knowledge on how mitigating risks due to unfairness might influence other critical system properties, potentially enhancing quality assurance processes. Meanwhile, researchers may benefit by exploring the interplay between multiple non-functional requirements of ML-enabled systems, gaining insights into current challenges and informing the design of next-generation quality assurance mechanisms. To achieve my objectives, I formulated three research questions. The first,

RQ$_1$, sought to understand how bias mitigation algorithms impact *social sustainability* through fairness, particularly in scenarios involving multiple protected attributes. Social sustainability, as defined in [242], encompasses properties that influence the individual's physical and psychosocial well-being, making fairness a critical component of this dimension. This analysis serves as both a *preliminary* and *confirmatory* step, designed to corroborate existing findings in fairness literature while framing fairness within the broader theoretical construct of social sustainability. This framing establishes a foundational understanding of fairness that enables the exploration of its interplay with environmental and economic dimensions in subsequent research questions. More particularly, bias mitigation algorithms are explicitly designed to deal with social sustainability properties and, therefore, I expected to observe a positive effect of all the algorithms [50]. Yet, a preliminary investigation may quantify the impact of existing bias mitigation algorithms on social sustainability properties, other than serve as an evidence-based assessment [197] of their performance and potential combinations thereof. At the same time, the preliminary analysis can support the findings observed by Chen et al. [64, 65] in recent investigations into the capabilities of bias mitigation algorithms, possibly discovering additional insights that may advance the current body of knowledge. While the fairness literature has focused on a single attribute at a time [50, 63, 271], the analysis of multiple protected attributes simultaneously needs more exploration [65]. Therefore, I focus all research questions on analyzing multiple protected attributes. Based on the argumentation above, I formulated **RQ$_1$**:

> **RQ$_1$ – Social Sustainability**
>
> *How do bias mitigation algorithms impact social sustainability regarding multiple protected attributes?*

Building on the results of **RQ$_1$**, I turned the attention to environmental sustainability in **RQ$_2$** by investigating the impact of bias mitigation algorithms on metrics such as carbon emissions and energy consumption. These metrics are particularly relevant in the broad context of sustain-

ability, as they quantify the trade-offs between addressing fairness and the ecological footprint of ML-enabled systems. By assessing the environmental impact of these algorithms, I aimed to enhance practitioners' awareness of the energy and resource costs associated with bias mitigation and inspire the development of novel tools to balance fairness and environmental sustainability. Hence, I asked:

> **RQ_2 – Environmental Sustainability**
>
> *How do bias mitigation algorithms impact environmental sustainability regarding multiple protected attributes?*

Finally, **RQ_3** examined the economic sustainability of bias mitigation algorithms, extending the analysis beyond accuracy—a primary focus in traditional studies on fairness. While accuracy remains an important metric, the study incorporates additional dimensions, such as training time and model storage requirements, to provide a more comprehensive perspective. These metrics directly influence operational costs, resource efficiency, and the long-term feasibility of deploying fairness-enhancing algorithms in real-world scenarios. This more comprehensive view may offer insights into the trade-offs between fairness improvements and economic considerations, enabling researchers and practitioners to better evaluate the broader cost-effectiveness of bias mitigation techniques. By moving beyond accuracy, the study highlights the hidden costs associated with fairness-enhancing methods, thereby contributing to the understanding of how these techniques align with economic sustainability objectives. Therefore, I asked:

> **RQ_3 – Economic Sustainability**
>
> *How do bias mitigation algorithms impact economic sustainability regarding multiple protected attributes?*

Figure 4.1 shows an overview of the study design. As shown, I first determined the common ground for the quantitative benchmark study, i.e., the most appropriate datasets, the ML models to experiment with, and

the metrics to estimate social, environmental, and economic sustainability. Afterward, I selected the bias mitigation algorithms to assess, picking those available within the well-known AIF360 toolkit [23]. Finally, I ran the selected bias mitigation algorithms against the benchmark and analyzed the corresponding results to address the research questions. In particular, a benchmark study is a standard tool for the evaluation and comparison of competing systems or components according to specific characteristics, such as performance, dependability, or security [196]. Benchmark studies can be tested repeatedly and quickly without requiring human subjects [334]. Consequently, the results section will present findings that are based solely on quantitative analysis without any subjective interpretations or personal biases, as this type of research approach exploits numerical analysis and statistical techniques rather than descriptive or interpretive methods. However, I complement my findings with extensive discussions in a dedicated section (4.4). In terms of reporting, I employed the guidelines (i) by Wohlin et al. [368] and (ii) by the *ACM/SIGSOFT Empirical Standards*[1].

4.2.1 *Objects of the study*

The *objects* of the study were the datasets to be used for the experimentation. To address the research questions, I required the selection of fairness-relevant datasets—otherwise, I could not observe significant outcomes from the application of bias mitigation algorithms. Table 4.1 describes the main characteristics of the datasets selected for the study. As shown, I identified four popular datasets such as *Adult* [21], *Statlog* [158], *Mep15* [152], and *Compas* [126]. The reason behind this selection is manifold.

First, these datasets have been widely employed in previous literature on software engineering for artificial intelligence [164, 234] and ML fairness [27, 50, 65, 163]: as such, their adoption allowed us to have a common ground to compare the findings with previous achievements in the field, e.g., the four datasets are among those used by Chen et al. [64, 65]. In

[1]Available at: https://github.com/acmsigsoft/EmpiricalStandards. Given the nature of the study and the currently available standards, I followed the *"General Standard"*, and *"Benchmarking"* guidelines.

Table 4.1: Datasets used in the experimentation.

Name	#Sample	#Feature	Protected Attributes	Favorable label	Description
Adult	32,560	104	sex,race	7508 (18,80%)	The goal is to classify individuals into specific annual income classes and determine whether their income is above or below $50,000.
Mep15	15,830	42	sex,race	2718 (17.17%)	The goal is to predict individuals' health care needs based on how Americans pay for medical care, health insurance, and out-of-pocket expenses.
Statlog (German Credit Card)	1,000	61	sex,age	700 (70%)	The purpose is to predict a bank customer's ability to repay, or not, a loan.
COMPAS	7214	8	sex,race	3963 (54,93%)	the goal is to predict, through demographic information and criminal history, whether a defendant will reoffend within two years.

the second place, the selected datasets cover activities that pertain to individuals' personal information in various fairness-sensitive domains, including finance, social, and medical sectors: as a consequence, I could assess the impact of bias mitigation algorithms in various contexts, possibly uncovering patterns that better describe the characteristics of these algorithms. Lastly, this fixed amount of datasets selected allowed us to relax the constraints on the number of investigations to be performed on different objects, hence increasing the number of experiments for each, improving the confidence in driving conclusions on these four datasets. Consistently with prior investigations [64, 65, 271], I chose *'Sex'*, *'Age'* and *'Race'* as protected attributes. As a consequence of this choice, the investigation could be performed on *multiple protected attributes*, which has been noted as a challenging and more realistic condition for the evaluation of any bias mitigation algorithm [65].

4.2.2 *Subjects of the Study*

The *subjects* of the study were the machine learning models to experiment with to assess their sustainability properties. In this respect, I relied on previous literature in the field [28, 50, 335, 345, 386]. Particularly, the selection was informed by the work by Hort et al. [162], who provided a comprehensive survey of ML models frequently employed in fairness research. I selected and trained four different algorithms: *Logistic Regression* (LR), *Linear Support Vector Classification.* (SVC), *Random Forest* (RF), and *XGBoostClassifier* (XGB). According to Hort et al. [162], these are among the most frequently used in the evaluation of bias mitigation methods. In particular,*LR* serves as a reference point for evaluating the relative effectiveness of bias mitigation techniques, being frequently employed as a baseline model due to its simplicity and interoperability; *SVC* is known for its strong classification capabilities and its ability to handle complex decision boundaries; *RF* is recognized for its robustness and resistance to overfitting, making it widely adopted for fairness evaluations across diverse datasets; lastly, *XGB* combines computational efficiency with strong predictive performance, making it a robust method for evaluating fairness and sustainability trade-offs. These models were trained using as features the attributes included in the four datasets described in Section 4.2.1. All of them pertained to binary classification tasks. For each ML algorithm, I defined seven variants, one for each of the independent variables of the study plus one baseline where I did not apply any treatment. In this way, I could experiment with one ML model for each bias mitigation algorithm considered in the study, i.e., each of them applied a different bias mitigation algorithm, against which I compared a baseline model that did not apply any bias mitigation algorithm. Following this design, I defined a total of 28 experimental models. In terms of data preprocessing, I removed missing or invalid values and converted continuous features to categorical ones. In addition, I adapted the preprocessing for equity analysis by identifying protected attributes, defining privileged and non-privileged groups, and clarifying the outcome or favorable label for predictions. I set

sex, *race*, and *age* as privileged groups according to their availability in the selected datasets. As for hyper-parameters configuration, I used the default configurations coming from relevant studies [63–65, 163].

Table 4.2: Variables of the study

Name	Scale	Operationalization
Independent variables:		
Baseline Model	Nominal	Models without any bias mitigation algorithms.
Bias Mitigation Algorithms	Nominal	Reweighing, Meta Fair Classifier, Gerry Fair Classifier, Exponentiated Gradient Reduction, PrejudiceRemover, Grid Search Reduction.
Dependent variables:		
RQ$_1$ Mean Difference	Ratio	$P(Y=1\|D=0) - P(Y=1\|D=1)$
RQ$_1$ Equal Opportunity Difference	Ratio	$TPR_{D=\text{unprivileged}} - TPR_{D=\text{privileged}}$
RQ$_1$ Average Odds Difference	Ratio	$\frac{1}{2}\left(\left(FPR_{D=\text{unprivileged}} - FPR_{D=\text{privileged}}\right) + \left(TPR_{D=\text{unprivileged}} - TPR_{D=\text{privileged}}\right)\right)$
RQ$_2$ Energy consumption	Ratio	Power usage × Training time
RQ$_2$ Carbon Emission	Ratio	Carbon Intensity factor × Energy consumption
RQ$_3$ Training Time	Ratio	$Time_{\text{end}} - Time_{\text{start}}$
RQ$_3$ Accuracy	Ratio	$(TP+TN)/(TP+TN+FP+FN)$
RQ$_3$ F1 score	Ratio	$(2\times TP)/(2\times TP+FP+FN)$
RQ$_3$ Storage Weight	Ratio	The size of the trained model

4.2.3 *Empirical Study Variables*

Once I had defined the context of my study, I proceeded with the definition of the independent and dependent variables of the empirical study. Table 4.2 summarizes the design choices taken in this respect.

Independent Variables. The independent variables of the study were represented by the bias mitigation algorithms that I aimed to assess with respect to their impact on sustainability. I selected techniques that have a potential impact on all evaluation measures relevant to the research questions. In particular, I focused on algorithms that operate *before* or *during*

model training, as they directly influence the training phase and can affect key metrics such as energy consumption, storage weight, and other metrics considered. In contrast, post-processing algorithms, which operate exclusively during the *inference phase*, were excluded as they do not impact the metrics studied in this work. In fact, including post-processing methods would have introduced inconsistencies in the analysis and compromised its comprehensiveness. Therefore, the study is based on state-of-the-art pre-processing and in-processing algorithms [27, 64]. The former (referred to as 'PRE' in the remainder of the section) are methods that attempt to mitigate bias in training data to promote a more equitable model. The latter (referred to as 'IN') are methods to optimize training algorithms to improve fairness. In addition, the study also includes a baseline as an independent variable. More specifically, the study featured the following pre-processing and in-processing algorithms:

- *Baseline.* A baseline where no bias mitigation algorithms were applied.

- *PRE#1 - Reweighing* [188]. This allows the weight attributed to individual instances of the dataset to be changed, identifying instances belonging to protected and unprotected groups looking for ideal weights aimed at mitigating any type of discrimination present within the dataset.

- *IN#1 - Meta Fair Classifier*. [49]. It represents a classifier exploited in the in-processing phase to modify the standard training set into a training set which considers fairness metrics. Once the training set is modified, additional standard models are trained on the new modified training set to provide a training phase that primarily uses fairer training sets.

- *IN#2 - Gerry Fair Classifier* [190]. This is an algorithm for learning classifiers that are fair with respect to rich subgroups. The rich subgroups are defined by linear functions on the protected attributes, and the notions of fairness are statistical in nature.

- *IN#3 - Exponentiated Gradient Reduction* [4]. The algorithm is able to reduce fair classification to a sequence of cost-sensitive classification problems, returning a randomized classifier with the lowest empirical error subject to fair classification constraints.

- *IN#4 - PrejudiceRemover* [189]. The algorithm adds a term of discrimination-aware regularization to the learning objective.

- *IN#5 - Grid Search Reduction* [4]. It reduces fair classification to a sequence of cost-sensitive classification problems, returning the deterministic classifier with the lowest empirical error subject to fair classification constraints among the candidates sought.

The selection of this specific set of algorithms was motivated by two main factors. First, I prioritized algorithms with robust, standardized, and optimized implementations: I therefore selected the algorithms available in the AI FAIRNESS 360 (AIF360) toolkit [23]. This choice ensured methodological rigor: while I am aware of the existence of other bias mitigation algorithms, relying on unstandardized or non-optimized fairness libraries might have introduced inconsistencies and inaccuracies in the analysis, particularly for environmental metrics that are highly sensitive to implementation efficiency. For example, less efficient implementations might lead to inflated resource consumption, longer execution times, and higher energy usage, thereby skewing the results and compromising the validity of the conclusions. By leveraging AIF360, which has been validated in both academic research and industry practice [88, 209], I minimized such risks and ensured the reliability and reproducibility of the findings.

Second, the selected algorithms were required to handle *multiple protected attributes*. In this respect, the selected algorithms, as for their implementation, allowed for the specification of more than one protected attribute in the form of "a subset of features for which fairness is desired" as specified in the AIF360 documentation.[2] More specifically, the chosen algorithms operate on AIF360 objects called `StandardDataset`, which

[2]AIF360 documentation: https://aif360.readthedocs.io/en/stable/.

include a parameter titled `protected_attribute_names(list)`, implying that more than an attribute can be indicated as protected. In cases where an algorithm requires a single attribute, this is explicitly stated—for example, the *DisparateImpactRemover* algorithm does not operate on a `StandardDataset` but instead includes the specific parameter `sensitive_attribute (str)`, which is the reason why I did not include it in the experiments. This selection ensures the flexibility and applicability of the chosen methods for scenarios involving multiple protected attributes, as argued in other studies on the matter, e.g., the recent study by Chen et al. [65].

Dependent Variables. The dependent variables of the study were defined based on the specific perspectives targeted, i.e., social, environmental, and economic sustainability, and based on the currently available metrics to measure each of them.

RQ$_1$. *social sustainability metrics.* Table 4.2 overviews the social sustainability metrics employed when addressing **RQ$_1$**, which are known as *'mean difference'* (alias of statistical parity difference), *'average odds difference'*, and *'equal opportunity difference'*. These metrics are recognized as standard instruments to measure the extent to which a ML model produces fair outcomes [106, 107, 263], providing insights into potential discriminatory patterns within the data that significantly influence the model's knowledge and functionality. Let D be the protected attribute, where 1 indicates the privileged group and 0 indicates the unprivileged group. Let Y be the actual label, with 1 denoting the favorable class and 0 being the unfavorable class. Based on these definitions, the considered metrics operate as follows:

- *'mean difference'* [38] calculates the disparity in favorable rates ($Y = 1$) between the privileged ($D = 1$) and unprivileged ($D = 0$) groups.
- *'equal opportunity difference'* measures the maximum difference between privileged ($D = 1$) and unprivileged ($D = 0$) subgroups in true positive rates (TPR).

- *'average odds difference'* is a measure that averages the differences between false positive rates (FPR) and TPR between privileged ($D = 1$) and non-privileged ($D = 0$) groups. It combines FPR and TPR to assess the overall performance of a model across different groups.

The metrics used in this context measure fairness between privileged and unprivileged groups. A metric value of zero signifies equal distribution and fairness, so the groups are fair. A value other than zero indicates a disparity in the benefit received by the non-privileged group or the privileged group. A value of 1 indicates a strong bias in favor of the privileged group. Otherwise, a value of -1 indicates a strong bias in favor of the non-privileged group. These metrics were computed through AI FAIRNESS 360 [23].

RQ$_2$. *environmental sustainability metrics.* When addressing **RQ$_2$**, I estimated the environmental sustainability of ML models by quantifying the *'energy consumption'* (measured in Joules) and *'CO2 emissions'* (measured in g/CO2) of the model at training time. These metrics are inherently hardware-dependent and quantitatively represent the resources required to build a model. To quantify energy consumption and carbon emissions, I exploited CODECARBON,[3] a library that leverages two widely used energy measurement tools: *RAPL* and *Nvidia pynvml* [60, 114, 291, 375]. *RAPL* is employed to calculate the resources utilized by the CPU and RAM, while *pynvml* is employed for GPU computations. CODECARBON provides implementations and patterns, with its central entity, i.e., `TrackerEmission`, capturing critical data on energy consumption and resources. These pieces of information are then employed by CODECARBON to generate a conclusive report on environmental sustainability in an emissions.csv file. While *'CO2 emissions'* is indeed positively correlated with *'energy consumption'*, I argue that both metrics are necessary because they provide different insights. Energy consumption captures the amount of energy required. At the same time, carbon emissions offer a more complete picture by considering the environmental impact,

[3]The CODECARBON toolkit: https://codecarbon.io/.

which depends on the carbon intensity of the energy source. By including both, I preserve critical information regarding emissions, which is vital for practitioners aiming to reduce not just energy use but also the associated carbon footprint. Moreover, since the tool I utilized computes both metrics, I opted to retain them for transparency and practical relevance to potential users, who may value the distinct insights provided by each. Table 4.2 describes how CODECARBON evaluates these metrics.

RQ$_3$. *economic sustainability metrics.* In **RQ$_3$**, I first estimated economic sustainability by assessing the *prediction quality* of the model, considering well-known metrics such as *'Accuracy'* and *'F1 score'*. The rationale behind the selection of these metrics as a proxy for economic sustainability lies in the observation that higher values in these metrics indicate better efficiency and reliability, contributing to economic sustainability by reducing costs, saving time, enhancing customer satisfaction, providing a competitive edge, and aligning with business goals [36, 234]. In addition, I also computed *'training time'* (measured in seconds) and *'storage weight'* (measured in KB). These two metrics have a direct impact on resource consumption, operational costs, and overall efficiency, hence influencing the economic impact of ML models. Following the methodology used in prior studies, I computed the macro-averaged values for F1 score to facilitate a balanced performance comparison across favorable and unfavorable classes. This involves averaging both classes' F1 score results. A higher score in this metric implies improved ML accuracy [64, 65].

4.2.4 *Experimental Hypotheses, Execution, and Analysis*

After defining the experimental subjects and objects, I designed the working hypotheses, which enabled the execution of the experiments and the subsequent data analysis. This section reports on these aspects, detailing the rationale and research methods employed to address the objectives of the study.

Experimental Hypotheses. I aimed at analyzing how the independent variables, namely the bias mitigation algorithms, affect the dependent variables, including social, environmental, and economic sustainability metrics. As such, I defined the following experimental elements.

Let μ_{B_i} and μ_{B_j} be the ML models built using a independent variables B_i and B_j, respectively, where $B_i, B_j \in$ {Baseline, Reweighing, Meta Fair Classifier,...}; let S be the set of sustainability metrics considered in the study. As for $\mathbf{RQ}_1$, let S_{so} be a social sustainability metric in the set of sustainability metrics considered in the study. the null hypothesis was the following:

$$H_0^{B_i,B_j,S_{so}} : \mu_{B_i}^{S_{so}} = \mu_{B_j}^{S_{so}} \; \forall i \neq j$$
$$S_{so} \in \left\{ \begin{matrix} \text{mean difference, average odds difference,} \\ \text{equal opportunity difference} \end{matrix} \right\}$$

The null hypothesis $H_0^{B_i,B_j,S_{so}}$ determines the effect of the chosen algorithms on the dependent variable S_{so}. Furthermore, μ_{B_i} and μ_{B_j} represents the average measurement result of variable S_{so}. This leads to the following alternative hypothesis, stating that for each dependent variable S_{so}, a statistically relevant difference can be observed between independent variables:

$$H_a^{B_i,B_j,S_{so}} : \mu_{B_i}^{S_{so}} \neq \mu_{B_j}^{S_{so}} \; \forall i \neq j$$
$$S_{so} \in \left\{ \begin{matrix} \text{mean difference, average odds difference,} \\ \text{equal opportunity difference} \end{matrix} \right\}$$

As for $\mathbf{RQ}_2$, I defined a null and alternative hypothesis for each dependent variable related to environmental sustainability metrics S_{en}:

$$H_0^{B_i,B_j,S_{en}} : \mu_{B_i}^{S_{en}} = \mu_{B_j}^{S_{en}} \; \forall i \neq j$$
$$S_{en} \in \{\text{energy consumption, carbon emission}\}$$

The corresponding alternative hypothesis was formulated as follows:

$$H_a^{B_i,B_j,S_{en}} : \mu_{B_i}^{S_{en}} \neq \mu_{B_j}^{S_{en}} \; \forall i \neq j$$

$$S_{en} \in \{\text{energy consumption, carbon emission}\}$$

As for **RQ**$_3$, I described the null and alternative hypotheses for each dependent variable related to economic sustainability metrics S_{ec}:

$$H_0^{B_i,B_j,S_{ec}} : \mu_{B_i}^{S_{ec}} = \mu_{B_j}^{S_{ec}} \; \forall i \neq j$$

$$S_{ec} \in \{\text{accuracy, f1 score, training time, storage weight}\}$$

The corresponding alternative hypothesis was formulated as follows:

$$H_a^{B_i,B_j,S_{ec}} : \mu_{B_i}^{S_{ec}} \neq \mu_{B_j}^{S_{ec}} \; \forall i \neq j$$

$$S_{ec} \in \{\text{accuracy, f1 score, training time, storage weight}\}$$

It is important to note that the experiment was balanced with respect to its factor, as each treatment contains unique mitigation algorithms, each of which belonged to the same performance level represented by the treatment. **Experiment Execution.** I designed the procedure depicted in Figure

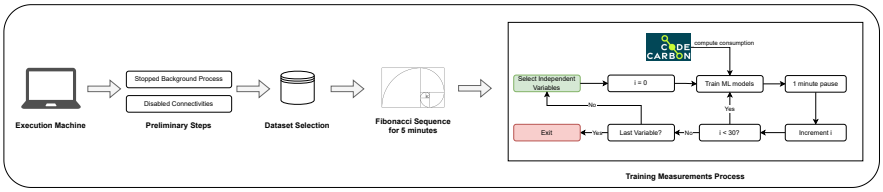


Figure 4.2: Execution process.

4.2. Specifically, I conducted a benchmark study, which is defined as *"a standard tool for the competitive evaluation and comparison of systems or components based on specific characteristics such as performance, dependability, or security"* [196]. One of the primary advantages of benchmark studies is their capacity for repeated and rapid testing without relying

on human subjects [319]. This characteristic aligns with the quantitative nature of the research, where results are derived entirely through objective, numerical analysis and statistical techniques, eliminating subjective interpretations or personal biases. By grounding the methodology in objectivity, the findings presented in the results section stem directly from the adopted approach, adhering to established standards for benchmark studies. I conducted experiments involving ML models: the four classifiers were trained using each of the independent variables of the study, i.e., a baseline model that did not include any bias mitigation algorithm plus the models trained using the six bias mitigation algorithms employed. A 70%-30% training-test ratio was applied to assess the sustainability implications of each model. The experiments were performed on a machine running Ubuntu Linux, equipped with an AMD Ryzen 7 5800H CPU, an RTX 3060 GPU, and 16 GB of RAM. I configured the environment to support `Python 3.6`, `AIF 0.5.0`, `scikit-learn 1.2.2`, `scikit-posthocs 0.9.0`, and `xgboost 2.0.2` libraries.

Further clarifications on the execution are worth discussing to understand all the aspects included in this experimentation. First, I included 4 ML models and 7 bias mitigation techniques in the study. Evaluating each bias mitigation technique for the 4 ML models, I collected a total of 28 experimental models. Therefore, these experimental models are then evaluated considering 4 datasets, resulting in 112 experimentation objects. Finally, every single experimentation object is repeated 30 times to account for problems related to potential non-determinism caused by the hardware configuration. Therefore, I obtained a total of 3,360 experiments. This amount of re-executions was required to account for the potential non-determinism of the measurements caused by the hardware/software configuration of the experimental machine.

In addition, as energy consumption is strongly influenced by hardware temperature, I followed existing guidelines [75] and executed a five-minute Fibonacci sequence before each measurement with the aim of warming the CPU up. Before running the experiments, I stopped all the unnecessary background processes to let the machine reach a stable condition [114].

Then, I ran the models to get their energy consumption and other metrics written in a .csv file with CodeCarbon. I introduced a one-minute pause after each benchmark execution to minimize the impact of CPU/GPU warm-up and overall system acceleration.

I first trained ML models without bias algorithms to create a first benchmark baseline and then applied each treatment described in Section 4.2.3. After training the models, I collected measures on the test set to address my research questions. Finally, I stored ML models in `.pickle` files for further analysis.

Data Analysis. As a final step of my research method, I statistically analyzed the data coming from the 3,360 experiments. In this respect, I first independently considered the experiments executed on each of the datasets in my study. As such, I had 840 distributions to consider at the time: these provided information on the sustainability metrics computed during the 30 executions of the 28 ML models on a given dataset. The raw data produced are in my online appendix (see *"Measurements"* folder) [85].

I statistically compared each pair of models against all the sustainability metrics in the study by employing (1) quantile-quantile (Q-Q) plots; and (2) the Shapiro-Wilk normality test, with a significance level of $\alpha = 0.05$ [308]. These methods allowed us to determine whether my metric data adhered to a normal distribution. Upon inspecting the Q-Q plots and the Shapiro-Wilk test results, it became evident that not all models exhibited normal distributions.

After identifying the non-normal distribution, I applied the Friedman test [322], a non-parametric statistical test used to detect treatment differences across multiple algorithms for each metric and dataset. However, Friedman's test does not indicate which groups are significantly different from each other; for this reason, if the Friedman test result was significant, with α below = 0.05, indicating that there are differences among the solutions, I proceeded to conduct the Nemenyi post-hoc test [159, 254]. Nemenyi's post-hoc test is a multiple comparison test that is used to identify which pairs of groups are significantly different from each other. From the Nemenyi results, the Holm-Bonferroni correction [160], a gen-

eral method for controlling the family-wise error rate (FWER) in multiple comparisons, was applied. This correction can be applied regardless of whether the statistical tests are parametric or non-parametric. It adjusts the significance level α to account for the number of comparisons made, reducing the risk of Type I errors (false positives) in multiple comparisons, even in non-parametric settings like those involving the Nemenyi test.

To evaluate which specific solutions differed and interpret the post-hoc results, I used a representation based on Critical Difference (CD) diagrams [87]. These diagrams rank multiple groups along the x-axis, connecting statistically indistinguishable groups with horizontal crossbars. The designed plots retain the statistical rigor of the original CD diagrams while improving clarity. They highlight groups of methods that do not show statistically significant differences for each metric and emphasize the average performance of each algorithm. To make the results more accessible, I introduced visual cues, such as distinct colors, to indicate whether methods performed better, worse, or were statistically equivalent to the baseline. Additionally, algorithms that did not exhibit significant differences are presented in separate rows, further enhancing clarity.

4.3 ANALYSIS OF THE RESULTS

This section reports the quantitative insights from the data extraction and analysis phase. In particular, I address the study-specific research questions by visualizing and discussing the results of the experiments and statistical analyses. For each **RQ** and metric analyzed, I display a set of diagrams reporting a comparison of the application of each bias mitigation algorithm on each dataset through critical distance-like plots in the cases where the significance of the Friedman test is less than 0.05. The plots that depict the results organize algorithms into rows, with each row representing a group of methods that do not show statistically significant differences in performance for a given metric. To highlight the performance of each bias mitigation algorithm relative to the baseline, the algorithms are depicted in distinct colors. These colors indicate whether the algorithms

performed better (•), worse (•), or were statistically equivalent (•) to the baseline. The dotted line represents the best possible value for each metric. Additionally, the plot includes a caption summarizing the average metric values for each algorithm, ranking them from best to worst.

4.3.1 RQ_1 - *How do bias mitigation algorithms impact social sustainability regarding multiple protected attributes?*

The goal of this research question was to understand the impact of bias mitigation algorithms considering multiple protected attributes. To evaluate this aspect, I use mean difference, equal opportunity, and average odds difference. These metrics, used to assess fairness between privileged and unprivileged groups, are detailed in Section 4.2.3,

4.3.1.1 *Mean Difference Evaluation*

Figure 4.3 shows a series of plots of the `mean difference` for the datasets described in section 4.2.1. Adult, Mep, and Statlog datasets exhibit negative values, indicating a predisposition favoring the privileged group across all independent variables. Conversely, the Compas dataset's independent variables span negative and positive values, delineating a spectrum from privileged to unprivileged groups for all the algorithms.

Within the Adult dataset, the analysis reveals a pronounced bias with the Baseline (-0.22) compared to other algorithms, with the most favorable algorithm identified as MF (-0.0055). The groups {GF, GS, RW} and {EG, MF} demonstrate no significant variance. However, a notable deviation exists between all algorithms in this dataset and both the Baseline and the PR algorithm, underscoring a substantial differential impact.

Regarding the Mep dataset, the GS algorithm is the least effective (-0.25), underperforming relative to the Baseline, while EG stands out as the most efficacious algorithm (-0.0006). Statistical analysis batches the algorithms into groups {GS, PR}, {GS, RW}, {RW, GF}, and {GF, MF}, within which no significant differences are observed. Exceptionally, EG distinguishes itself by showing statistically significant variation from its counterparts.The

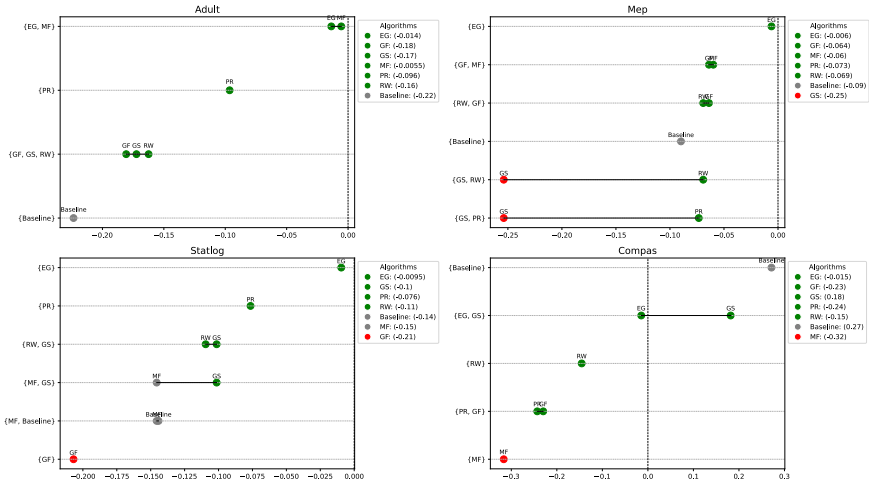


Figure 4.3: Impact of Fairness Algorithms on *Mean Difference* with different datasets. The *algorithms* are Reweighing (RW), Meta Fair Classifier (MF), Gerry Fair Classifier (GF), Exponentiated Gradient Reduction (EG), PrejudiceRemover (PR), Grid Search Reduction (GS). A green dot • indicates a statistically significant improvement against the baseline, a red dot • indicates a deterioration instead. Finally, a gray dot • indicates that there is no statistically significant difference with the baseline.

baseline shows significant variation from other algorithms. In the Statlog dataset, algorithm values within the groups {MF, Baseline}, {MF, GS}, and {RW, GS} appear homogenous, with no significant disparities. In contrast, GF, PR, and EG algorithms are statistically different, with GF presenting as the least preferable option (-0.21) and EG approximating the most equity (-0.0095).

Finally, the Compas dataset presents a broad range of values (-0.32 to 0.27), with the EG algorithm nearing 0 (-0.015). The algorithm groups {EG, GS} and {PR, GF} exhibit internal consistency, showing no significant variance. Yet, the Baseline, RW, and MF algorithms stand apart, demonstrating statistically significant differences from their counterparts, highlighting the nuanced performance across different evaluation metrics.

4.3.1.2 *Equal Opportunity Difference Evaluation*

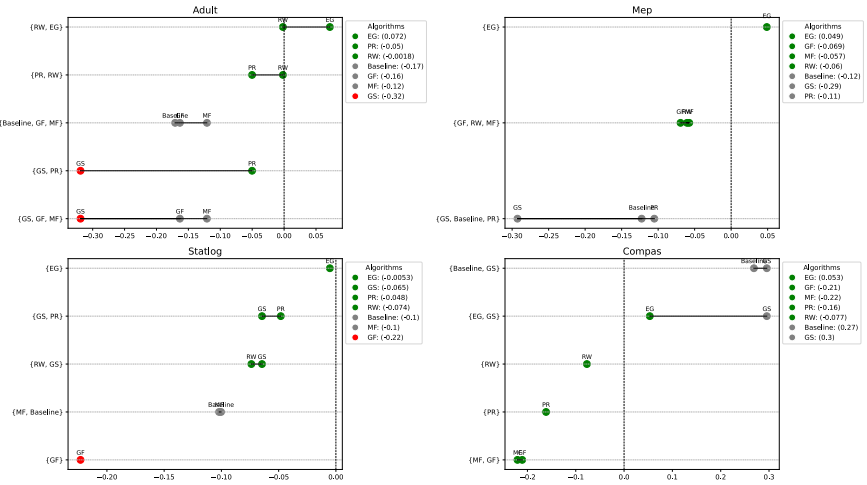


Figure 4.4: Impact of Fairness Algorithms on *Equal Opportunity Difference* with different datasets. The *algorithms* are Reweighing (RW), Meta Fair Classifier (MF), Gerry Fair Classifier (GF), Exponentiated Gradient Reduction (EG), PrejudiceRemover (PR), Grid Search Reduction (GS). A green dot • indicates a statistically significant improvement against the baseline, a red dot • indicates a deterioration instead. Finally, a gray dot • indicates that there is no statistically significant difference with the baseline.

For the `equal opportunity difference` metric, Figure 4.4 shows a series of critical distance diagrams highlighting how the algorithms differ. The datasets Adult, Mep, and Compas display a range of values indicative of biases toward both privileged and unprivileged groups across independent variables, while the Statlog dataset exhibits a bias favoring only the privileged group.

In the Adult dataset, the GS algorithm is the worst, decreasing the Baseline value from -0.17 to -0.32. The RW algorithm shows a slight advantage (0.0018), though this advantage is not significant when compared to EG (-0.072). The groups {Baseline, GF, MF}, {GS, PR}, {GS, GF, MF}, {PR, RW}, and {RW, EG}, do not show significant statistical variations. Each algorithm shares a statistical equivalence with at least one other in this scenario.

Regarding the Mep dataset, the GS algorithm is the least effective (-0.29), underperforming relative to the Baseline (-0.12), while EG stands out as the most efficacious algorithm (0.049). The algorithm groups {GS, Baseline, PR} and {GF, RW, MF} exhibit no discernible statistical differentiation. However, EG deviates, showing a statistically significant difference from other algorithms.

In Statlog, the algorithm values within the groups {MF, Baseline}, {RW, GS}, and {GS, PR} are statistically analogous, presenting no significant variance. In stark contrast, the GF and EG algorithms demonstrate significant statistical departures from the rest, with GF presenting as the least advantageous option (-0.22) and EG approximating the most equity (-0.0053).

Finally, for the Compas dataset, the algorithms GS (-0.22) and MF (-0.21) change the Baseline (0.27) from unprivileged to privileged values. For this dataset, EG is the best algorithm (0.053). The groups {MF, GF}, {EG, GS}, and {Baseline, GS} show no significant variance. However, PR and RW showed statistically significant differences from the others.

4.3.1.3 *Average Odds Difference Evaluation*

Figure 4.5 shows the last social metric analyzed, the `average odds difference`. The independent variables in all the datasets exhibit different values of biases toward both privileged and unprivileged groups. Concerning the Adult dataset, the groups {GS, GF, MF}, {Baseline, GF}, {GS, PR}, and {PR, RW} show no significant variance. However, EG is significantly different from other algorithms. The worst algorithm is GS (-0.19), and the best is RW (-0.022), even if not significantly different from PR (-0.043). Even in the Mep dataset, GS (-0.26) is the worst algorithm, deteriorating the Baseline (-0.081). The best algorithm in this case is EG (0.034), which is also significantly different from the other algorithms. The algorithm groups {GS, Baseline, PR} and {GF, RW, MF} are not significantly different.

In Statlog, the GF (-0.12) is the worst algorithm, deteriorating the Baseline value (-0.093). Additionally, the groups {GF, MF, Baseline}, {MF, Baseline, GS}, and {GS, RW} are not significantly different. PR (-0.0046), who

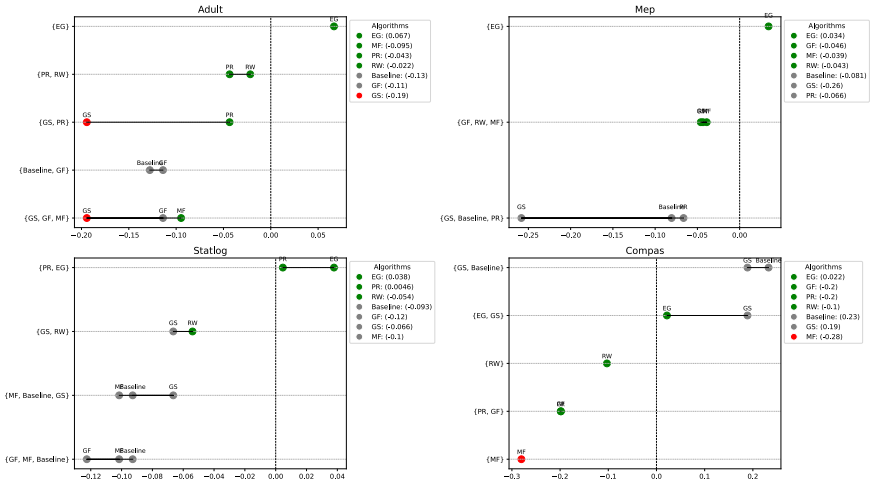


Figure 4.5: Impact of Fairness Algorithms on *Average Odds Difference* with different datasets. The *algorithms* are Reweighing (RW), Meta Fair Classifier (MF), Gerry Fair Classifier (GF), Exponentiated Gradient Reduction (EG), PrejudiceRemover (PR), Grid Search Reduction (GS). A green dot • indicates a statistically significant improvement against the baseline, a red dot • indicates a deterioration instead. Finally, a gray dot • indicates that there is no statistically significant difference with the baseline.

contributes to the group {PR, EG}, is the best algorithm in this case, and no algorithms are statistically significant compared to all others. For the Compas dataset, MF and RW have a statistically significant difference with respect to all algorithms. The baseline (0.23) has values trending toward the unprivileged group, and all algorithms reduce this value, even worsening as MF (-0.28) turns out to be the worst. The best algorithm is EG (0.022), suggesting that there is no statistically significant difference with GS (0.19). The groups {EG, GS} and {GS, Baseline} are not significantly different. Also, {PR, GF} show no statistically significant differences.

The analysis of the fairness metrics *mean difference, equal opportunity, and average odds difference* showed that the algorithms used have an impact and are not statistically significant to each other. Therefore, the null hypothesis is rejected $H_0^{B_i,B_j,S_{so}}$.

RQ_1 — Social Sustainability

*The findings from the **RQ_1** indicate that models not utilizing bias mitigation algorithms are more susceptible to bias. Conversely, applying algorithms designed to mitigate bias across various protected attributes generally enhances social metrics, particularly with the Exponentiated Gradient Reduction algorithm showing notable improvements. However, the Gerry Fair Classifier and Grid Search Reduction algorithms exhibit a deterioration in the metrics across datasets when I consider multiple protected attributes. Therefore, the results show that there is a significant difference with at least one of the bias mitigation algorithms for each dataset.*

4.3.2 ***RQ_2*** *- How do bias mitigation algorithms impact environmental sustainability regarding multiple protected attributes?*

To answer the second research question (**RQ2**), I analyzed the impact of bias mitigation algorithms on two environmental sustainability metrics, that are energy consumption and carbon emission, both described in Section 4.2.3. Through this detailed analysis, I provide an incisive perspective on the often overlooked environmental aspect of bias mitigation algorithms. To enhance readability, I have incorporated diagrams illustrating all the algorithms for energy consumption and carbon emission metrics. For additional diagrams and raw materials, please refer to the online appendix [85].

4.3.2.1 *Energy Consumption Evaluation*

I first analyze the `energy consumption` of the independent variables in each dataset. Figure 4.6 shows the critical difference plot for the results, which compares the energy consumption of seven independent variables with four datasets. The EG and GS algorithms are the worst for all datasets, consuming much more energy to remove bias from the models than the Baseline. Additionally, the EG and GS algorithms are statistically different.

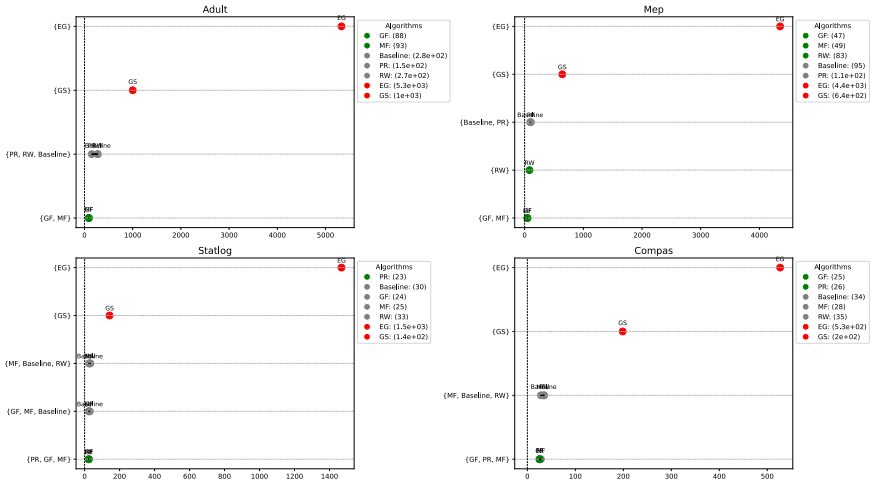


Figure 4.6: Impact of Fairness Algorithms on *Energy Consumption (Joule)* with different datasets. The *algorithms* are Reweighing (RW), Meta Fair Classifier (MF), Gerry Fair Classifier (GF), Exponentiated Gradient Reduction (EG), PrejudiceRemover (PR), Grid Search Reduction (GS). A green dot • indicates a statistically significant improvement against the baseline, a red dot • indicates a deterioration instead. Finally, a gray dot • indicates that there is no statistically significant difference with the baseline.

In the Adult dataset, the algorithm group {GF, MF} and {PR, RW, Baseline} suggest no statistically significant difference. On the one hand, the algorithms GF (88 Joule) and MF (93 Joule) required less energy to train the models compared with the Baseline (280 Joule). On the other hand, GS and EG required, respectively, over 1000 and 5000 Joule to mitigate the bias. In the same way, for the Mep dataset, the same algorithm groups {GF, MF} and {PR, Baseline} suggest that there is no statistically significant difference with GF (47 Joule) and MF (49 Joule), requiring less energy than the Baseline (95 Joule). Even in this case, GS and EG required more energy than the Baseline, respectively, over 640 and 4400 Joule. For the Statlog dataset, the algorithm groups {PR, GF, MF}, {MF, Baseline, RW} and {GF, MF, Baseline} are not significantly different, with PR (23 Joule) and GF (24 Joule) having the lowest values and EG and GS the highest.

In the same way, for the Compas dataset, the algorithm groups {GF, PR, MF} and {MF, Baseline, RW} are not significantly different; the algorithms GF (25 Joule) and PR (26 Joule) required less energy than the Baseline (34 Joule). Even in this case, the GS and EG algorithms use more energy than the Baseline, respectively, over 200 and 530 Joule.

4.3.2.2 *Carbon Emission Evaluation*

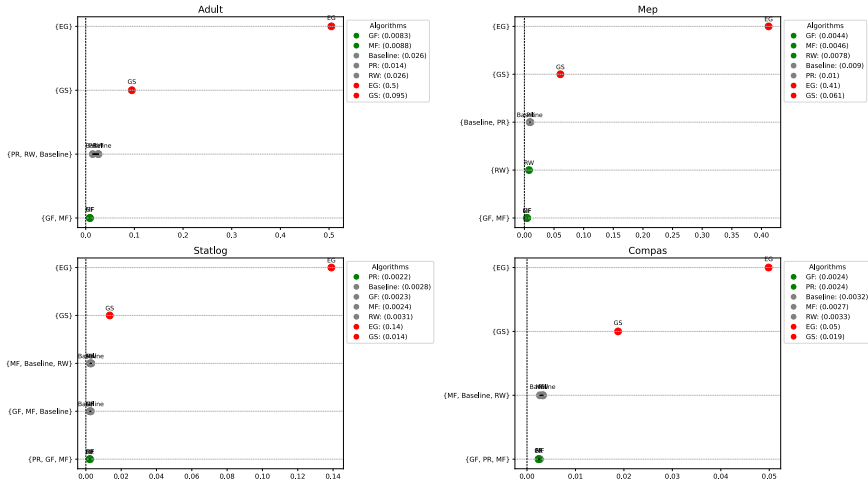


Figure 4.7: Impact of Fairness Algorithms on *Carbon Emission (g/CO2)* with different datasets. The *algorithms* are Reweighing (RW), Meta Fair Classifier (MF), Gerry Fair Classifier (GF), Exponentiated Gradient Reduction (EG), PrejudiceRemover (PR), Grid Search Reduction (GS). A green dot • indicates a statistically significant improvement against the baseline, a red dot • indicates a deterioration instead. Finally, a gray dot • indicates that there is no statistically significant difference with the baseline.

Figure 4.7 shows a visual comparison of the environmental impact of several fairness algorithms, as measured by `carbon emission` in grams of CO2, across four datasets. The EG algorithm appears to produce the most carbon emissions for all datasets, reaching 0.5 grams of CO2 for the Adult dataset and 0.41 grams for the MEP dataset, highlighting its substantial environmental footprint.

Among the algorithms, EG and GS demonstrate statistically significant differences in carbon emissions across the datasets. For the Adult dataset, algorithms in the group {GF, MF} showed no statistical difference from each other, and the GF (0.0083 grams) and MF (0.0088 grams) algorithms emit significantly lower carbon dioxide than the Baseline (0.026 grams). Conversely, the PR algorithm (0.014 grams) cuts emissions compared to both the Baseline (0.026 grams) and RW (0.026 grams), hence leading to algorithms in the group {PR, RW, Baseline} having no statistical distinction between them. Similarly, two algorithm groups in the MEP dataset showed no statistical difference between their components: {GF, MF} and {Baseline PR}. Particularly, GF (0.0044 grams) and MF (0.0046 grams) reduce emissions compared to the baseline. RW's emissions (0.0078 grams) are also lower than those of the Baseline (0.009 grams) and PR (0.01 grams), and these last two are statistically indistinguishable.

In the Statlog dataset, both PR (0.0022 grams), GF (0.0023 grams), and MF (0.0024 grams) show significantly lower emissions, resulting in no statistical difference within the group {PR, GF, MF}. The MF algorithm also undercuts the Baseline (0.0028 grams) and RW (0.0031 grams) in emissions without marked statistical separation. In addition, the algorithms GF, MF, and Baseline resulted as non-statistically different, hence resulting in two other groups: {GF, MF, Baseline} and {MF, Baseline, RW}. Finally, in the Compas dataset, I only have two groups of algorithms within which there is no statistical difference: {GF, PR, MF} and {MF, RW, Baseline}. GF and PR each emit (0.0024 grams) of CO2, lower than the Baseline (0.0032 grams), alongside MF with (0.0027 grams) of emissions. Also, the second group, comprising MF (0.0027 grams) and RW (0.0033 grams), exhibits no significant differences in carbon emissions from the baseline. The analysis of the environmental metrics *carbon emission and energy consumption* showed that the algorithms used have an impact and are not statistically significant to each other. Therefore, the null hypothesis is rejected $H_0^{B_i,B_j,S_{en}}$.

RQ_2 — Environmental Sustainability

> *The analysis of **RQ**$_2$ reveals the profound impact of the chosen algorithms on both CO2 emissions and energy consumption during the model production phase. Interestingly, the GF algorithm is the most energy-efficient, boasting the lowest emission rates. On the other hand, the EG and GS algorithms present a sharp contrast, increasing energy consumption and, as a result, significantly exceeding the emission levels of the baseline model. In comparison, the EG and GS algorithms increase energy use, leading to higher emissions than the Baseline models. In contrast, the MF and PR algorithms were particularly efficient, requiring less energy for model training. In this case, the results show a significant difference with at least two bias mitigation algorithms for each dataset.*

4.3.3 **RQ**$_3$ - *How do bias mitigation algorithms impact economic sustainability regarding multiple protected attributes?*

In the third research question (**RQ3**), I aim to explore the effects of bias mitigation algorithms on economic sustainability. I consider the models' accuracy, F1 score, training time, and storage weight to assess such aspects. These metrics are described in Section 4.2.3. Due to readability issues, I have incorporated diagrams illustrating all the algorithms for training time and storage weight metrics; additional diagrams and raw materials are in the online appendix [85]

4.3.3.1 *Accuracy Evaluation*

The analysis begins with examining `Accuracy` as depicted in Figure 4.8. Starting with the Adult dataset, I observe a decline in accuracy metrics across all algorithms when compared to the Baseline. Notably, RW's accuracy dips slightly to (85%) from the Baseline's (86%), a difference that is not statistically significant. However, MF's accuracy significantly drops to (39%), indicating a substantial deviation. The algorithms in the group {EG, PR, GS} maintain an accuracy of 80% without significant statistical

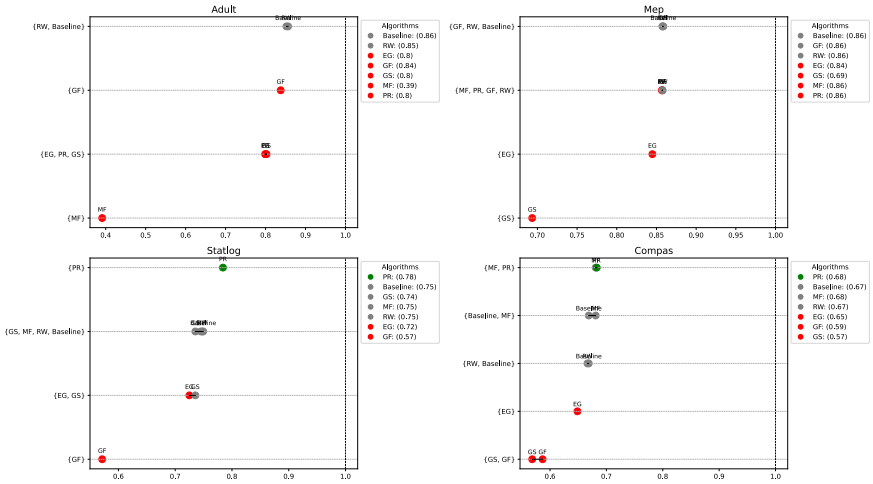


Figure 4.8: Impact of Fairness Algorithms on *Accuracy* with different datasets. The *algorithms* are Reweighing (RW), Meta Fair Classifier (MF), Gerry Fair Classifier (GF), Exponentiated Gradient Reduction (EG), PrejudiceRemover (PR), Grid Search Reduction (GS). A green dot • indicates a statistically significant improvement against the baseline, a red dot • indicates a deterioration instead. Finally, a gray dot • indicates that there is no statistically significant difference with the baseline.

divergence. In Mep, the Baseline (86%) has the same accuracy as the others in the group {GF, RW, Baseline} and is not statistically different between them. Additionally, the group {MF, PR, GF, RW} is not statistically different. In this case, GS is the worst algorithm (69%) followed by EG (84%), with GS and EG statistically different. For the Statlog dataset, the PR algorithm shows an accuracy of (78%) from the Baseline's (75%), standing out as statistically significant. The groups {GS, MF, RW, Baseline} and {EG, GS} reveal no significant accuracy differences since they all hover around (75%) accuracy. Conversely, GF shows (57%) accuracy, underperforming significantly with respect to the others. In the Compas dataset, both PR and MF algorithms match in accuracy at (68%), outperforming the Baseline's (67%). The groups {RW, Baseline}, {Baseline, MF} and {GS, GF} are not statistically different. Here, GS and GF are the least effective, with accuracy of (57%) and (59%), respectively. EG is statistically different to the others.

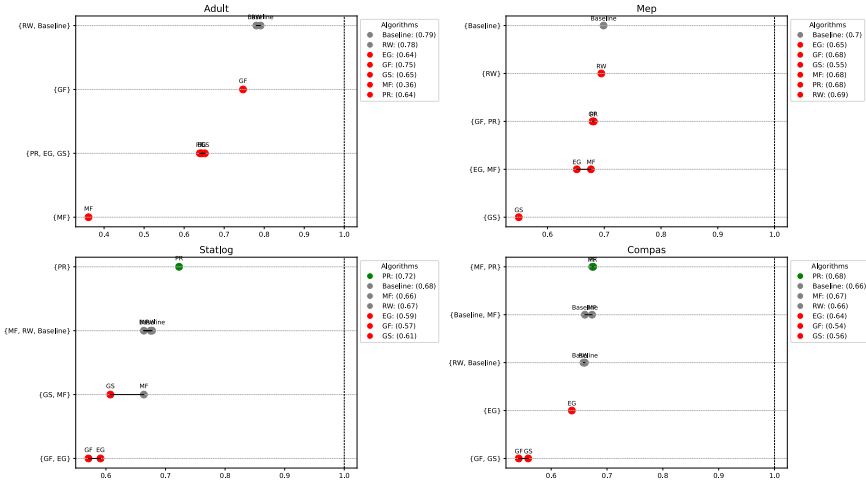


Figure 4.9: Impact of Fairness Algorithms on *F1 score* with different datasets. The *algorithms* are Reweighing (RW), Meta Fair Classifier (MF), Gerry Fair Classifier (GF), Exponentiated Gradient Reduction (EG), PrejudiceRemover (PR), Grid Search Reduction (GS).A green dot • indicates a statistically significant improvement against the baseline, a red dot • indicates a deterioration instead. Finally, a gray dot • indicates that there is no statistically significant difference with the baseline.

4.3.3.2 *F1 score Evaluation*

Examining the `F1 score`, I found similar results with accuracy evaluation. Figure 4.9 shows the result of the F1 score across the datasets. In Adult, the Baseline (79%) shows a higher F1 score than the algorithms but is not statistically different from RW (78%). Also, the group {PR, EG, GS} is not statistically different. Conversely, MF (36%) is the worst algorithm, decreasing the percentage considerably. The MF and GF are statistically different from the others.

For the Mep data set, the baseline (70%) also shows higher values, but in this case, it is statistically different. No statistical differences are found between the pairs {EG, MF} and {GF, PR}, indicating similar values within these groups. GS (55%) and RW (69%) are statistically different, with the GS algorithm being the worst in this case.

In Statlog, the PR (72%) algorithm surpasses the F1 score of the Baseline (68%), and PR is statistically different from other algorithms. The {MF, RW, Baseline}, {GS, MF}, and {GF, EG} groups show no statistical differences. GF (57%) is the worst in this case.

In Compas, PR (68%) and MF(78%) algorithms, besides not being statistically different, surpass the F1 score of the Baseline (66%), and there is no statistical difference in the group {Baseline, MF}. Also, the groups {RW, Baseline}, and {GF, GS} are not statistically different, with GF (54%) and GS (56%) having the lowest values. Only the EG algorithm, in this case, is statistically different.

4.3.3.3 *Training Time Evaluation*

Figure 4.10 details the `training time` required for each dataset, indicating that EG and GS consistently require the longest training times across all datasets, marking them as statistically distinct from other algorithms. This observation aligns with the trends identified in RQ2. In the Adult dataset, the EG algorithm takes significantly longer to train (43 seconds) than the Baseline's (2 seconds). The groups {GF, MF, PR} and {MF, Baseline, RW} do

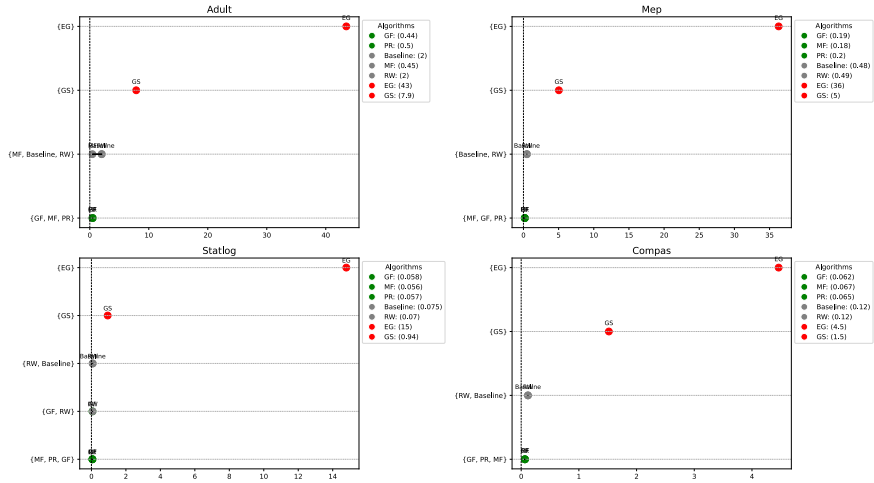


Figure 4.10: Impact of Fairness Algorithms on *Training Time (seconds)* with different datasets. The *algorithms* are Reweighing (RW), Meta Fair Classifier (MF), Gerry Fair Classifier (GF), Exponentiated Gradient Reduction (EG), PrejudiceRemover (PR), Grid Search Reduction (GS).A green dot • indicates a statistically significant improvement against the baseline, a red dot • indicates a deterioration instead. Finally, a gray dot • indicates that there is no statistically significant difference with the baseline.

not show statistical differences in training time, despite GF (0.44 sec), MF (0.45 sec), and PR (0.5 sec) being quicker than the others.

In Mep, the groups {MF, GF, PR} and {Baseline, RW} exhibit no statistical difference in training time, with MF (0.18 sec), GF (0.19 sec), and PR (0.2sec) requiring less time than Baseline (0.48 sec). Even here, EG (36 sec) and GS (5 sec) are the worst algorithms.

For the Statlog dataset, the groups {GF, RW}, {RW, Baseline} and {MF, PR, GF} are not statistically different, with MF (0.056 sec) and PR (0.047 sec) required less time than Baseline (0.075 sec). EG (15 sec) and GS (0.94 sec) are outliers, again needing longer than the rest.

In Compas, the groups {GF, PR, MF} and {RW, Baseline} are not statistically different with GF (0.062 sec), PR (0.065 sec), and MF (0.067 sec) that required less training time than Baseline (0.12 sec). The algorithms EG (4.5

sec) and GS (1.5 sec), even in this case, required more training time than others.

4.3.3.4 *Storage Weight Evaluation*

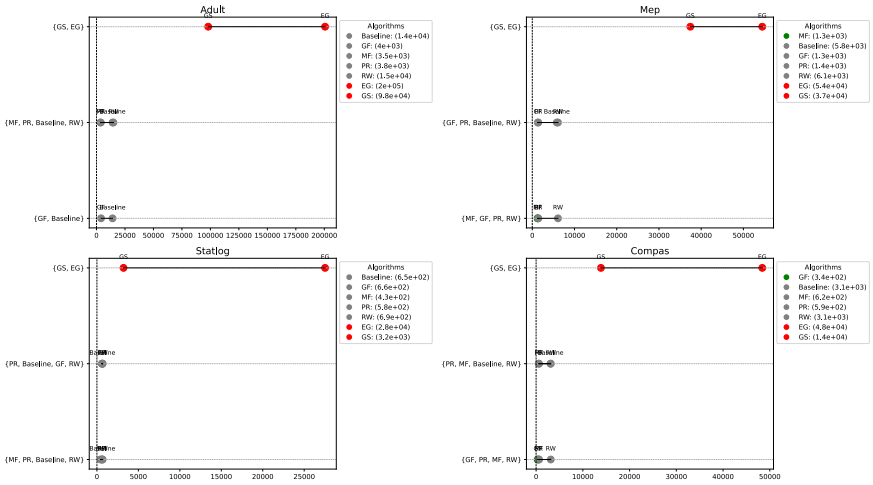


Figure 4.11: Impact of Fairness Algorithms on *Storage Weight (KB)* with different datasets. The *algorithms* are Reweighing (RW), Meta Fair Classifier (MF), Gerry Fair Classifier (GF), Exponentiated Gradient Reduction (EG), PrejudiceRemover (PR), Grid Search Reduction (GS). A green dot • indicates a statistically significant improvement against the baseline, a red dot • indicates a deterioration instead. Finally, a gray dot • indicates that there is no statistically significant difference with the baseline.

The last metric considered for the RQ_3 is `storage weight`, depicted in Figure 4.11. In the Adult dataset, no statistical differences are shown in the groups {MF, PR, Baseline, RW}, {GF, Baseline} and {GS, EG}. The GS and EG algorithms increase the model storage weight, respectively, over 98.000 and 200.000 KB, comparing them with the Baseline, which is only 14.000 KB. On the other hand, MF (3.500 KB), GF (3.800 KB), and PR (4.000 KB), even if there are no statistical differences with the Baseline, reduced the storage weight. In Mep, the groups {GF, PR, Baseline, RW}, {MF, GF, PR, RW} and {GS, EG} showed no statistical differences. However,

GS (37.000 KB) and EG (54.000 KB) algorithms are the worst compared with others. Even in these case the algorithms MF (1.300 KB), GF(1.300 KB), and PR (1.400 KB) reduced the storage weight, even though there were no statistical differences with the Baseline. In the Statlog dataset, the group {GS, EG}, {PR, Baseline, GF, RW} and {MF, PR, Baseline, RW} are not statistical difference. Even in this case, GS (3.200 KB) and EG (28.000 KB) have the worst values.

For the Compas dataset the GS (48.000 KB) and EG (14.000 KB) algorithms are the worst compared to others, and they also are not statistically different. On the other hand, the groups {GF, PR, MF, RW} and {PR, MF, Baseline, RW} are not statistically different. GF (3.400 KB) is the best algorithm, even if not statistically different from others. The analysis of the economic metrics *accuracy, f1 score, training time, storage weight* showed that the algorithms used have an impact and are not statistically significant to each other. Therefore, the null hypothesis is rejected $H_0^{B_i,B_j,S_{ec}}$.

RQ$_3$ — Economic Sustainability

*The investigation of **RQ**$_3$ reveals the influence of the chosen algorithms on the economic metrics evaluated. This finding underscores the dual potential of these techniques to improve or decrease model quality. The Accuracy and F1 score of Prejudice Remover in Statlog and Compas is higher than the Baseline. On the other hand, the other algorithms significantly reduced these metrics. Exponentiated Gradient Reduction and Grid Search Reduction algorithms demonstrate higher training times across datasets, indicating increased computational costs, and are statistically different. Additionally, these algorithms notably increase storage weight. Conversely, some algorithms manage to slightly reduce storage weight, though not significantly compared to the Baseline.*

4.4 DISCUSSION AND IMPLICATIONS

In the following section, we first discuss the main results of the study, attempting to provide insights into the added value brought by our research to the current state of the art. Secondly, we provide the implications that our findings have for researchers and practitioners.

4.4.1 *Discussion of the Findings*

At first, we discuss our findings in relation to the state of the art and the qualitative insights that may explain them.

On the relation with the state of the art. Our study builds on and extends the existing body of knowledge by addressing critical gaps and offering a broader perspective on the interplay between the three sustainability dimensions in ML-enabled systems. First, regarding $\mathbf{RQ}_1$, our results corroborate previous findings that position fairness as a cornerstone of social sustainability. While our results align with earlier work, two factors distinguish their significance. By replicating and validating previous findings, our study enhances the *ecological validity* [302] of these insights, i.e., the extent to which experimental results can be generalized to real-world applications. Moreover, our analysis positions fairness as a foundational element for examining its interplay with other dimensions of sustainability, such as environmental and economic. We believe that such an integrated perspective enables the exploration of trade-offs among fairness, energy consumption, and resource efficiency, and our contribution leads to further investigation of these relationships. Second, as for $\mathbf{RQ}_2$, our work addresses a knowledge gap by systematically examining the environmental implications of bias mitigation algorithms. By operationalizing environmental sustainability through metrics such as energy consumption and carbon emissions, we provide new insights into the ecological trade-offs introduced by these algorithms. Our findings reveal that while certain algorithms, e.g., *Exponentiated Gradient Reduction*, significantly improve fairness, they also exhibit substantial energy demands and higher carbon

emissions. Conversely, computationally efficient algorithms, e.g., *Meta Fair Classifier* and *Gerry Fair Classifier*, demonstrate lower environmental impacts, providing practitioners with actionable strategies to balance fairness with ecological responsibility. These results contribute to a more holistic understanding of the sustainability of ML-enabled systems by emphasizing the need to select fairness techniques carefully during the design process, aiming at finding a suitable solution considering the trade-off between the sustainability aspects.

As for **RQ**$_3$, our research expands the scope of fairness studies by framing the evaluation of bias mitigation algorithms within the economic sustainability dimension. As reported earlier, previous work has predominantly assessed these techniques through the lens of accuracy, not considering broader operational implications such as training time and storage requirements. By incorporating these additional metrics, we offer a multi-faceted view of the economic trade-offs associated with fairness-enhancing techniques. Our findings reveal that while certain bias mitigation algorithms introduce computational overhead, such as *Exponentiated Gradient Reduction* and *Prejudice Remover*, they also open avenues for balancing fairness with economic feasibility. Such a perspective enriches the current body of knowledge, providing a framework to assess the practical and financial implications of implementing fairness techniques in real-world settings.

In summary, our empirical study advances the state of the art by integrating fairness into a comprehensive sustainability framework that spans its social, environmental, and economic dimensions. This integrated approach provides a holistic understanding of the trade-offs introduced by bias mitigation techniques, equipping practitioners and researchers with the tools to make informed decisions that balance ethical, operational, and environmental considerations in the design of sustainable and responsible ML-enabled systems.

Bias Mitigation Algorithms Versus Sustainability: A Qualitative Perspective. While our work primarily presented a benchmark study with a quantitative focus, it might also provide qualitative insights into the potential

reasons underlying the observed results. Our findings suggest multiple considerations that highlight the complex interplay between bias mitigation techniques, dataset characteristics, and the stochastic nature of ML processes, which open avenues for further investigation. A first observation from our results is the variability in the performance of bias mitigation algorithms across different datasets, particularly in terms of energy consumption and resource efficiency. This suggests that the characteristics of the datasets themselves—such as feature dimensionality, group imbalances, and overall complexity—can significantly impact determining the results of these algorithms. Specifically, datasets with high dimensionality or imbalanced distributions between privileged and unprivileged groups may require additional computational resources for bias mitigation techniques to converge effectively. This observation is particularly relevant to $\mathbf{RQ}_1$, as our results indicate that fairness outcomes are not solely determined by the algorithms but also by the data properties that interact with algorithmic mechanisms. For example, in the Compas dataset, all three social metrics used have higher values than the other datasets; this could be due to the number of features being lower than the other datasets. In this respect, future research should explore these interactions to provide more tailored guidance on selecting appropriate algorithms based on dataset characteristics.

Another perspective relates to the intrinsic computational characteristics of bias mitigation algorithms themselves. Pre-processing methods such as *Reweighing*, which involve data manipulation that reduces the dataset size, tend to exhibit lower computational overhead. Conversely, in-processing methods like *Exponentiated Gradient Reduction*, which rely on iterative optimization steps to improve discrimination, result in higher energy consumption and longer execution times. While these theoretical expectations provide a framework for anticipating the impact of bias mitigation algorithms on computational performance, our findings also suggest that empirical results may deviate from these expectations due to variations in implementation efficiency, interactions with specific datasets, and differences in model configurations. These insights emphasize the

importance of our benchmark study, which uses quantitative measurements under controlled conditions to capture the practical effects of bias mitigation algorithms. This perspective is directly connected to $\mathbf{RQ}_2$, as it illustrates how algorithm-specific factors interact with sustainability metrics.

In addition, the stochastic nature of ML processes introduces further variability into sustainability metrics. Variations in random initialization, data splits, and probabilistic elements within the algorithms themselves can lead to fluctuations in energy consumption and training time. These stochastic effects may also explain unexpected findings, such as certain algorithms achieving lower resource consumption despite their inherent computational complexity. Understanding how randomness influences sustainability outcomes is critical for improving the robustness and predictability of bias mitigation techniques in real-world applications. Future research could extend these insights by examining how individual ML algorithms are affected by different bias mitigation strategies, enabling more responsible algorithm selection and promoting better sustainability practices.

Finally, our findings indicate that some bias mitigation algorithms may inadvertently regularize model behavior, simplifying decision boundaries or reducing overfitting. For instance, algorithms such as *Meta Fair Classifier* or *Prejudice Remover* might promote less complex models that require fewer computational resources, thereby reducing energy consumption or training time. While this phenomenon is promising from a sustainability perspective, it raises important questions about the trade-offs between simplicity and fairness. This aspect raises the critical issue of whether such regularization aligns with the intended fairness objectives or results in unintended biases in other areas. These considerations are particularly relevant in the context of $\mathbf{RQ}_3$, as they highlight the need to balance fairness improvements with resource efficiency and operational feasibility.

In conclusion, we acknowledge that future research should complement our findings with qualitative methods, like case studies, interpretability analyses, or interviews with practitioners, to uncover the underlying causes

of the observed phenomena. The insights of these qualitative analyses would provide a more comprehensive understanding of the nature of bias mitigation algorithms and the trade-offs between fairness and sustainability in bias mitigation algorithms, possibly guiding the development of more effective and efficient approaches.

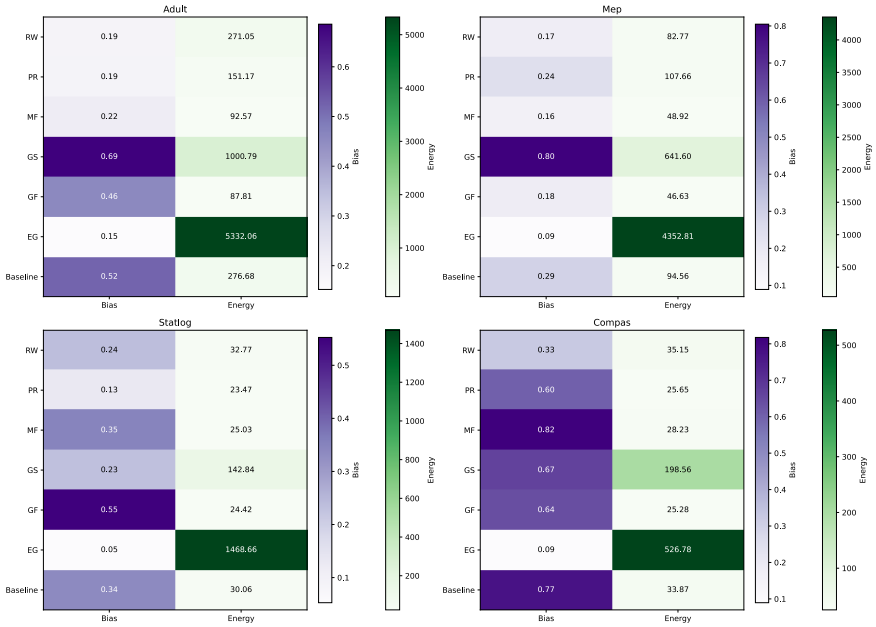


Figure 4.12: Energy Consumption Vs. Cumulative Bias for each dataset. The *algorithms* are Reweighing (RW), Meta Fair Classifier (MF), Gerry Fair Classifier (GF), Exponentiated Gradient Reduction (EG), PrejudiceRemover (PR), Grid Search Reduction (GS). The numerical values represent cumulative bias (purple), where lower values are better, and energy consumption (green), where lower values are better.

4.4.2 *Implications of the Study*

The results of our work have a number of practical implications for researchers, practitioners, and project managers, which we elaborate in the following.

On the need for trade-off analysis to deploy models in real scenarios. The deployment of machine learning systems with bias mitigation algorithms in real-world scenarios often requires practitioners to navigate complex trade-offs among multiple non-functional requirements, such as fairness, energy efficiency, storage costs, and accuracy. Our findings highlight that these trade-offs are not merely theoretical but manifest concretely in different algorithms' performance across various sustainability dimensions. These observations highlight the importance of systematic trade-off analyses during the development and deployment phases of ML models. Current research has studied the balance between accuracy and fairness [65, 163], accuracy and energy efficiency [349] and, more recently, fairness and efficiency [42]. However, our study expands this understanding by jointly studying the three dimensions of sustainability and exploring interactions among additional non-functional requirements, such as storage weight, training time, and carbon emissions.

To address these challenges, practitioners must prioritize trade-offs based on application-specific requirements. For instance, in critical domains where fairness cannot be compromised, high-energy-consuming algorithms such as *Exponentiated Gradient Reduction* might be justified. Conversely, for edge devices or IoT applications where energy and storage constraints dominate, algorithms like *Reweighing* or *Gerry Fair Classifier* may provide a better balance of fairness and resource efficiency. These considerations highlight the importance of designing adaptable systems that can accommodate varying trade-off priorities depending on the deployment context.

Beyond the findings of this study, there is an urgent need for tools and frameworks that can assist practitioners in understanding and optimizing these trade-offs—for example, visualizations that compare cumulative fairness metrics with energy and storage requirements, as demonstrated in Figures 4.12 and 4.13, could help stakeholders make informed decisions. Moreover, future work should explore adaptive, context-aware algorithmic approaches that dynamically adjust their behavior to balance multiple non-functional requirements under specific constraints.

Implication #1. While researchers are starting investigating novel approaches to handle the trade-offs among multiple non-functional requirements, our findings represent a call for further research actions aiming to quantify, optimize, and visualize these trade-offs.

Implication #2. Future research should focus on adaptive and context-aware bias mitigation algorithms capable of dynamically balancing fairness and other sustainability metrics based on application-specific requirements. These advancements could ensure that ML-enabled systems remain effective and efficient in diverse real-world scenarios.

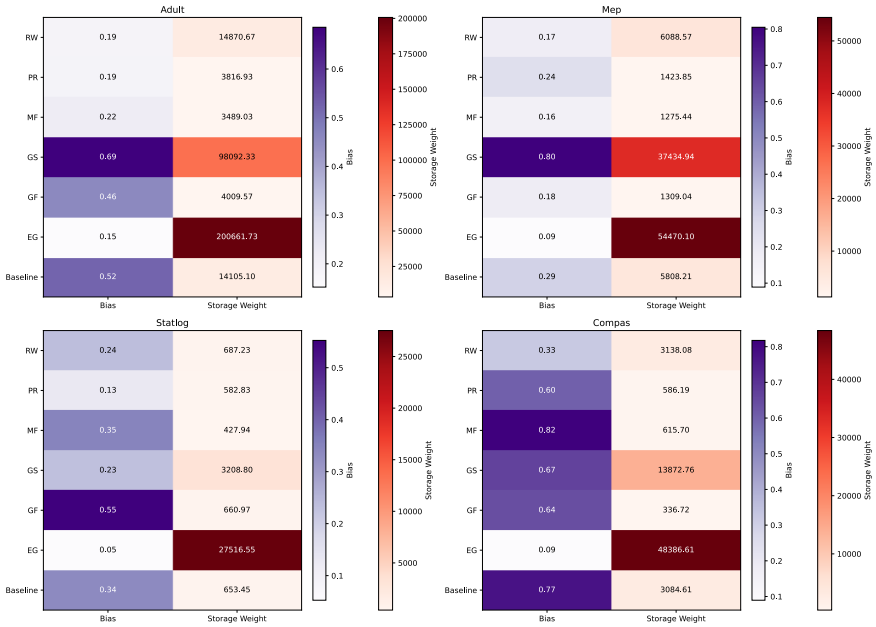


Figure 4.13: Cumulative Bias Vs. Storage Weight for each dataset. The *algorithms* are Reweighing (RW), Meta Fair Classifier (MF), Gerry Fair Classifier (GF), Exponentiated Gradient Reduction (EG), PrejudiceRemover (PR), Grid Search Reduction (GS). The numerical values represent cumulative bias (purple), where lower values are better, and storage weight (red), where lower values are better.

For the sake of completeness, let us further elaborate on these matters by discussing the figures in detail, highlighting the trade-offs between the

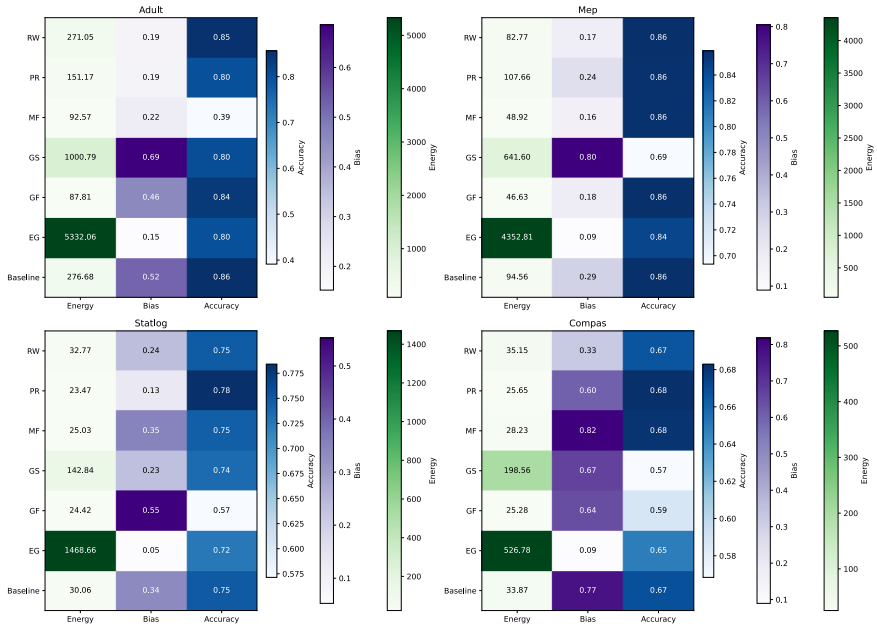


Figure 4.14: Cumulative Bias Vs. Energy Consumption Vs. Accuracy for each dataset. The *algorithms* are Reweighing (RW), Meta Fair Classifier (MF), Gerry Fair Classifier (GF), Exponentiated Gradient Reduction (EG), PrejudiceRemover (PR), Grid Search Reduction (GS). The numerical values represent cumulative bias (purple), where lower values are better; energy consumption (green), where lower values are better; and accuracy (blue), where higher values are better.

various metrics analyzed in this study. Figure 4.12 provides a visualization of the trade-off between cumulative bias [27], calculated as the sum of the absolute values of the three social fairness metrics, and energy consumption for all algorithms across the datasets. As shown in the figure, a high energy demand is associated with fairness improvements, particularly for *Exponentiated Gradient Reduction*, which achieves the highest fairness improvements but consistently creates substantial energy costs. At the same time, *Reweighing* and *Prejudice Remover* demonstrate a more balanced trade-off by delivering moderate fairness gains while consuming significantly less energy. These insights are potentially interesting for prac-

titioners operating in energy-constrained environments, where energy efficiency must be weighed carefully against fairness objectives.

Figure 4.13 shows the relationship between cumulative bias and storage weight. The results indicate that algorithms like *Reweighing* and *Prejudice Remover* maintain lower storage footprints while achieving significant fairness improvements. These characteristics make them highly suitable for resource-constrained deployments, such as IoT or mobile applications. At the same time, algorithms like *Exponentiated Gradient Reduction* and *Grid Search Reduction* demonstrate a trend of higher storage costs relative to their bias reduction capabilities, emphasizing the importance of assessing average algorithmic behaviors when making deployment decisions in environments with strict storage constraints. This analysis highlights the importance of aligning algorithmic choices with resource-related non-functional requirements in real-world scenarios.

Figure 4.14 provides a multi-dimensional perspective by illustrating the interplay among cumulative bias, energy consumption, and accuracy. This analysis highlights the intricate trade-offs that practitioners must navigate. The results highlight that no algorithm achieves optimal performance across all three dimensions simultaneously. *Exponentiated Gradient Reduction*, for instance, delivers the highest fairness improvements but at the expense of accuracy and energy efficiency. In contrast, *Reweighing* and *Meta Fair Classifier* provide a more balanced performance, making them more appropriate for use cases that require a compromise between fairness, accuracy, and energy consumption. This figure reinforces the importance of a holistic evaluation framework that considers multiple non-functional requirements simultaneously, tailored to specific deployment contexts. As a consequence of this analysis, we may argue that there is a need for decision-making tools that help practitioners navigate these trade-offs based on their specific operational constraints.

Implication #3. Practitioners would need decision-support systems that prioritize deployment context when selecting bias mitigation algorithms. These systems should integrate multi-objective optimization techniques to evaluate trade-offs between fairness, energy consumption, storage efficiency, and accuracy dynamically. Such tools could guide algorithm selection based on the operational constraints and goals of specific applications, such as IoT deployments, energy-intensive systems, or high-stakes domains.

Implications for AI Engineering research. Recent research in Software Engineering for Artificial Intelligence has introduced approaches designed to manage multiple protected attributes concurrently [50, 63, 270]. Despite this, popular bias mitigation toolkits like AIF360 and Fairlearn have yet to integrate these advancements, and this results in a lack of experimentation of these solutions in studies like ours. Implementing these approaches in such toolkits would enable developers to reduce model bias in their systems while providing researchers with opportunities to explore additional non-functional requirements. Moreover, multi-objective optimization in bias mitigation should go beyond the traditional fairness-accuracy trade-off. By leveraging our findings and adopting our sustainability framework, researchers should create solutions that consider other non-functional requirements to produce higher-quality systems that align with business needs. Our study specifically addressed sustainability requirements, but extending multi-objective optimization to include a broader range of requirements could help overcome existing limitations and facilitate the creation of responsible and robust ML tools.

Implication #4. Our work emphasizes the need to integrate existing bias mitigation approaches from the literature—particularly those capable of addressing multiple protected attributes simultaneously—into widely used tools. Furthermore, there is a pressing need for new approaches that can effectively handle multiple non-functional requirements, striking a balance between them without compromising any single requirement.

From a broader perspective, the results of our research may improve standards and guidelines designed to manage the non-functional requirements of ML-enabled systems. In particular, our findings have the potential to serve as valuable resources that support and augment existing or developing standards such as *ISO/IEC 25059* and *ISO/IEC 20226*.[4]. These standards, which are currently emerging in the field, aim to establish a clear and comprehensive set of criteria for assessing the quality of software systems. By integrating our findings, it is conceivable that these standards could be further enriched, providing a more robust and practical guide for operators and developers who need to handle non-functional requirements specific to ML-enabled systems. For example, adding guidelines for systematically evaluating trade-offs between energy consumption and accuracy could guide developers in creating more sustainable ML-enabled systems. This integration could facilitate a more systematic and standardized approach to ensuring that these systems are functional but also lightweight, unbiased, and accurate. Moreover, our results have implications at multiple levels. Beyond technical optimizations, they inform policy and strategic decisions within organizations, suggesting a framework for integrating non-functional requirements into the regular standards development workflow. This holistic approach can help institutions create more robust and practical guidelines for developing ML-enabled systems.

Implication #5. Our **RQs** provide insights that may be useful in extending and supplementing emerging regulatory standards by describing relevant non-functional requirements and offering better criteria for evaluating the trade-offs of ML-enabled systems. Moreover, our results highlight current research gaps for further study of non-functional requirements of ML-enabled systems, suggesting that these requirements should be integrated into the development process. This work encourages deeper investigation into how non-functional factors influence system design and performance, potentially defining new directions for future research.

[4]The *ISO/IEC 25059* standard: https://iso25000.com/index.php/en/iso-25000-standards/iso-25059 and ISO/IEC 20226 standard: https://www.iso.org/standard/86177.html

5

INTO THE ML-UNIVERSE: AN IMPROVED CLASSIFICATION AND CHARACTERIZATION OF MACHINE-LEARNING PROJECTS.

5.1 INTRODUCTION

In the realm of SE4AI, researchers primarily engage with three kinds of ML projects: (1) projects that apply ML techniques to solve specific problems, (2) projects that develop ML methodologies and models, and (3) projects that provide support tools and libraries to facilitate ML development. This tripartite categorization is crucial as it enables researchers to tailor their studies according to the specific nature and objectives of the projects under investigation.

Unfortunately, classifying projects according to this framework poses several challenges. First, the interpretation of project descriptions can be biased due to ambiguous or misleading information, which often lacks clarity or provides incorrect details [344, 400]. Second, the boundaries between categories can overlap, as projects may simultaneously apply ML techniques and develop methodologies, making it difficult to assign them to a single category. Third, the diversity in application contexts and the lack of a standardized taxonomy for ML projects contribute to inconsistencies and challenges in accurately categorizing these projects [356]. These challenges can contribute to interpretation bias, undermining the reliability of research findings and the development of best practices in the field. As a result, studies may become less comparable and risk overlooking critical aspects of the projects under investigation.

To exemplify the challenges and potential bias due to project misclassification, let us consider the study recently conducted by Calefato *et al.* [40], who aimed at assessing the support given by MLOps to the automation

of ML projects. To perform this study, the authors required projects using ML techniques to address real-world problems so that they could assess the actual support provided by MLOps. As such, the authors identified 155 candidate ML projects relying on a pattern-matching strategy proposed by Biswas et al. [26]—this verifies the existence of keywords suggesting that the project exploits ML technologies. Upon conducting a follow-up manual inspection, Calefato *et al.* [40] found out that 105 of the 155 were misclassified as ML projects but did not contain ML components; 49 were ML projects defining libraries and/or producing new ML models, and only one was a genuine ML project useful for their investigation. According to their manual investigation, the misclassification was primarily due to buzzwords such as *"Artificial Intelligence"* or *"Machine Learning"* in the *README* file. On the one hand, the study shows that *misclassifying the nature of ML projects may lead to incorrect project sampling that significantly biases the interpretation of the conclusions drawn.* On the other hand, this example emphasizes *the need for approaches that may equip researchers with instruments to support the collection of ML projects based on their nature.*

Some attempts to mitigate those classification issues have been proposed so far. Widyasari *et al.* [364] released NICHE, i. e., a dataset of ML projects classified based on their engineering quality, distinguishing between "well-engineered" and "non-well-engineered" systems. Gonzalez *et al.* [136] differentiated projects based on their purpose, such as whether they offer solutions to specific problems or provide toolkits for developing ML solutions.

While these classification schemas help provide an initial framework to understand the nature and purpose of ML projects, they would have only partially addressed the issues exemplified by Calefato et al. [40]. On the one hand, the classification by Widyasari *et al.* [364] would focus solely on the engineering properties of candidate ML projects. On the other hand, the classification by Gonzalez *et al.* [136] would mitigate the misclassification of non-ML projects but fail to differentiate ML projects that apply ML techniques to solve real-world problems or develop new ML technologies,

thus lacking a *granular classification* of the ML projects. To illustrate the importance of granularity, consider two repositories commonly labeled as *'Applied AI and ML Projects'* under Gonzalez et al.'s classification schema. The first, PERSON-DETECTION-AND-TRACKING [120], demonstrates a complete pipeline for detecting and tracking individuals using computer vision techniques. This project involves training custom ML models, implementing object detection algorithms, and fine-tuning parameters, qualifying it as a *producer* of machine learning models, i.e., a project that uses ML libraries to create a custom ML solution. In contrast, the second repository, ONTO2VEC [121], applies pre-trained models to generate vector representations of biological ontologies. Rather than creating new models, it focuses on using existing ML models for knowledge representation and analysis, making it a *consumer* of existing ML solutions. Both repositories fundamentally differ in goals and technical requirements, yet current classification schemas might group them under the same broad category, *'Applied AI and ML Projects.'* This lack of differentiation may hinder their distinct purposes, complicating efforts to analyze project characteristics, compare methodologies, and draw actionable insights tailored to specific project types.

As highlighted in **Gap #3**, there is still a lack of automated support for classifying ML projects. To deal with such a granularity issue, this chapter builds on top of the classification proposed by Gonzalez et al. [136], proposing an automated heuristic-based mechanism, coined **M**ACHINE LEARNING **A**UTOMATED **R**ULE-BASED CLASSIFICATION **K**IT (MARK), to classify GITHUB PYTHON projects at a finer-grained level based on PYTHON APIs analysis: *'ML-Model Producer,' 'ML-Model Consumer,'* and *'ML-Model Producer & Consumer.'* Specifically, this mechanism is meant to be *combined* with the automated classification instrument proposed by Gonzalez et al. [136]: while these authors proposed a mechanism to distinguish between *'ML-Libraries & Toolkit'* and *'Applied Projects,'* the contribution focuses on refining the classification of the applied projects to provide a more detailed understanding of their nature, differentiating the way ML is

used. This integrated classification procedure offers researchers a powerful tool to enhance the precision and relevance of analyses in SE4AI.

The potential value of MARK can be illustrated by reconsidering the challenges encountered by Calefato et al. [40]. Had MARK been available, it could have addressed the reported challenges by providing a more granular classification that distinguishes among *'ML-Model Producer,' 'ML-Model Consumer,'* and *'ML-Model Producer & Consumer'* projects. This finer granularity would have allowed the researchers to filter out irrelevant repositories more effectively and categorize the remaining projects according to their specific nature and purpose. Consequently, Calefato et al. [40] could have improved construct validity by ensuring that their dataset better reflected the intended constructs of their study, i.e., real-world ML projects. Additionally, conclusion validity would have been strengthened, as their findings would have been based on a more accurately classified and representative dataset, thereby reducing biases introduced by manual corrections. Furthermore, MARK's automated approach could have significantly reduced the manual effort required to validate project classifications. Systematically distinguishing between different types of ML projects would have eliminated the need to rely on error-prone manual sampling methods, allowing researchers to focus on analyzing genuinely relevant projects. This improvement would save time and ensure greater consistency in project selection, ultimately leading to more reliable research outcomes.

I validate the classification schema through two empirical studies. First, I assess the accuracy of the classifier on 4,603 ML projects from two well-known datasets, NICHE [364] and the dataset provided by Gonzalez *et al.* [136]. Second, I characterize the classified PYTHON projects using engagement and collaboration, activity, and structural metrics. While the first empirical study focuses on showcasing the capabilities of MARK, the second study aims to demonstrate the practical benefits of the classification schema for researchers. Specifically, if significant differences are observed between *'ML-Model Producer,' 'ML-Model Consumer,'* and *'ML-Model Producer & Consumer'* projects in terms of the analyzed metrics, it would

validate the importance of distinguishing and classifying projects based on their nature, thus highlighting the schema's value in enhancing the accuracy and relevance of research findings. The main findings of the work show that the classifier has high F1-scores, especially when classifying *'ML-Model Producers.'* As for the classification of *'ML-Model Consumers,'* I identify inherent challenges that may foster further research on the matter. Perhaps more interestingly, I identify significant differences among the classified projects in terms of the repository metrics considered, which demonstrate the usefulness of classifying PYTHON ML projects according to their nature and purpose, other than providing initial insights into the socio-technical dynamics of ML projects. Particularly, projects belonging to the categories *'ML-Model Producer & Consumer'* and *'ML-Libraries & Toolkit'* exhibit notable differences in terms of various repository metrics, such as size, activity, and the number of contributors, compared to projects in the *'ML-Model Producer'* category. The findings suggest that *'ML-Model Producer & Consumer'* and *'ML-Libraries & Toolkit'* projects are often developed and maintained within larger and more complex environments, which may reflect their broader scope and greater complexity. I conclude the work by discussing the major implications of the study and the outlook for future research enabled by this work.

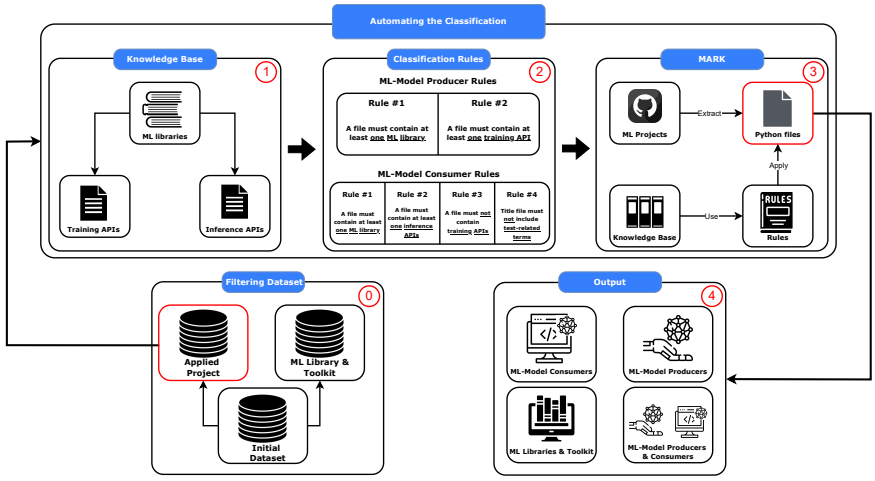


Figure 5.1: MARK: Overview of the Design.

5.2 MARK: MACHINE LEARNING AUTOMATED RULE-BASED CLASSIFICATION KIT

In this section, I introduce **M**ACHINE LEARNING **A**UTOMATED **R**ULE-BASED CLASSIFICATION **K**IT (MARK), a novel approach for the classification of machine learning projects. MARK is based on heuristics, built from a Knowledge Base of APIs obtained on a filtered dataset. The following sections describe the design of the approach, while Figure 5.1 provides an overview of the major steps applied to design this approach.

5.2.1 *Dataset Specification*

This work builds on the top of the study conducted by Gonzalez *et al.* [136], which introduced an automated classification procedure to distinguish between *'ML-Libraries & Toolkit'* and *'Applied Projects'* using natural language processing. I aim to further refine the classification of projects within the *'Applied Projects'* category. Consequently, the input for this approach is represented by a dataset that has been previously classified as *'Applied Projects.'* Currently, MARK focuses on categorizing applied projects containing components written in Python, given its success as a valuable and widely used programming language with many available ML libraries [282]. The focus on Python can be extended to include compatibility with other programming languages.

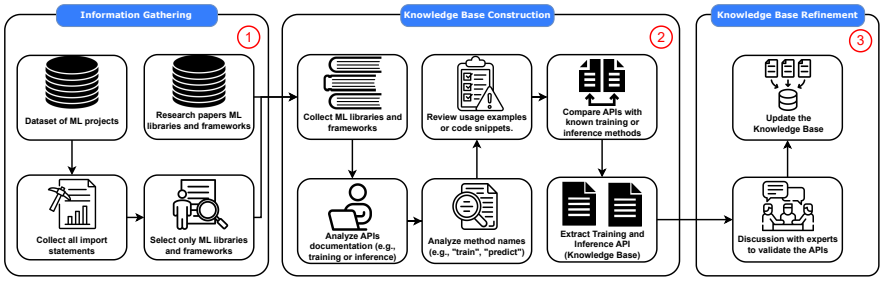


Figure 5.2: Overview of the Knowledge Base Construction and Refinement.

5.2.2 *Definition of Knowledge Base*

MARK leverages ML libraries and API definitions to classify ML projects. The Knowledge Base (KB) includes training and inference libraries and APIs. The definition of the KB involved three main steps, presented in Figure 5.2. As shown in the figure, the workflow began with conducting a preliminary *information gathering* phase (Step ① in Figure 5.2), whose aim was the identification of initial resources for extracting the KB. These initial resources included: (1) relevant research papers, i.e., in this case, the paper by Nguyen et al. [256] from which I derived the initial catalog of ML libraries and frameworks; and (2) state-of-the-art datasets of ML projects, i.e., in this case, those by Rzig et al. [295] and Widyasari et al. [364], which I used to collect additional ML libraries and frameworks. From a methodological standpoint, the first step was implemented through the following phases:

1. I collected relevant datasets and analyzed the literature to identify foundational or seed papers. I selected the one by Nguyen et al. [256] as the most suitable for this scope. However, should the KB need to be updated in the future, multiple papers, including ours, may be used as a starting point to snowball and find additional relevant resources, ensuring that all current catalogs of ML libraries and frameworks are considered. The initial catalog I considered [256] leveraged the most widely utilized PYTHON libraries and frameworks—specifically, the paper identified ten ML and Deep Learning (DL) libraries capable of training and making inferences with ML models. Nonetheless, when analyzing the work by Nguyen et al. [256], I realized that it could not comprehensively capture the current set of ML libraries and frameworks, since it was published in 2019. As such, I had to update this initial set, as described in the following phases. For future updates, however, the papers identified in the first step may already include up-to-date catalogs of ML libraries and frameworks. If this is the case, a researcher interested in

replicating this work may directly proceed to the second step of the KB workflow, i.e., *'Knowledge Base Construction.'*

2. To update the KB, I proceeded with a manual, tool-supported mining exercise against datasets of ML projects. Specifically, I developed a simple static analysis inspection tool to identify additional libraries and frameworks within this dataset. The tool was designed to scan each PYTHON file, extract all import statements, and organize them into a formatted spreadsheet, deleting duplicates and streamlining subsequent analysis and classification. Through this step, I collected all references to libraries `import` and `from` in 1,463 PYTHON files, finding 2,487 unique library imports. These import statements were then manually analyzed by I to determine whether they referenced libraries or served other purposes, e.g., importing local modules. Non-library references were excluded during this filtering process. Afterward, for each identified library, I manually compared it against the catalog provided by Nguyen et al. [256]. If the library was already available within the Nguyen et al.'s catalog, it was filtered out to avoid redundancy. Conversely, if the library was not present in their catalog, it was added to the set of libraries and frameworks considered in this study. As a final step, for each newly identified library, I manually collected its name and linked it to the corresponding official documentation or repository. This step enabled the cataloging of newly emerged libraries and frameworks, each linked to an established reference source. This ensured that the information was accurate, up-to-date, and reflective of the current state of the field, providing a reliable basis for subsequent analyses. The output of this first step was a refined and expanded list of ML libraries and frameworks, comprising 83 libraries, which served as the foundation for the second step of the workflow.

The second step, i.e., the *'Knowledge Base Construction'* (Step ② in Figure 5.2), began by examining the functionality of each library or framework to classify its APIs into categories such as training or inference:

1. I reviewed the official documentation for explicit descriptions of the API's functionality. For each library, he specifically searched for method APIs in its official documentation leveraging its website, e. g., Gluon,[1] or its GitHub repository, e. g., Anago.[2] The rationale for this step is that official documentation may indeed provide explicit descriptions of the API's intended functionality, e.g., whether an API is explicitly designed for tasks such as training models, performing inference, or supporting other ML-related operations. This step helped establish a baseline understanding of the core purpose of the API under consideration.

2. I analyzed method names for predefined keywords (e.g., *"train," "predict"*) aligned with common conventions in ML development. These keywords provided strong cues about the functionality of the API. For example, methods named `train_model()` are likely related to training tasks, while those named `predict_output()` may indicate inference operations. Identifying these patterns informed the subsequent classification of the APIs.

3. I reviewed code examples or snippets to verify operational behavior and functionality. For example, snippets that show the use of an API to fit models on training data or generate predictions can confirm its classification. This step was particularly useful when the documentation lacked explicit descriptions.

4. Finally, I compared the API with similar, well-documented training or inference APIs to assess functional similarity. For example, if an API demonstrates behavior analogous to TensorFlow's `fit()` method, it can be classified as a training API. This comparative approach added a layer of verification.

Based on the collected information, I categorized the 312 APIs from the 83 libraries as either *'training'* or *'inference'* APIs. This represented the

[1] https://cv.gluon.ai/contents.html
[2] https://github.com/Hironsan/anago/tree/master

initial version of the KB. Afterward, cross-checking with domain experts was essential to validate the results and resolve any ambiguities. This led to the final step, i.e., the *'Knowledge Base Refinement'* (Step ③ in Figure 5.2). These experts may include co-authors of the study, as in this case, or external collaborators with relevant domain knowledge. More specifically, the second and third authors of the paper jointly validated the categories assigned to each of the 312 APIs by I, using three predefined categories: "disagree," "further discussion," and "agree." If both authors agreed, the API was accepted without further discussion. In case of disagreement, a focus group meeting was held to decide whether to include the API as part of the KB. Following the validation, I computed Cohen's Kappa [70] to measure inter-rater agreement, which reached 0.90—indicating a high level of agreement among the three inspectors involved in the KB refinement process. As a result of the validation, 308 APIs were included in the KB, while four were excluded because they did not refer to training or inference mechanisms. The last step outputs a refined set of ML libraries and frameworks, each accompanied by classifying its APIs into categories such as training or inference.

5.2.3 *Classification Rules*

To build MARK I adopted a heuristic-based approach. The reason for this decision is twofold. First, heuristic methods provide greater transparency and interpretability in the classification process, making it easier to understand the rationale behind each decision. Second, given the nature of the problem and the relatively unexplored state of the art, I prioritized evaluating the performance of heuristic-based solutions due to their simplicity and efficiency. This was particularly important because the approach is designed to be scalable and suitable for large-scale analyses, such as mining software repositories. Had the heuristic-based methods shown inadequate performance, I would have considered more complex machine learning techniques, as recommended by prior research [108]. However, exploring these advanced techniques remains part of the future research agenda. As

for the classification criteria, it is worth remarking that I was interested in defining classification rules that only discriminated among *'ML-Model Producers,' 'ML-Model Consumers,'* and *'ML-Model Producers & Consumers,'* as the input of MARK is represented by a dataset where *'ML Libraries & Toolkits'* projects are already classified. The classification rules were then defined according to the heuristics reported in the following:

ML-MODEL PRODUCER: I examined API methods related to model training. A project was labeled as an *'ML-Model Producer'* if it contained an import statement for an ML producer library and an invocation of a related training method. Let F be a file belonging to the project P, let L be the set of ML libraries, and let A be the set of training APIs; I first defined the following two predicates:

$$\text{containsMLLibrary}(F, l) = \begin{cases} \text{True} & \text{if } F \text{ contains } l, \text{ with } l \in L \\ \text{False} & \text{otherwise} \end{cases}$$

$$\text{containsTrainingAPI}(F, a) = \begin{cases} \text{True} & \text{if } F \text{ contains } a, \text{ with } a \in A \\ \text{False} & \text{otherwise} \end{cases}$$

Starting from the definition of these predicates, I classify a project as *'ML-Model Producer'* if it meets **both** the following classification rules:

Rule #1: $\exists\, l \in L : containsMLLibrary(F, l)$ i. e., a file must contain at least one ML library;

Rule #2: $\exists\, a \in A : containsTrainingAPI(F, a)$ i. e., a file must contain at least one training API.

In other terms, if there exists at least one file F in the project such that F contains at least one ML library and at least one training API, the project was labeled as an *'ML-Model Producer'*.

ML-MODEL CONSUMER: I searched for API methods associated with model inference. A project was classified as an *'ML-Model Consumer'* if the library and the inference methods were identified. In formal terms, let the definitions of $containsMLLibrary(F, l)$, $containsTrainingAPI(F, a)$ be inherited from the *'ML-Model Producer'* classification context, let I be the set of inference APIs, and T be the set of testing-related words (i. e., test, eval, example, and validate) I define the following predicates:

$$\text{containsInferenceAPI}(F, i) = \begin{cases} \text{True} & \text{if file } F \text{ contains i} \in \text{I} \\ \text{False} & \text{otherwise} \end{cases}$$

$$\text{containsTesting}(F) = \begin{cases} \text{True} & \text{if filename } F \text{ contains \{test, eval,} \\ & \text{example, validate\}} \\ \text{False} & \text{otherwise} \end{cases}$$

From the predicates, I classify a project as *'ML-Model Consumer'* if a file F in the project meets **all** the following classification rules:

Rule #1: $\exists\ l \in L : containsMLLibrary(F, l)$ i. e., a file must contain at least one ML library;

Rule #2: $\exists\ i \in I : containsInferenceAPI(F, a)$ i. e., a file must contain at least one inference API.

However, *'ML-Model Producers'* sometimes use the same methods as the *'ML-Model Consumer'* to test their models, which could increase the presence of false positives in this study due to possible misclassifications. To mitigate possible threats to validity, I applied two additional criteria:

Rule #3: $\nexists\ a \in A : containsTrainingAPI(F, a)$ i. e., a file must not contain training API.

Rule #4: $containsTesting(F) = False$ i. e., the filename does not include sub-strings related to testing.

In other words, if there exists at least one file F in the project such that F contains at least one ML library, at least one inference API, but the same file F does not contain a training API, and does not use the model only for testing, the project was labeled as an *'ML-Model Producer'*.

It is important to remark that the two sets of criteria that allow us to classify projects respectively in *'ML-Model Producer'* and *'ML-Model Consumer'* are not mutually exclusive. If a file F of a project P satisfies all the rules for classification as an *'ML-Model Producer,'* and another file F' of the same project P satisfies all the rules for classification as an *'ML-Model Consumer,'* then the project P is classified as an *'ML-Model Producer & Consumer.'* As an implication of the heuristics reported above, it is possible for MARK to classify an entire project as *'ML-Model Producer & Consumer,'* but not an individual file as *both* a producer and a consumer of ML models. This design decision is not due to the impossibility of a file theoretically serving as both a producer and a consumer. Rather, it is the result of a deliberate design decision in MARK to exclude this scenario. The heuristic rules were designed to assign each file a single dominant role (producer or consumer) to address a critical challenge in identifying real inference services within an ML project. Specifically, APIs that call a newly built model can serve two distinct purposes: actual inference (using the model in production) or evaluation during the training phase (testing its performance). The key challenge was distinguishing between these two cases. To address this challenge, I *assumed* that a Python file maintains a consistent goal. If a file includes training APIs and invokes inference APIs, *I interpreted the latter as being used for evaluation rather than production-level inference.* This assumption was guided by the nature of ML practices observed during the preliminary analyses and the trade-offs involved in ensuring the precision of this approach. In the context of the empirical study, I analyzed the impact of the *file consistency assumption* on the classification performance of this approach (see Section 5.4).

As the reader may observe, the classification rules look pretty straightforward. While the rules themselves may seem simple, their design and implementation were anything but trivial. The true innovation lies in the systematic and effective application of API-related information to classify entire software projects. This process was far from linear; it required multiple iterations and refinements to ensure accuracy and robustness. As such, the technical novelty is not merely in the classification itself, but in how I harnessed API data in a comprehensive, scalable manner, capable of supporting large-scale analyses while maintaining a high level of precision.

5.2.4 *Tool-Supported MARK Classification*

To provide researchers with a readily-usable instrument, I implemented the classification rules into an automated instrument. This instrument, coined MARK, has two main functionalities: (i) cloning GITHUB projects from a predefined list and storing their files locally, and (ii) automatically classifying the cloned ML projects. Specifically, the tool explores all PYTHON files in a repository and applies the previously described rules to classify each project.

5.3 RESEARCH QUESTIONS AND METHODICAL OVERVIEW

To overcome the limitations of the state of the art described in the previous section, the *goal* of the study is to refine the classification of ML projects, with the *purpose* of characterizing them better. The *perspective* is of researchers interested in understanding the nature of ML projects and having a framework to more precisely and efficiently identify study subjects based on the scope of their investigations.

Specifically, I have devised a two-stage investigation. In the first stage, I designed and evaluated a novel API-based, heuristic tool to classify ML projects named **M**ACHINE LEARNING **A**UTOMATED **R**ULE-BASED CLASSIFICATION **K**IT (MARK). As detailed in the following sections, MARK first collects existing datasets to acquire a knowledge base, which is leveraged

to automatically classify ML projects according to their nature, distinguishing between projects that develop ML models i. e., *'ML-Model Producers'* and projects that utilize ML techniques i. e., *'ML-Model Consumer.'* This investigation addresses the following research question:

> **RQ_1—Projects Classification**
>
> *How effectively can I classify ML projects according to their inherent nature and functionality?*

Once I had assessed the effectiveness of the automated classification mechanism, I performed a further step ahead to showcase the usefulness of the classification. In particular, I analyzed the differences between the classified projects using repository metrics, i.e., community engagement, activity, and contributor collaboration. Such an analysis does not aim to provide an exhaustive overview of the use of the classification but rather to provide insights into how to leverage it to understand project dynamics and community involvement. Should there be differences, I would have demonstrated the meaningfulness of classifying projects according to their nature and functionality, highlighting the potential benefits such classification may have for researchers interested in assessing the socio-technical dynamics of ML projects. As such, I addressed the following research question:

> **RQ_2— Project Characterization**
>
> *How does the classification of machine learning projects relate to repository metrics?*

To report the results, I follow the guidelines by Wohlin et al. [368] and apply the *"General Standard"* recommended by the *ACM/SIGSOFT Empirical Standards*.[3]

[3] https://github.com/acmsigsoft/EmpiricalStandards.

5.4 RQ_1. EMPIRICAL STUDY: ON THE PERFORMANCE OF MARK

The *goal* of the first empirical study is to assess the performance of MARK in accurately classifying ML projects as either *'ML-Model Producer'* and *'ML-Model Consumer,'* with the *purpose* of validating the reliability and effectiveness of the classification schema. The *perspective* is of researchers and practitioners in the SE4AI field, who are interested in ensuring that the classification tool can be effectively applied in real-world scenarios, providing a robust foundation for further analysis and decision-making in ML project categorization.

5.4.1 *Data Collection*

Following the dataset specification defined in Section 5.2.1, I applied an extraction and filtering process to ensure compatibility to the approach. I started the data collection using two popular datasets. The first is a dataset proposed by Gonzalez *et al.* [136] and revised by Rzig *et al.* [295], which includes 4,031 ML projects. The second is NICHE [364], a dataset containing 572 ML projects. Combining them, I started the study with 4,603 ML projects. Before using these datasets, I applied a series of preprocessing steps to ensure consistency. Therefore, I integrated the two datasets and executed the following steps:

1. **Deletion of duplicate projects.** I first removed duplicated projects: specifically, I found 117 duplicates, which led to a set of 4,486 unique projects.

2. **Removal of Toy Projects and Tutorials.** I manually inspected the NICHE dataset to identify possible toy, empty, or tutorial projects. Please consider that I cleaned only the NICHE dataset as the dataset by Gonzalez *et al.* [136] has already been revised and cleaned by Rzig *et al.* [295]. More specifically, the first two authors analyzed the NICHE projects by reading the descriptions of their main pages on GITHUB and all websites linked to those pages. At the end of this step, 48 projects were removed.

3. **Collection of ML projects.** I cloned projects from GITHUB to verify their availability. I found that 92 projects had migrated to other version control systems or closed their repositories. By executing this step, I successfully collected 4,346 projects.

4. **Selection of ML-Applied projects**: This step aimed to distinguish all the considered projects between the two categories proposed by Gonzalez *et al.* [136], i. e., *'ML-Libraries'* and *'Applied Projects.'* Since the approach focuses on the second category, I extended this categorization to the second dataset to ensure consistency across all analyzed projects. Specifically, I replicated the procedure described by Gonzalez *et al.* [136] and labeled each repository in the NICHE dataset [364] as either "applied ML" or "ML libraries." The first two authors manually classified the projects using the same methodology as Rzig et al. [295], originally applied to the Gonzalez et al. [136] dataset. Conflicts or unclassified projects were resolved through discussion, and to ensure reliability, the third author validated the classifications. Any disagreements were addressed in an additional meeting to finalize the categorization. As a result, 245 projects were classified as "ML libraries," and 157 as "applied ML."

Finally, I collected the set of applied ML projects. As a final result of the dataset extraction phase, I collected 3,005 *'ML applied projects'* and 1,349 *'ML-Libraries & Toolkits.'* To maximize the number of included projects, I did not impose additional constraints on factors such as years of activity, number of stars, or other characteristics, as these filters had already been applied [136, 295, 364].

Table 5.1: Statistical Metrics.

Metric	Formula
Accuracy	$(TP+TN)/(TP+TN+FP+FN)$
Precision	$(TP)/(TP+FP)$
Recall	$(TP)/(TP+FN)$
F1-Score	$(2\times TP)/(2\times TP+FP+FN)$

5.4.2 *Data Analysis*

To answer $\mathbf{RQ}_1$, I aimed to understand how accurate MARK is in classifying projects through four well-known statistic metrics [276] (see Table 5.1). To validate the MARK, I created an oracle to manually classify each project into two categories: *'ML-Model Producer'* or *'ML-Model Consumer.'*

These categories were chosen because the approach specifically targets them, building on the work of Gonzalez et al. [136] and the *'ML Library & Toolkit'* has been manually classified, making further validation unnecessary.

To validate the approach, I considered the set of projects that I labeled as "applied ML" i. e., I selected 3,005 projects. However, manually verifying the classification correctness requires high classification time to validate all of them manually. Therefore, I selected a representative subset for classification. In particular, I employed simple random sampling to produce unbiased samples from the dataset population [15, 332]. I utilized an online sample size calculator to draw statistically significant conclusions and estimate the necessary sample size.[4] This tool enabled us to determine the appropriate number of samples to maintain a margin of error of 5% and a confidence level of 95% and ensured that the sample size was sufficient to achieve reliable and valid results, thereby enhancing the robustness of the validation process. The first and second authors independently manually classified the oracle. For each project, they marked whether a project is an *'ML-Model Producer,'* an *'ML-Model Consumer,'* or both. The third author reviewed and verified the classifications for objectivity to ensure the oracle's validity and maintain unbiased judgment. I considered a statistically significant sample of 341 randomly extracted projects.

Besides the analysis of the MARK classification as a whole, I also conducted a step-by-step methodology to evaluate the contribution of each component of the approach. I began by assessing the impact of the knowledge base on a baseline version of MARK, which includes only Rules #1 and #2 for the classification of *'ML-Model Producer'* and *'ML-Model Con-*

[4] https://www.qualtrics.com/blog/calculating-sample-size/

sumer.' This involved the comparison between the performance metrics obtained when using the original knowledge base from Nguyen et al. [256] with the enhanced knowledge base developed as part of the study. Afterward, I investigated the effect of the heuristic rules on the classification performance of *'ML-Model Consumer,'* focusing on the role played by Rules #3 and #4. In this respect, it is worth remarking that I could not investigate the effect of removing Rules #1 and #2 from the classification of *'ML-Model Producer'* and *'ML-Model Consumer'*: these rules are indeed the foundational heuristics for the classification of the projects. Rule #1 ensures that files contain at least one ML library, while Rule #2 enforces the presence of at least one relevant API (training APIs for producers and inference APIs for consumers). Without these rules, MARK would lack the fundamental criteria for distinguishing between producer and consumer roles, making it impossible to perform any meaningful classification. As such, Rules #1 and #2 are indispensable for the functioning of MARK and were locked in all configurations, serving as the baseline upon which additional rules and refinements were evaluated. As a consequence of these considerations, it is also worth remarking that I could not proceed with a finer-grained investigation of the heuristic rules composing the *'ML-Model Producer'* classification. This is because the classification relies solely on the two foundational rules and does not involve any additional heuristics. This explains why the analysis focused exclusively on the *'ML-Model Consumer'* classification, namely on the case where the impact of the additional heuristic rules could be meaningfully evaluated.

Table 5.2: Confusion Matrix for *'ML-Model Producer'*

		Predicted	
		Negative	Positive
Actual	Negative	91	1
	Positive	13	236

Table 5.3: Confusion Matrix for *'ML-Model Consumer'*

		Predicted	
		Negative	Positive
Actual	Negative	252	10
	Positive	25	54

5.4.3 *Analysis of the Results*

In this section, I evaluate the performance of MARK in classifying ML projects. The evaluation is divided into three parts. First, I analyzed how MARK identifies the categories *'ML-Model Producer'* and *'ML-Model Consumer,'* discussing the misclassifications and their causes. This evaluation assessed MARK's capability to distinguish between these categories and identifies challenges and opportunities for improvement. Second, I conducted a configuration analysis to examine the contribution of each component and classification rule in MARK. This analysis helps determine which elements have the most significant impact on classification accuracy. Finally, I investigated whether MARK can recognize a new category (i. e., *ML-Model Producer & Consumer'*) that combines the characteristics of *ML-Model Producer'* and *'ML-Model Consumer.'* The rationale for this approach is that classifying a project as both a producer and a consumer does not involve applying additional heuristics. Rather, it is a direct consequence of the existing rules used to classify projects as either producers or consumers. In other words, if a project contains files satisfying both sets of conditions rules, it is assigned to this dual-category. Consequently, misclassifications reported for the two base categories inherently propagate to errors in this combined classification. By adopting this reporting approach, I could first reveal the accuracy of the heuristics applied by the approach and then analyze how its misclassifications would impact the accuracy of the combined classification.

5.4.3.1 *Assessment of 'ML-Model Producer' and 'ML-Model Consumer' Classification*

Table 5.7 shows the confusion matrix for *'ML-Model Producers.'* The high number of true positives (236) and negatives (91) indicates MARK effectively identifies projects producing ML models. It returned only a false positive, demonstrating that the tool is unlikely to classify projects that do not produce any ML models as producers. However, the 13 false negatives show that the approach occasionally misclassifies producer projects

as non-producers. This misclassification can occur due to projects using libraries that are not part of the APIs KB or building custom neural networks [406, 410]. Overall, the matrix highlights the strong performance of MARK in identifying ML Model Producers, which is critical for categorizing and analyzing these projects. Table 5.3 shows the confusion matrix for the *'ML-Model Consumer'* classification, highlighting a more challenging scenario. True negatives (252) and positives (54) indicate that the approach correctly identifies many non-consumer and consumer projects. However, 25 false negatives and 10 false positives highlight the difficulty of distinguishing *'ML-Model Consumers'* from other projects. The number of false positives suggests that the approach sometimes confuses projects with features similar to those of consumers. Additionally, the false negatives indicate cases where real consumer projects are not recognized. Several factors contribute to these misclassifications. Some libraries allow for predictions by directly entering the variable into the model without using specific inference APIs [119, 124]. The approach may miss projects using external libraries that are not included in the initial set [122, 127]. Projects sometimes create custom APIs for inference, which are not included in the KB. This approach may incorrectly classify these as *'ML-Model Consumers'* since it detects the inference method but does not account for its actual usage context [123, 405, 411]. These issues highlight the need to refine the classification criteria to improve the accuracy and recall for identifying *'ML-Model Consumers.'*

Table 5.4: Statistical Metrics for *'ML-Model Producers'* and *'ML-Model Consumers'*

Metric	*'ML-Model Producers'*	*'ML-Model Consumers'*
Accuracy	95.89 %	89.74 %
Precision	99.58 %	84.62 %
Recall	94.80 %	68.75 %
F1-Score	97.13 %	75.86 %

Table 5.4 shows the statistical metrics for the classification performance of *'ML-Model Producer'* and *'ML-Model Consumer.'* For *'ML-Model Pro-*

ducer,' the approach shows an accuracy of 95.89%, precision of 99.58%, recall of 94.80%, and an F1-Score of 97.13%. These high values indicate that the approach reliably identifies producer projects with low misclassification. In contrast, the metrics for *'ML-Model Consumer'* show lower performance, i.e., an accuracy of 89.74%, precision of 84.62%, recall of 68.75%, and the F1-Score of 75.86%. This result suggests that the approach struggles more with identifying *'ML-Model Consumer'* projects, particularly in terms of false negatives, which impact the overall recall of the approach. In this respect, a first observation to make is concerned to the relative importance that precision and recall have for the use case envisioned for MARK. I argue that the high precision of MARK is sufficient to meet its primary objectives. Since MARK is primarily intended for research, its high precision ensures that the projects it classifies are both reliable and accurate, which is crucial for the integrity of subsequent analyses. By minimizing false positives, MARK enables researchers to study ML project characteristics with greater confidence, reducing the risk of skewed results caused by misclassified instances. This precision is particularly valuable in research contexts where data quality and trustworthiness are paramount, ensuring that the findings can be confidently applied in practice. While the lower recall suggests that some projects may be missed, this is likely less concerning in a research context where a complete dataset is often unnecessary, i.e., the approach would still guarantee the selection of a significant and representative sample of projects for analysis. Although further refinements could enhance recall, I believe that MARK already effectively supports its research-oriented goals.

In any case, to better understand the causes of the classification errors made by MARK, the first three authors conducted a joint manual analysis of the misclassified instances. The goal of this collaborative effort was to evaluate the approach's errors, identify specific areas where the heuristic-based method may need refinement, and summarize the main causes of misclassification along with their relative frequency. Table 5.5 reports the causes of misclassification broken down by classification (*'ML-Model Producers'* and *'ML-Model Consumers'*) and further divided into false

Table 5.5: Causes of Misclassification and their Frequency.

Reason Misclassification	ML-Model Producer		ML-Model Consumer	
	False Positive	False Negative	False Positive	False Negative
Custom Solution	0	8 (2.35%)	1 (0.29%)	15 (4.40%)
Keyword for API not exclusive for Inference	0	1 (0.29%)	3 (0.88%)	3 (0.88%)
Testing and Evaluation	0	0	7 (2.04%)	0
Do not use ML API	0	3 (0.88%)	0	2 (0.59%)
Library not included in this work	0	1 (0.29%)	0	2 (0.59%)
Both training and inference in the same file	0	0	0	2 (0.59%)
Training Keyword not used for training	0	0	0	1 (0.29%)
Not Machine Learning	1 (0.29%)	0	0	0
Total	1 (0.29%)	13 (3.81%)	10 (2.93%)	25 (7.3%)

positives and false negatives. This division highlights how each cause specifically affects the respective classifications.

For *'ML-Model Producers,'* the *'Custom Solution'* category was the most common source of misclassification, with eight false negatives (2.35%). This suggests that projects implementing bespoke solutions for model training without relying on commonly recognized ML APIs pose a challenge to the classification approach. These custom implementations often lack standard markers, making them harder to detect using heuristic-based methods. Other sources of false negatives were *'Do not use ML API,'* with three false negatives (0.88%), and *'Library not included in this work'* with one false negative (0.29%). A single false positive (0.29%) was recorded in the *'Not Machine Learning'* category. Overall, the classification of *'ML-Model Producer'* projects led to a single false positive (0.29%) and 13 false negatives (3.81%), for a total of 14 (4.10%) misclassification cases.

For *'ML-Model Consumers,'* the most frequent misclassification was in the *'Custom Solution'* category, where projects implemented their methods instead of using the standard APIs, with 15 false negatives (4.40%) and one false positive (0.29%). *Testing and Evaluation* have five false positives (2.04%), where projects were misclassified as they used files solely for testing and evaluating models rather than consuming them. *Keyword for API not exclusive for Inference* have three false negatives (0.88%) and three false positives (0.88%) where keywords typically associated with inference were also used for other tasks, leading to misclassification, while *Do not*

use ML API 2 false negatives (0.59%). Additional false negatives were found in *Both training and inference in the same file* (2 cases, 0.59%) *Library not included in this work* (1 case, 0.29%), and *Training Keyword not used for training* (1 case, 0.29%). Altogether, the classification of *'ML-Model Consumer'* projects led to 10 false positives (2.93%) and 25 false negatives (7.3%), for a total of 35 (10.23%) misclassification cases.

These results reveal that the majority of misclassifications were linked to the *'Custom Solution'* category for both *'ML-Model Consumers'* and *'ML-Model Consumers,'* with a higher number of cases observed in *'ML-Model Consumers'* (35) compared to *'ML-Model Consumers'* (14). The rare instances where MARK misclassified projects highlight specific edge cases that challenge the current heuristic-based approach. These findings highlight the opportunity for further refinements of this approach, including the integration of additional libraries and heuristics to effectively address these unique challenges and enhance the overall robustness of the classification process.

Table 5.6: Statistical Metrics for *'ML-Model Producers'* and *'ML-Model Consumers'* with different Knowledge Base and Rules (R) configurations of MARK. The symbol 🔒 indicates essential rules for classification.

Knowledge Base	R#1	R#2	R#3	R#4	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
MARK *'ML-Model Producers'* Configuration: Knowledge Base								
Nguyen et al.	🔒	🔒	-	-	95.01	99.57	93.60	96.49
Our dataset (MARK)	🔒	🔒	-	-	**95.89** (+0.88)	**99.58** (+0.01)	**94.80** (+1.20)	**97.13** (+0.64)
MARK *'ML-Model Consumers'* Configuration: Knowledge Base								
Nguyen et al.	🔒	🔒	✗	✗	74.78	47.89	85.00	61.26
Our dataset	🔒	🔒	✗	✗	75.66 (+0.88)	48.99 (+1.11)	91.25 (+6.25)	63.76 (+2.50)
MARK *'ML-Model Consumers'* Configuration: Rules								
Our dataset	🔒	🔒	✗	✗	75.66	48.99	91.25	63.76
Our dataset	🔒	🔒	✓	✗	86.80 (+11.14)	71.60 (+22.61)	72.50 (-18.75)	72.05 (+8.29)
Our dataset	🔒	🔒	✗	✓	76.54 (+0.88)	50.00 (+1.01)	88.75 (-2.50)	63.96 (+0.20)
Our dataset (MARK)	🔒	🔒	✓	✓	**89.74** (+14.08)	**84.62** (+35.63)	**68.75** (-22.50)	**75.86** (+12.10)

5.4.3.2 *MARK Configurations*

When it turns to the impact of each component on MARK, the results are reported in Table 5.6. The table presents the performance metrics—accuracy, precision, recall, and F1-score—under different configurations

of MARK. In each section of the table, i.e., *"MARK 'ML-Model Producers' Configuration: Knowledge Base',"* *"MARK 'ML-Model Consumers' Configuration: Knowledge Base,'* and *"MARK 'ML-Model Consumers' Configuration: Rules'* the first row represents the baseline configuration against which subsequent configurations are compared, allowing us to measure the incremental impact of each modification.

First, considering the impact of the knowledge base, I observed slight improvements in the performance metrics for both the classification of *'ML-Model Producers'* and *'ML-Model Consumers.'* Specifically, the enhanced knowledge base developed in this study, compared to the original knowledge base from Nguyen et al. [256], resulted in marginal increases across accuracy, precision, recall, and F1-score. For instance, in the case of *'ML-Model Consumers,'* recall improved by +1.20%, precision remained stable with a minor increase of +0.01%, and the F1-score rose by +0.64%. Similarly, for *'ML-Model Producers,'* I observed comparable trends, with slight yet consistent improvements in classification performance.

First, the results indicate that the enhanced knowledge base provided a more comprehensive and updated representation of relevant ML libraries and APIs, which reduced misclassifications. The improved recall demonstrates that the enhanced knowledge base enabled MARK to identify a slightly higher number of true positive classifications for producers and consumers. This improvement can be attributed to the inclusion of more recent and diverse ML components absent in the original knowledge base, thereby justifying the effort invested in creating the enhanced version. Although the scale of the observed improvements may be considered modest at first sight, these findings remain significant for two main reasons. First, the enhancements highlight the importance of periodically updating the knowledge base to capture the evolving landscape of ML tools and practices. Second, even incremental improvements in recall and F1-score enhance the overall robustness of the classification process, which is particularly valuable in scenarios where identifying as many relevant projects as possible is crucial. As such, these results highlight the practical value of the enhanced knowledge base in maintaining and extending MARK's

long-term utility. At the same time, the relatively minor scale of these improvements can be attributed to the fact that the foundational rules (Rules #1 and #2) already capture the most common and well-established ML libraries and APIs. As a result, the original knowledge base may have already addressed a significant portion of the classification needs. On the one hand, this highlights the value of the previous efforts by Nguyen et al. [256], as their knowledge base effectively captured essential elements of the ML ecosystem that are still relevant nowadays. On the other hand, this result provides insights into the long-term maintainability of the knowledge base. These findings show that updating the knowledge base nearly five years after its initial inception led to only slight improvements in classification performance. This suggests that the most well-established ML libraries and APIs remain relevant over extended periods, with relatively few new additions being widely adopted. This observation implies that frequent updates to the knowledge base may not be necessary, as its foundational content continues to support effective classification. Consequently, the enhanced knowledge base emerges as a relatively long-lasting contribution that researchers can rely on without needing constant revisions.

In conclusion, these results provide additional value to the enhanced knowledge base, as they suggest it can serve as a stable and scalable resource for future studies. By requiring only periodic updates to incorporate major developments or new trends in ML libraries, the enhanced knowledge base offers researchers a reliable tool for analyzing ML projects while minimizing the maintenance overhead. Moreover, its longevity enhances its practical utility for a wide range of applications, further solidifying its relevance in the evolving field of ML project classification.

When it turns to the impact of the classification rules used to classify *'ML-Model Consumers,'* the results suggest that the inclusion of Rules #3 and #4 provide incremental stability to the classification capabilities of the approach. Specifically, the MARK version without Rule #3 achieves high recall (91.25%) but suffers from low precision (48.99%), indicating a significant number of false positives in the classification. The introduction of Rule #3 substantially improves precision to 71.60% (+22.61%) while

reducing recall to 72.50% (-18.75%). These results suggest the validity of the assumption that files containing both training and inference APIs are more likely to leverage the latter for evaluation purposes rather than genuine inference. By excluding such files from the consumer classification, Rule #3 effectively reduces misclassifications, particularly those resulting from overlapping producer-consumer characteristics. This trade-off between precision and recall reflects the heuristic's emphasis on minimizing false positives, which aligns with MARK's primary goal of ensuring the reliability and trustworthiness of its classifications. While recall decreases, the overall improvement in F1-score (+8.29%) demonstrates that Rule #3 achieves a better balance between the two metrics, reinforcing its role in enhancing the robustness of the classification process.

As for Rule #4, this is the rule that may have the most significant impact on false negatives, as its primary function is to exclude files that contain terms indicative of testing or evaluation (e.g., *"test," "eval,"* or *"validate"*) from being classified as consumers. While this exclusion reduces false positives by ensuring that testing-related files are not mistakenly identified as consumer files, it can inadvertently lead to false negatives in cases where such files genuinely serve inference purposes but happen to include these terms in their filenames. When analyzing the results shown in Table 5.6, I observe that Rule #4 slightly improves precision compared to the version of MARK that includes only Rules #1 and #2 (50%, +1.01%) but marginally decreases recall (88.75%, -2.50%). Despite these modest changes, the application of Rule #4 results in an improved F1-score (63.96%, +0.20%), indicating a slightly better balance between precision and recall. This suggests that Rule #4, while not drastically altering overall performance, plays a dual role: it reduces false positives by excluding files primarily dedicated to testing or evaluation but also slightly increases false negatives compared to the MARK version solely including Rules #1 and #2.

Finally, assessing the final version of MARK, which includes both Rules #3 and #4 alongside Rules #1 and #2 for the classification of *'ML-Model Consumers,'* I observed significant improvements in precision (84.62%, an increase of +35.63%) but also a notable reduction in recall (68.75%, a

decrease of -22.50%) compared to the version of MARK that excludes Rules #3 and #4. This finding leads to two important conclusions that ultimately justify the design decisions to implement the approach. First, including all rules results in the best possible balance between precision and recall, as demonstrated by the highest F1-score (75.86%) achieved among all evaluated configurations. This suggests that the combined application of Rules #3 and #4 provides a robust framework for accurately distinguishing *'ML-Model Consumers'* by addressing false positives and false negatives holistically. Second, and more critically, the final version of MARK prioritizes maximizing precision, aligning with its intended design goal. As discussed earlier in this section, MARK is designed to support researchers in selecting a subset of ML projects for further analysis, particularly in scenarios where ensuring the correctness of the classification is paramount. High precision ensures that the identified consumer projects are highly reliable, minimizing the risk of including irrelevant or misclassified projects. By achieving a precision of 84.62%, the final version of MARK effectively minimizes misclassifications, which is crucial for studies requiring high-quality curated datasets. Although the notable decrease in recall indicates that some relevant consumer files are not identified, I deem this trade-off acceptable in the context of MARK's primary objectives. The ability to confidently classify projects as consumers outweighs the need to capture every relevant file, particularly when a representative sample is sufficient for most research applications.

5.4.3.3 *Assessment of 'ML-Model Producer & Consumer' Classification*

After collecting the results of the performance classification in individual projects, I evaluated the ability of MARK to assess its effectiveness in finding *'ML-Model Producer & Consumer'* projects. This category combines elements that characterize a project as a *'ML-Model Producer'* and a *'ML-Model Consumer.'* Table 5.7 shows the confusion matrix for detecting this category. Moreover, Table 5.8 summarizes the performance of the approach in detecting this category. Combining the information provided by the confusion matrix and the overall performance metrics, some

considerations emerge. An accuracy of 90.62% demonstrates the strong overall reliability of the tool in identifying this category. The precision of 83.93% indicates that most positive classifications are correct, showing that the tool effectively identifies relevant projects while maintaining a low false positive rate. According to the confusion matrix, nine instances were misclassified as part of the *'ML-Model Producer & Consumer'* category. Additionally, the recall of 67.14% reflects the tool's ability to identify a significant portion of actual *'ML-Model Producer & Consumer'* projects, with 47 true positives and 23 false negatives. The recall score is strictly limited by applying the rules to detect *'ML-Model Consumer'* (i. e., Rule #3 and Rule #4), as also denoted by similar scores. However, the F1-Score of 74.60% shows a balance between precision and recall, highlighting the tool's ability to identify *'ML-Model Producer & Consumer'* while offering opportunities for continued improvements. In summary, the performance of MARK in classifying *'ML-Model Producer & Consumer'* is closely related to the tool's ability to accurately detect individual categories of *'ML-Model Producer'* and *'ML-Model Consumer.'* In fact, the results for the new category are well-aligned with the detection performance of the individual categories, reflecting the underlying consistency and effectiveness of the classification approach across individual and combined project categories. Therefore, the room for improvement observed in the detection of single categories is directly related to the misclassification of these projects in the *'ML-Model Producer & Consumer'* category. Enhancing the precision and recall for the *'ML-Model Producer'* and *'ML-Model Consumer'* projects will naturally lead to improvements in the classification of the combined *'ML-Model Producer & Consumer'* category.

RQ$_1$ — Projects Classification

MARK could identify 'ML-Model Producer' with a precision of 99.89% and a recall 94.80%. However, the performance is slightly lower for the 'ML-Model Consumer' classification, which shows a precision of 84.62% and a recall of 68.75 %. Moreover, this approach can also classify the 'ML-Model Producer & Consumer' category with 83.93% of

Table 5.7: Confusion Matrix for *'ML-Model Producer & Consumer'*

		Predicted	
		Negative	Positive
Actual	Negative	262	9
	Positive	23	47

Table 5.8: Performance Metrics for *'ML-Model Producer & Consumer'*

Metric	*ML-Model Producers & Consumers*
Accuracy	90.62 %
Precision	83.93 %
Recall	67.14 %
F1-Score	74.60 %

> *precision and 67.14% of recall. Challenges in the classification process arise from custom implementations, non-standard APIs, and testing-related terms, which may contribute to misclassifications. Despite these limitations, the high precision aligns with MARK's primary goal of providing reliable classifications for research purposes. The detailed analysis highlights opportunities for refinement, including the expansion of the knowledge base and improved heuristics to address edge cases. Overall, MARK effectively supports its intended research-oriented use case, ensuring accurate and representative datasets for further analysis.*

5.5 RQ_2. EMPIRICAL STUDY: ON THE USEFULNESS OF MARK

The *goal* of the study was to understand the socio-technical characteristics of ML projects classified according to the schema, with the *purpose* of gaining insights into how the differences among these projects validate the usefulness of the classification. The *perspective* is of researchers who are interested in exploring the distinct patterns, challenges, and opportunities that emerge from different types of ML projects, ultimately aiming to refine best practices and improve the effectiveness of tools and methodologies in the SE4AI domain.

Table 5.9: Distribution of Projects by Category.

Category	Description	# Projects
ML Libraries & Toolkits	Projects that provide tools to develop ML solutions.	1,349
ML-Model Producers	Projects that invoke a training function.	1,359
ML-Model Consumers	Projects using ML models without building them.	56
ML-Model Producers & Consumers	Projects that simultaneously produce and utilize ML models within the same context.	493

5.5.1 *Data Collection*

The first research question evaluated MARK's ability to distinguish between *'ML-Model Producers'* and *'ML-Model Consumers'*. This evaluation was essential to validate the core functionality of MARK in correctly categorizing projects using machine learning models. However, the second empirical study expanded the scope to include all four project classes: *'ML-Model Producers,' 'ML-Model Consumers,' 'ML-Model Producers & Consumers'* and *'ML Libraries & Toolkits'*. This broader analysis enabled us to evaluate the overall effectiveness of integrating MARK with the original classification by Gonzalez et al. [136] as a research tool. It also allowed us to explore and compare the significant differences across all project categories, further demonstrating the utility of this combined classification approach. To achieve this goal, I essentially simulated a researcher's activities when conducting an empirical study comparing the characteristics of different ML projects. I began by using the datasets employed in **RQ**$_1$, specifically (1) the dataset initially proposed by Gonzalez et al. [136] and later refined by Rzig et al. [295], and (2) NICHE [364]. From these datasets, I applied MARK to classify projects labeled as *'Applied ML'* while retaining those labeled as *'ML Libraries & Toolkits'* as a separate category. Using MARK, I further identified projects related to *'ML-Model Producers'* and *'ML-Model Consumers'*. Subsequently, I added a step to identify projects classified in both *'ML-Model Producers'* and *'ML-Model Consumers,'* cat-

egorizing them as *'ML-Model Producers & Consumers'*. The final set of projects classified by MARK is in Table 5.9, which shows the distribution of projects among the four categories.

Afterwards, I computed repository metrics, following previous works [6, 78, 250]. These metrics described projects along multiple dimensions, characterizing the *socio-technical* dynamics arising in those communities. I collected data from these categorized ML projects using the GitHub API [128]. Table 5.10 details the metrics collected.

Firstly, I focused on **engagement and collaboration** metrics, including the *number of stars* and *number of forks*, which highlight user interaction levels. I also collected the *number of contributors* to gain insights into the collaborative nature of the projects and the extent of community engagement. Moreover, the data collection covered **activity metrics**, essential for evaluating each repository's activity rate and development efforts. Specifically, I gathered data on the *number of commits* and the *activity rate*, measured by the frequency of commits over time. Lastly, I included **structural metrics** to assess the overall complexity and maturity of the projects. I measured each repository's *size* regarding lines of code (LOC), which helps gauge the project's scale and potential maintenance demands. Additionally, I considered the *project age*, providing context on the project's maturity.

As already discussed in Section 5.3, it is worth remarking that this analysis is not intended to offer an exhaustive overview of the classification's applications but rather to provide insights into how it can be leveraged by researchers. The focus on project dynamics and community engagement stems from the availability of well-established metrics and tools for their measurement. Identifying any differences would validate the value of classifying projects based on their nature and functionality, thereby underscoring the potential benefits for researchers interested in studying the characteristics of projects within the ML Universe.

Table 5.10: Metrics selected for analyzing community engagement, activity, and collaboration aspects of ML projects.

Metric Categories	
Engagement and Collaboration Metrics	**Number of Forks:** The number of times the repository has been forked by other users.
	Number of Stars: The number of stars the repository has received from users.
	Number of Contributors: The number of contributors to the repository.
Activity Metrics	**Number of Commits:** The number of commits of the repository.
	Activity Rate: The rate of activity measured as the frequency of commits over time.
Structural Metrics	**Project Age:** The amount of time passed since the creation date of the project.
	Size: The size of the repository expressed in lines of code (LOC).

5.5.2 *Data Analysis*

The collected metrics were then used to evaluate whether a specific metric can be distinctly associated with a single ML category. Therefore, starting from the analyzed ML projects, I aimed to statistically analyze the differences between the categories.
Specifically, considering Table 5.10, let $X = \{\#Forks, \#Stars, \#Contributors, \#Commits, ActivityRate, ProjectAge, Size\}$ the set of metrics, I formulated the following null hypotheses for each of the proposed metrics:

H0: *For each $x \in X$, there is no statistically significant difference in terms of x between 'ML-Model Producers,' 'ML-Model Consumers,' 'ML-Model Producers & Consumers,'* and *'ML Libraries & Toolkits.'*

This leads to the following alternative hypothesis, stating that for each dependent variable, a statistically relevant difference can be observed between categories:

HA: *For each $x \in X$, there is a statistically significant difference in terms of x between 'ML-Model Producers,' 'ML-Model Consumers,''ML-Model Producers & Consumers,'* and *'ML Libraries & Toolkits.'*

Before conducting any statistical tests, I first assessed the normality of the data to determine the appropriate statistical methods. I began by evaluating the properties of each distribution using the Shapiro-Wilk test [133] with a significance level of $\alpha = 0.05$, applied to each project category. This test helped us assess whether the data followed a normal distribution. If the data had been normally distributed, I would have employed parametric tests such as ANOVA. However, since the Shapiro-Wilk test revealed that the data did not conform to a normal distribution, I opted for non-parametric tests better suited to the data's characteristics. Specifically, I applied the Kruskal-Wallis H Test [72], a non-parametric method, to compare the medians across the four groups. This test is robust to the assumptions of normality homogeneity and provides an initial assessment of significant differences between the groups. Upon detecting significant differences using the Kruskal-Wallis H Test, I conducted pairwise Mann-Whitney U tests [243] with Holm-Bonferroni correction [160] to control for Type I errors. These pairwise comparisons allowed us to identify which specific ML project categories exhibit statistically significant differences from each other.

5.5.3 *Analysis of the Results*

Before analyzing the outcomes of the repository metrics analysis, I examined the prevalence of ML libraries in the project categories. Figure 5.5 shows the top 10 ML Libraries used respectively in *'ML-Model Producers'* and *'ML-Model Consumers.'* Common outcomes between ML project categories show the usage of common libraries, which is hugely different from other libraries. Specifically, "tensorflow" is the most used library for both the ML projects categories, 811 *'ML-Model Producers'* and 271 in *'ML-Model Consumers'* projects. This discrepancy between the most used libraries and others highlights the tendency of ML practitioners to use the

most popular libraries, underrating the potentiality of other solutions. This phenomenon might be due "popularity bias" [2]. On the one hand, the main differences highlight the use of "sklearn" library in *'ML-Model Producer'* projects aimed specifically at producing ML models. On the other hand, "keras" library is more used than "sklearn" in *'ML-Model Consumer'* projects. Finally, other libraries that allow the training and use of different kind of ML models are highlighted in this analysis, such as "gym" library with 40 references in *'ML-Model Producer'* projects and "transformers" library, with 27 references in *'ML-Model Producer'* projects and nine in *'ML-Model Consumer'* Projects.

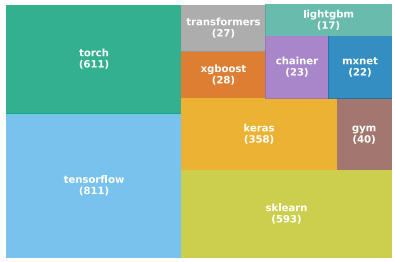


Figure 5.3: Top 10 ML Libraries used in *ML-Model Producers'* projects.

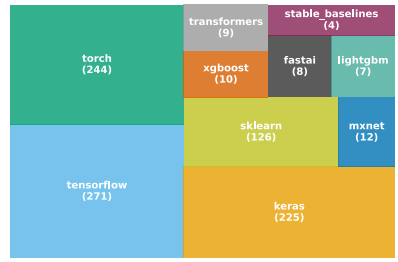


Figure 5.4: Top 10 ML Libraries used in *ML-Model Consumers'* projects.

Figure 5.5: Comparison of the top 10 ML libraries used by ML-Model Producers' and ML-Model Consumers'.

HYPOTHESIS TESTING Several outcomes are enlighted when analyzing the community engagement, activity, and collaboration metrics. Table 5.11 shows the results of the Kruskal-Wallis H test applied for all the metrics. Additionally, results of Mann-Whitney U test post-hoc analysis are reported to identify statistically significant differences between ML project categories. Observing the results of the Friedman Test, all the metrics report a *p-value* lower than 0.005, leading to the rejection of the formulated null hypotheses and concluding that there are significant differences between the three categories. The *number of stars* showed significant differences in several groups, such as {MP, LT} ($p < 0.001$) and {MPC, MP}

($p < 0.001$). In contrast, the difference between {MP, MC} (1) and {MPC, LT} (1) were not statistically significant. For the *size*, significant differences were observed in groups {MPC, MP} ($p < 0.001$), {MPC, MC} ($p = 0.03$), and {MPC, LT} ($p < 0.001$). However, groups such as {MC, LT} ($p = 1$) and {MP, MC} ($p = 1$) indicated there are no statistical differences. The *number of forks* followed a similar pattern, with significant differences between {MP, LT} ($p < 0.001$) and {MPC, MP} ($p < 0.001$). Nonetheless, the differences between {MP, MC} ($p = 0.74$) and {MPC, LT} ($p = 0.40$) and other groups were not statistically different. The *number of commits* exhibited significant differences in most groups such as {MP, LT} ($p < 0.001$) and {MPC, MP} ($p < 0.001$). However, the group {MP, MC} ($p = 0.87$) does not show significant statistical variations. For *project age*, significant differences were found in {MP, LT} ($p < 0.001$) and {MPC, LT} ($p < 0.001$). In contrast, the other groups, such as {MPC, MP} ($p = 1$) and {MPC, MC} ($p = 1$) demonstrated no statistical differences. The *activity rate* showed notable differences in most of the groups, such as {MP, LT} ($p < 0.001$) and {MPC, MP} ($p < 0.001$). However, the group {MP, MC} ($p = 0.82$), revealed no significant variation. Finally, the *number of contributors* exhibited significant differences in almost all groups, such as {MP, LT} ($p < 0.001$) and {MPC, MP} ($p < 0.001$). At the same time, there are no statistical differences in the group {MPC, MC} ($p = 0.82$). As a main outcome of the tests, the classification identified project categories with many characteristics for each group, each category of which could be associated with a repository metric. While many metrics displayed statistically significant differences across the groups, several comparisons revealed little or no variation, highlighting the diversity of certain project characteristics.

STATISTICAL ANALYSIS To understand category differences, I observed their distribution, represented in Figure 5.6. MP and MC appear to have lower metric values than the two categories. Specifically, examining the distribution of stars across each ML project category, MP projects show fewer stars, with a median value of 93, compared to LT projects (210) and MPC projects (205). This difference was previously found to be statistically

Table 5.11: Kruskal-Wallis H and Mann-Whitney U tests. Number(#), ML-Model Producer (MP), ML-Model Consumer (MC), ML-Library & Toolkit (LT), ML-Model Producers & Consumers (MPC).

Categories	Community engagement Metrics			Activity Metrics			Collaboration Metrics
Tests	# Stars	Size	# Forks	# Commits	Project Age	Activity Rate	Contributors
Kruskal-Wallis	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001
Mann-Whitney {MP, MC}	1	1	0.74	0.87	1	0.82	0.008
Mann-Whitney {MP, LT}	<0.001	0.07	<0.001	<0.001	<0.001	<0.001	<0.001
Mann-Whitney {MC, LT}	0.02	1	0.1	<0.001	0.1	<0.001	0.003
Mann-Whitney {MPC, MP}	<0.001	<0.001	<0.001	<0.001	1	<0.001	<0.001
Mann-Whitney {MPC, MC}	0.02	0.03	0.03	<0.001	1	0.001	0.82
Mann-Whitney {MPC, LT}	1	<0.001	0.4	<0.001	<0.001	<0.001	<0.001

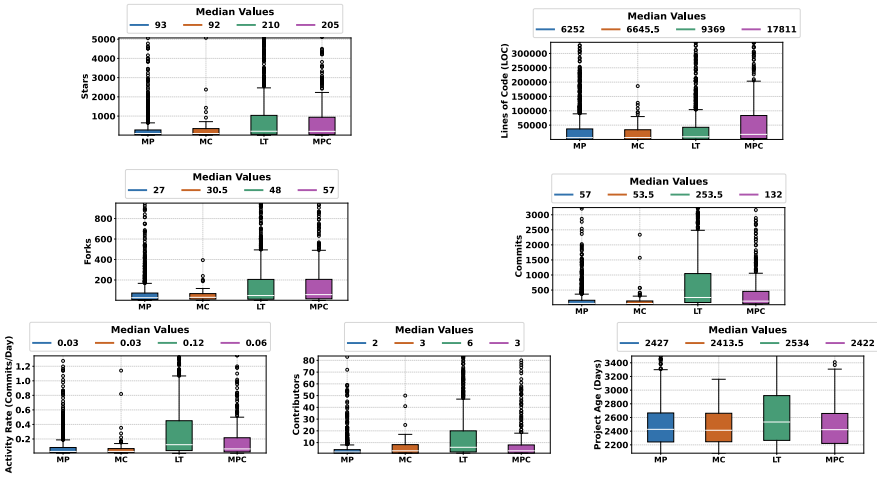


Figure 5.6: Distribution of the categories of ML projects among the selected metrics. The categories are ML-Model Producer (MP), ML-Model Consumer (MC), ML-Library & Toolkit (LT), ML-Model Producers & Consumers (MPC).

significant. In contrast, MC projects have a similar star distribution to MP, with a median value of 92. Overall, LT and MPC categories showed higher star counts compared to MP and MC. Statistically significant differences were also observed in project size, measured by LOC. MPC projects exhibited the largest size, with a median LOC of 17,811. LT projects followed with a median LOC of 9,369, notably higher than MP (median: 6,252) and MC (median: 6,645.5). Regarding the number of forks, MPC projects had the highest median value of 57. LT projects reported a slightly lower median of 48 forks. Meanwhile, the MP and MC categories showed fewer forks, with median values of 27 and 30.5, respectively. LT projects also reported the highest number of commits, with a median value of 253.5. In contrast, MPC projects had a lower median of 132 commits, while MP and MC categories displayed notably lower values, with medians of 57 and 53.5, respectively. The differences between MP and MC in this aspect are relatively small. A similar trend was observed for activity rate and the number of contributors, where LT and MPC projects generally outperformed MP and MC projects. Finally, regarding project age (measured as days since the starting date), LT projects reported the highest median value of 2,534 days. However, the differences between categories are minor, as the distributions are quite similar.

> **RQ_2 — Projects Characterizations**
>
> *Significant differences emerge across the project categories. Projects in the 'ML-Model Producers & Consumers' and 'ML Library & Toolkit' categories are characterized by larger environments, with higher median values for stars, size (LOC), forks, and commits. In contrast, 'ML-Model Producers' projects are notably smaller and exhibit distinct differences, particularly in their lower median values across most metrics.*

5.6 DISCUSSION AND IMPLICATIONS

This study highlights several implications for the SE4AI community, quantifying and shedding light on SE researchers' challenges and opportunities.

ON THE VALUE OF MARK RQ_1 shows that an automated classification is feasible and may reach good performance already with a heuristic approach, hence representing a valuable addition to the current state of the art [136, 364]. The automation of ML project classification addresses several key challenges in the field of SE4AI. One of the most critical issues is the substantial time and effort required for manual classification, especially when dealing with large datasets. Manual classification is a resource-intensive process that demands specialized domain knowledge and attention to detail, which increases the risk of errors and inconsistencies. These misclassifications can hinder the accuracy of subsequent analyses and limit the ability to draw meaningful conclusions from the data [40, 295]. By automating the classification process, this approach mitigates these challenges and enhances the quality and reliability of research outcomes. From a practical perspective, this automation empowers SE4AI researchers to conduct more precise and context-aware analyses of ML projects. For instance, by distinguishing among *'ML-Model Producer,'* *'ML-Model Consumer,'* and *'ML-Model Producer and Consumer,'* the classification framework allows researchers to tailor their studies to specific types of ML projects, leading to more relevant and actionable insights. For example, those focused on developing ML models might prioritize *'ML-Model Producers,'* those interested in applying and integrating pre-trained models could concentrate on *'ML-Model Consumers.'* In contrast, those aiming at exploring the full lifecycle of ML pipelines may focus on may focus on *'ML-Model Producer and Consumer.'* This represents a key transversal aspect that can significantly benefit the research community. By enabling comparative studies that analyze differences in how various types of ML projects operate—whether in their development processes, model deployment strategies, or maintenance practices—researchers can

uncover critical insights that were previously overlooked. By accurately selecting and focusing on the right project samples, researchers can drive the creation of tailored best practices, tools, and methodologies specifically designed to meet the unique needs of each project type. This, in turn, will contribute to developing more effective and efficient ML systems across the board.

> **Implication #1.** This classification approach has the potential to expand the scope of research in the field of SE4AI, possibly enabling new avenues of inquiry, such as more targeted studies on ML project types, refined tool development, and the creation of best practices tailored to specific project characteristics.

ON THE CHALLENGES OF CLASSIFYING ML PROJECTS Building on the findings of RQ_1, I argue that there are still some challenges for properly classifying ML projects. This is especially true in the case of *'ML-Model Consumer'*. Specifically, this study highlights the need to refine and expand the detection mechanism to cover a wider range of projects, finally improving its recall. In this respect, two considerations must be pointed out. The assessment of various configurations of MARK revealed the intricate trade-offs between precision and recall. For example, Rule #3 significantly improved precision by minimizing false positives but reduced recall by excluding valid instances. This trade-off highlights the challenge of achieving a balance between these metrics. Nonetheless, the high precision achieved by the final configuration of MARK is a significant asset for SE4AI research. Precision often holds greater importance in this domain than recall, as it ensures that the projects identified are reliable and accurate. By minimizing false positives, the classification provides a trustworthy foundation for subsequent research, enhancing the quality and reliability of findings and ensuring that the results can be confidently applied in practice. Similarly, these findings suggest that the final configuration of MARK strikes the best compromise between precision and recall while achieving the highest precision among all evaluated configurations. This optimal balance ensures that the approach aligns with its primary objective: providing a depend-

able tool for research that minimizes noise and misclassification, thereby enhancing the quality of downstream analyses. While improving recall remains a goal for future work, the current precision-focused approach may already effectively support research requiring high-quality datasets, where inclusivity is less critical than reliability.

At the same time, these analyses identified some limitations that warrant further research. In particular, a significant challenge lies in accurately distinguishing between projects that consume ML models and those that include similar functionalities for testing or auxiliary purposes. This issue often arises because current heuristics may not fully capture the contextual usage of APIs within the codebase. Two potential improvements can be considered to address this. First, adopting a *contextual analysis* approach: instead of relying solely on API calls, incorporating a deeper analysis of the files within the project—examining their purpose, interactions, and overall context—could significantly enhance classification accuracy. Second, leveraging *advanced AI-based solutions*, e.g., NLP techniques, could improve the analysis of comments, documentation, and code artifacts, thereby improving the accuracy of project classification. For instance, natural language processing techniques, such as pre-trained models like BERT or fine-tuned large language models, could be employed to disambiguate whether a file containing both producer and consumer APIs is used for evaluation or inference, hence contributing to increasing the overall classification accuracy of this approach. Similarly, prompt engineering could enable large language models to classify ML projects more effectively by leveraging context-aware insights from file descriptions. While promising, these approaches may require significant computational resources and further experimentation to assess their scalability and practicality.

Implication #2. Further research is required to address the inherent challenges of classifying ML projects. Possible solutions involve contextual analysis and AI-based solutions to differentiate *'ML-Model Consumers'* from *'ML-Model Producers'*.

ON THE USE OF THE CLASSIFICATION IN THE WILD When it turns to the second research question, I could first conclude that projects within the *'ML-Library & Toolkit,' 'ML-Model Producers,' 'ML-Model Consumers,'* and *'ML-Model Producers & Consumers'* categories exhibit significant differences in terms of repository metrics.

From a conceptual standpoint, the differences observed confirm the usefulness of the proposed classification and the overall significance of considering the nature of ML projects when performing empirical studies. Building on previous insights, the results underscore that differentiating ML projects is crucial for accurately investigating the structure, organization, and team dynamics of these projects. Similarly, the findings highlight the importance of tailored approaches when studying or developing tools for these projects. While the aim was not to exhaustively analyze all differences among project types, it is reasonable to expect that significant variations could also be observed in other areas, such as collaboration patterns, code quality metrics, or scalability concerns. As such, the findings represent a call to researchers in the field to explore these differences further, deepening the understanding of the unique challenges and opportunities presented by each type of ML project.

From a practical standpoint, the findings provide valuable insights into the socio-technical dynamics of ML projects, highlighting the distinct requirements and characteristics of different project categories. For instance, high levels of community engagement often correlate with strong support networks, essential for fostering innovation and sustaining project momentum. Similarly, activity levels within a project indicate its vitality and responsiveness to emerging developments, while patterns of contributor collaboration reflect the project's inclusivity and openness to diverse perspectives. The observed differences suggest varying maturity levels, impacting stability, reliability, and feature set comprehensiveness. Specifically, the analysis shows that different ML project categories have varying socio-technical requirements. *'ML-Library & Toolkit'* projects tend to have more commits and slightly more contributors, indicating a higher level of collaboration and a potentially more complex development process.

These projects may benefit from robust communication and coordination tools to manage contributions effectively alongside more stringent quality assurance processes to maintain the integrity and performance of integrated models. Additionally, *'ML-Model Producers & Consumers'* tend to exhibit higher lines of code and forks, reflecting their dual-purpose nature and the inherent complexity of combining model development with the deployment and distribution. The larger line of code suggests intricate pipelines and diverse functionalities, while the higher number of forks highlights their widespread use and adaptation by the community. These projects require technical expertise and collaborative practices to manage the full lifecycle of ML pipelines effectively. Conversely, *'ML-Model Producer'* projects, with fewer contributors, may prioritize the depth of expertise and specialized skills in model development. In these cases, quality assurance may focus more on the accuracy and reliability of the models rather than broader collaborative processes. This differentiation is crucial for researchers when prioritizing efforts, as newer projects might be ideal for cutting-edge research. In comparison, older projects could be more suited for stable, long-term deployments.

The findings of this study may be instrumental for researchers in social software engineering, who are interested in understanding community dynamics and collaboration patterns, as well as for those working on software maintenance and evolution, who are focused on project longevity, stability, and the impact of maturity on software quality. In addition, given that socio-technical dynamics play a crucial role in shaping software quality [204, 265], the differences observed in this study suggest that these dynamics likely extend to the technical domain, potentially influencing the quality assurance practices employed across different ML projects. As such, these findings may stimulate additional research on the process and product quality of these systems.

Implication #3. The differences observed in socio-technical dynamics are likely to influence other research domains, extending beyond the scope of this study. These findings carry significant implications for researchers in areas like social software engineering, software maintenance and evolution, and software quality, providing them with valuable insights into the distinct characteristics of ML projects. These insights can be leveraged to understand better and address the unique challenges associated with these projects.

ON THE CLASSIFICATION AND BEYOND The classification builds on existing knowledge of training and inference APIs, aiming not to be a static resource but an evolving effort that requires ongoing research and updates. The periodic updates and refinements of the knowledge base are fundamental for maintaining its relevance and usefulness over time. The results show that even incremental updates to the KB can lead to modest but meaningful improvements in recall and F1-score, emphasizing the importance of periodically capturing the evolving landscape of ML tools and practices. This effort helps ensure that the classification remains robust and reflective of the current state of the field. I may keep the classification mechanism relevant and comprehensive by incorporating the latest advancements in ML APIs and analyzing emerging libraries that support cutting-edge models. Moreover, by making the study materials publicly available, I hope to foster collaboration within the research community. As periodic updates do not require substantial changes to the foundational rules and can focus on incremental refinements, this collaborative effort may ensure that MARK remains a scalable and long-lasting contribution to SE4AI research.

At the same time, I foresee future efforts extending the classification beyond the characterization of the currently supported classes of projects, i.e., *'ML-Model Producers,' 'ML-Model Consumers,'* and *'ML-Model Producers and Consumers.'* Although I did not systematically analyze the projects within each category, initial observations suggest differences that could lead to more refined classifications. Specifically, within *'ML-Model*

Producers' I identified variations in ML model architectures, ranging from traditional models like Decision Trees to more complex architectures such as CNNs. For instance, the MODEL-ZOO project [125] offers a variety of ML models, which the software engineering community could study to explore additional project categories and examine the differences in performance or non-functional requirements among different models. On the contrary, projects within *'ML-Model Consumer'* tend to be more task-oriented, such as those focused on computer vision, audio, and NLP, which could be further studied in terms of their integration with other software components or their operation in more heterogeneous environments. Furthermore, the results show the massive adoption of libraries such as TensorFlow, Keras, Sklearn, and Torch for ML projects. This outcome suggests that studies that do not consider this set of libraries and frameworks can accidentally exclude a large number of ML projects, decreasing the generalizability of their results for researchers and practitioners. In addition, qualitative and quantitative studies could be leveraged to refine the classification further by exploring the underlying motivations for adopting certain popular libraries and frameworks over others. Qualitative approaches, e.g., surveys and interviews, could provide insights into developer preferences and decision-making processes. Meanwhile, quantitative methods, e.g., mining software repositories, could analyze usage patterns and performance metrics across different projects. These studies could uncover trends and best practices, leading to context-aware classifications of the sub-types of projects within each category.

Implication #4. The classification approach is the preliminary step for defining continuous evolving activities to identify and classify ML projects, enlarging opportunities for SE4AI research.

PART III

LEVERAGING LARGE LANGUAGE MODELS TO SUPPORT SUSTAINABLE ML ENGINEERING

6

SUSTAINABLE PRACTICES IN ML-ENABLED SYSTEMS: AN LLM-BASED REPOSITORY MINING STUDY

6.1 INTRODUCTION

The growing use of ML systems raises concerns about their sustainability, particularly in terms of energy consumption and CO_2 emissions, have become increasingly pressing [96, 205]. The development and deployment of ML-enabled systems are resource-intensive, especially during training and inference [304]. For example, training models like LLaMA-65B consume around 449 MWh of electricity, producing approximately 173 tons of CO_2 emissions [337], raising concerns about their long-term sustainability, especially considering the increasing demand for powerful models.

While the focus has traditionally been on maximizing accuracy, there is increasing recognition of the need to balance this with environmental impact, leading to the concept of Green AI [304]. Green AI advocates for reducing computational demands while maintaining accuracy, emphasizing a responsible approach to AI development and deployment [347]. Maintaining this balance is essential to promote sustainable ML engineering and is important for society and users of ML systems, who are increasingly concerned about climate change. By addressing the environmental impact of ML-enabled systems, Green AI helps foster trust and align technological progress with broader societal goals of environmental responsibility and climate action [233].

In recent years, the software engineering community has made considerable efforts in promoting software sustainability, with specific attention now being paid to ML-enabled systems [235, 330, 350]. This shift has led to the identification and synthesis of various green tactics aimed at reducing the environmental footprint of ML models [182, 350]. For instance, Jär-

venpää et al. [182] recently elicited a comprehensive catalog of 30 green architectural tactics for ML-enabled systems, which were derived from a thorough analysis of the literature and validated through expert focus groups. These tactics encompass strategies across different phases of ML development and deployment, such as data-centric and management. A code example of a green architectural tactic is the quantization techniques provided by PyTorch[1], such as dynamic quantization shown below, to optimize ML models.

```
import torch
# create a model instance
model_fp32 = M()
# create a quantized model instance
model_int8 = torch.ao.quantization.quantize_dynamic(
    model_fp32,  # the original model
    {torch.nn.Linear},  # a set of layers to dynamically
    quantize
    dtype=torch.qint8)  # the target dtype for quantized weights
```

While the work by Järvenpää et al. [182] offers valuable guidance on Green AI, the actual relevance of these tactics is not yet established.

> ⚠ *There is still a gap of knowledge on the extent to which green tactics are actually implemented in real-world contexts. Moreover, it is unclear if there are additional practices not yet documented in the literature that practitioners employ when developing ML-enabled systems.*

An improved understanding of the practical adoption of green AI tactics is crucial for both researchers and practitioners. Researchers can gain insights into the relevance and impact of certain tactics, which can inform future studies and refinements of existing guidelines. Practitioners can benefit from knowledge of the most widely employed tactics in the wild, which can guide them in making sustainable choices. Furthermore, assessing the impact of the documented tactics will make previous findings more actionable and tangible, thereby enhancing their practical utility.

[1] https://pytorch.org/docs/stable/quantization.html

As highlighted in **Gap #4**, there is still a lack of analysis of sustainable practices in open-source repositories. To overcome the limitations of previous studies, I recently presented a preliminary paper [82] where I investigated the extent to which developers adopt green tactics defined in the literature when developing ML-enabled systems. Specifically, I build upon the catalog proposed by Järvenpää et al. [182], analyzing the adoption of these tactics in 168 open-source ML projects on GITHUB. To this aim, I developed a novel LLM-based mechanism to identify green architectural tactics in software repositories at the code level. This earlier work revealed that while developers implement many cataloged tactics, others remain underutilized due to knowledge gaps or implementation challenges. In this article, I build on this foundation by addressing two critical limitations of the earlier study.

- First, **I broaden the scope beyond the predefined catalog of tactics** by enhancing the LLM-based mechanism to identify undocumented green tactics employed by developers in real-world ML-enabled systems. This approach allowed us to **extend the existing catalog by introducing new tactics that were not previously documented**.

- Second, I augment the empirical findings of the preliminary study with additional **quantitative and qualitative insights supported by detailed examples of the newly identified tactics**.

- Furthermore, **I provide an extended and refined description of the methodology employed to investigate the adoption of green tactics**, enhancing the comprehensiveness of this analysis.

- Finally, I provided discussions and implications of this work for researchers and practitioners, **highlighting how the findings of this study may impact future developments in green software engineering**.

The results reveal that all existing tactics in the literature are adopted in real software projects, with a different frequency for each tactic. Additionally, starting by analyzing code rather than papers as previous work [182,

350], I was able to find nine new green tactics and expanded the catalog of green architectural tactics, providing the frequency of adoption of these tactics and a code example to implement them. These findings provide a deeper understanding of how practitioners improve sustainability in software engineering for ML-enabled systems. This work is important for AI/ML researchers and software engineers who want to improve the sustainability of AI development and reduce environmental impact, which is currently a clear industrial need as the proliferation of companies and industry initiatives such as Green Coding [2] and the Green Software Foundation [3] show, by helping clients, government agencies, and open-source communities to understand and optimize the carbon footprint of their digital services.

Data availability statement. For the sake of verifiability and replicability, I have made all the data and materials used in this study publicly available [237]. The appendix includes the original dataset of 168 ML projects, preprocessed data, and all scripts used for data collection, preparation, and analysis. Interested readers can use this material to assess the robustness/-completeness of the process and encourage further research on adopting and updating green architectural tactics in ML-enabled systems.

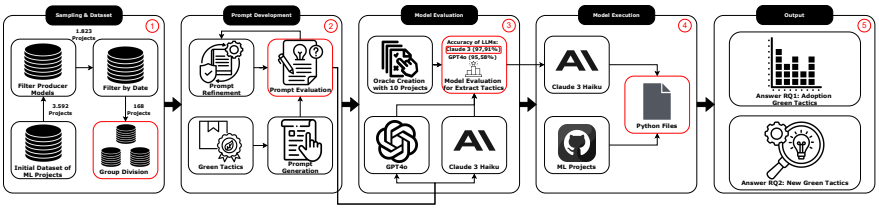


Figure 6.1: Overview of the research process.

6.2 RESEARCH GOAL AND QUESTIONS

To define the research goal, I follow the Goal-Question-Metric (GQM) approach [39]. The goal of this study is to *analyze* the adoption of green

[2] https://www.green-coding.io/
[3] https://greensoftware.foundation/

architectural tactics *with respect to* their frequency and possible catalog extension, from the *point of view* of researchers and practitioners, in the *context* of ML-enabled system development. More specifically, researchers seek to gain insights into the relevance and impact of green tactics in real software systems, which can inform future studies and guide the refinement of existing guidelines. For practitioners, understanding the most widely adopted tactics can help them implement sustainable practices in their projects, leading to reduced energy consumption, optimized system performance, and lower production costs.

To achieve the research objective, I focus on two main research questions (**RQ**s). First, $\mathbf{RQ}_1$ seeks to determine the extent to which the green tactics for ML-enabled systems, as catalogued by Järvenpää et al. [182], are adopted in the development of ML-enabled systems. Understanding this level of adoption is essential for bridging the gap between state-of-the-art research and the practices employed in real-world projects. Hence, I ask:

> **$\mathbf{RQ}_1$. Adoption of Green Tactics in Practice**
>
> *To what extent are green architectural tactics for ML-enabled systems adopted in software projects?*

Beyond assessing the adoption of existing tactics that as been addressed in the previous work [82], $\mathbf{RQ}_2$ aims to expand the catalog by uncovering new tactics observed in software repositories. These insights will deepen understanding and provide researchers and practitioners with valuable knowledge, contributing to the development of best practices and guidelines for sustainable software. In particular, I ask:

> **$\mathbf{RQ}_2$. Elicitation of Additional Green Tactics**
>
> *What additional green architectural tactics do developers adopt in building ML-enabled systems?*

To report the results of the exploratory mining study, I adhere to the standards for repository mining as outlined in the *"Repository Mining"* and *"General Standard"* categories of the *ACM/SIGSOFT Empirical Standards*.[4]

[4] https://github.com/acmsigsoft/EmpiricalStandards.

6.3 RESEARCH METHOD

This section describes the study with a quantitative ($\mathbf{RQ}_1$) and qualitative ($\mathbf{RQ}_2$) research approach to answer the **RQ**s. Figure 6.1 overviews the main steps conducted in the study.

6.3.1 *Sampling and Dataset Description*

I initially identified a candidate set of 3,592 ML projects coming from the combination of two popular datasets. The first dataset, proposed by Gonzalez *et al.* [136] and later revised by Rzig *et al.* [295], contains 3,020 ML projects. The second dataset, NICHE [364], includes 572 ML projects. Together, these datasets encompass a diverse array of ML projects, ranging from frameworks and tools to projects focused on creating ML models or integrating them into traditional software. However, not all projects within these datasets were suitable for the mining study. Specifically, the research aims to analyze the sustainable tactics used in the lifecycle of ML models. Therefore, I required to filter out projects that do not involve the training of ML models, as they would not provide relevant insights for the study. To achieve this, I applied two main filters to the combined dataset of 3,592 ML projects, refining it to a subset that aligns with the research objectives.

- **Selection of projects that produce ML models.** I selected projects specifically focused on building and training ML models, e.g., those that enable the training of neural networks for tasks like face detection and pose estimation. I identified these projects by analyzing their APIs, looking for the presence of import statements for ML libraries and the invocation of related training methods [136]. In this respect, I referred to a well-known and actively maintained list of AI & ML libraries [247], as previously done in literature [136]. After applying this criterion, the dataset was refined to include 1,823 ML projects.

- **Selection of active projects with recent commits (2024).** The 1,823 ML projects identified after the first filtering step were further refined based

on their activity levels. I used the GitHub API [128] to identify projects that had been updated in 2024, following the methodology of previous studies [24]. This search, conducted in May 2024, narrowed the dataset to 168 projects.

The final set comprises 168 projects. Table 6.1 summarizes the characteristics of these projects, including metrics such as age, number of stars, and number of contributors. Additionally, I use the lines of code (LOC) to classify projects into three size groups: small (LOC ≤ 10.000), medium (10.000 < LOC ≤ 100.000), and large (LOC > 100,000) . I use these groups to make comparisons in the analyses, which are in line with the methodology used by previous studies [24, 104]. After this classification, I created four groups: *'Small'* with 60 Projects, *'Medium'* with 59 Projects, *'Large'* with 49 Projects, and *'Total'* with 168 projects. Considering the group *'Total'*, the average project age is 6.63 years, with a median of 6.29 years. In terms of popularity, the projects have an average of 3,242.07 stars, with a median of 519.5 stars, indicating a wide range of visibility and adoption. The average project size is 144,818.84 LOC, with a median of 38,744.50 LOC, reflecting significant variability in the scale of the projects. On average, there are 14.41 contributors per project, with a median of 12 contributors. The average number of forks is 722.72, with a median of 122.5, suggesting that while some projects are highly replicated, others have a more modest reach. Finally, the average number of commits is 1,823.38, with a median of 674, highlighting varying levels of activity and development history among the projects. The selected projects show diverse metrics, which is satisfactory for the study's goals. This allows generalizable conclusions about the presence of green architectural tactics in different types of ML projects.

6.3.2 *Selection of Architectural Green Tactics*

After selecting the ML projects, I proceeded to identify the architectural tactics that would be the focus of the analysis. I chose to examine the green architectural tactics for ML-enabled systems defined by Järvenpää et al. [182], which build upon the foundational work of Verdecchia *et*

Table 6.1: Average (AVG) and Median (Med) of each group of projects.

Metric	Small	Medium	Large	Total
#Projects	60	59	49	168
Age (Avg)	6.30	6.67	6.98	6.63
Age (Med)	5.67	6.63	6.60	6.29
Stars (Avg)	1,101.68	2,543.78	6,703.76	3,242.07
Stars (Med)	255	669	563	519.5
Size (Avg)	3,466.75	46,739.27	435,995.41	144,817.84
Size (Med)	2,629.50	42,446	248,857	38,744.5
Contributors (Avg)	9.35	15.08	19.80	14.41
Contributors (Med)	4.5	13	21	12
Forks (Avg)	244.57	412.85	1,681.33	722.72
Forks (Med)	65.5	148	154	122.5
Commits (Avg)	532.63	1,114.46	4,257.47	1,823.38
Commits (Med)	291.5	691	1,506	674

al. [350]. This selection was based on the fact that the catalog provided by Järvenpää et al. [182] is, to my knowledge, the most comprehensive resource available, encompassing green practices across the entire ML pipeline. While other studies have addressed domain-specific aspects or isolated strategies [182, 248, 253], the catalog by Järvenpää et al. [182] organizes 30 green architectural tactics into six clusters, each targeting different aspects of energy efficiency and environmental sustainability within an ML pipeline. These clusters focus on various aspects of energy efficiency and environmental sustainability of a pipeline of ML-enabled systems.

From these candidate tactics, the study focuses on those pertaining to three specific clusters: *'Algorithm design'*, *'Model optimization'*, and *'Model training'*, encompassing a total of 15 tactics. These tactics are operational in nature, as they involve code-level and model-level adjustments that can be implemented using existing libraries or developed manually, depending on the developer's expertise. This flexibility ensures that the tactics are accessible to any development team, regardless of their resources, and can be effectively analyzed using the content available in GITHUB

repositories. While other tactics from the original study are undoubtedly valuable, they present significant challenges in terms of measurability and observability. For instance, tactics like *'T24-Use energy-efficient hardware'* and *'T25-Use power capping'* focus on hardware choices, which are not typically documented in code repositories. Similarly, *'T29-Retrain the model if needed'* involves long-term strategies that are rarely captured in GITHUB projects. Given that GITHUB repositories generally provide only the code, without detailed information about the underlying hardware or long-term operational plans, these aspects fall outside the scope of what can be effectively measured in the study. Consequently, the study focuses on the code in GITHUB ML projects, applying a targeted subset of the green tactics identified in the literature. Table 6.2 describes the selected tactics.

6.3.3 *Extraction of Architectural Green Tactics*

Detecting the presence of green architectural tactics in software repositories poses significant challenges, as it requires discerning whether developers have implemented specific practices to support environmental sustainability efforts. Developers rarely annotate code with explicit environmental sustainability intentions, making it difficult to discern whether a particular implementation reflects a green tactic. In exploring alternatives, I considered both manual and automated approaches. Manual inspection is largely impractical for the context because it is time-consuming, prone to errors, and requires a deep familiarity with diverse code languages. Additionally, recognizing a tactic may involve interpreting detailed code instructions or entire blocks, which can introduce ambiguity and inconsistency. Automated analysis, while more scalable, presents its own challenges. For example, let consider the case of the design pattern detection [241], where automated tools and algorithms are used to identify recurring structures in the code that correspond to known design patterns. This task is non-trivial because the same pattern can be implemented in multiple syntactic forms, and its recognition often requires an under-

Table 6.2: Green architectural tactics for ML-Enabled Systems selected in the scope of the work and the Large Language Model (LLM) detection rationale.

Tactic	Description	LLM Detection Rationale
T6. Choose an energy-efficient algorithm	Choose the most energy-efficient algorithm that achieves sufficient level of accuracy.	LLMs can recognize mentions of algorithm types and associate them with known energy profiles.
T7. Choose a lightweight algorithm alternative	If possible, choose lighter alternatives of existing algorithms.	LLMs are able to infer that 'lightweight' implies lower computational load, especially in model selection contexts.
T8. Decrease model complexity	Decrease the complexity of an ML model.	LLMs detect patterns indicating efforts to simplify model architecture (e.g., fewer layers, simpler structures).
T9. Consider reinforcement learning for energy efficiency	Use reinforcement learning to optimize energy efficiency at run time.	LLMs understand the concept of 'optimization' and can link RL applications to dynamic energy management strategies.
T10. Use dynamic parameter adaptation	Design parameters that are dynamically adapted based on the input data.	LLMs can detect descriptions of hyperparameter tuning that adapt based on input data, which aligns with energy efficiency goals.
T11. Use built-in library functions	Use built-in libraries for ML models if possible.	LLMs can identify common APIs or functions from well-known libraries (e.g., NumPy, PyTorch) and flag when they're used.
T12. Set energy consumption as a model constraint	Consider energy consumption as one predetermined parameter for optimizing the ML model.	LLMs interpret explicit mentions of constraints during training as optimization objectives, including energy constraints.
T13. Consider graph substitution	Replace energy-intensive model parts with similar, but less energy-consuming parts.	LLMs detect substitutions in model components that imply reduced resource usage, especially if semantically equivalent.
T14. Enhance model sparsity	Reduce the number of model parameters or set their values to zero.	LLMs recognize sparsity-related terminology and reduction in parameter counts, linking them to lower computation.
T15. Consider energy aware pruning	Prune neural networks starting from the most energy-intensive layer.	LLMs detect techniques such as 'pruning' and associate layer-level energy metrics with selective removal.
T16. Consider transfer learning	Use pre-trained ML models for other similar tasks.	LLMs can infer reuse from pretraining references and match it to the concept of transfer learning.
T17. Consider knowledge distillation	Use knowledge from a large ML model to train a smaller model.	LLMs identify 'teacher-student' language or simplification processes as indicators of knowledge distillation.
T18. Use quantization-aware training	Convert high-precision data types to lower precision during training.	LLMs recognize numeric precision reductions during training as quantization, which lowers computational demand.
T19. Use checkpoints during training	Use checkpoints to avoid a knowledge loss in case of a premature termination.	LLMs detect checkpointing mechanisms and relate them to training resumption or fault tolerance.
T20. Design for memory constraints	Consider possible memory constraints during training.	LLMs infer memory constraints from language indicating hardware limits or optimization strategies.

standing of object-oriented principles and semantic relationships between classes and methods. Similarly, the detection of green architectural tactics involves identifying higher-level design decisions that are not explicitly documented and may be scattered across different parts of the codebase. As such, the task of identifying green architectural tactics can be framed as a **complex pattern recognition problem**, for which ML is a natural fit. Among the available ML techniques, I adopted LLMs due to their ability to understand context, infer semantics across non-local code elements, and generalize patterns from natural language definitions to real code structures. LLMs offer a promising basis for detecting design intent at the tactical level that is not explicitly marked but is manifested in the structure and behavior of the code.

More specifically, I mined the latest snapshot of the repositories associated with the selected software projects, extracting all *Python* files—the files that may contain source code related to ML model creation, training, and deployment. To detect architectural green tactics, I applied LLMs directly to the extracted *Python* files. LLMs are highly effective at interpreting source code, enabling them to identify subtle patterns and context-specific practices within these files [226]. In particular, the advanced architecture and extensive training of LLMs allow them to capture complex relationships and subtle patterns within code, making them potentially effective at identifying context-specific instances of green tactics in ML projects. Indeed, by understanding the context of the source code implemented within a file, e.g., recognizing when an algorithm is optimized for energy efficiency or when a model is being pruned to reduce computational load, LLMs may accurately detect the application of green tactics even when these are not explicitly documented. Several factors influenced the decision to use LLMs over other ML models. First, LLMs are effective in natural language processing tasks due to their architecture and training on diverse language datasets, which enhances their ability to identify and interpret code within files [165]. Additionally, LLMs are adept at processing and analyzing large volumes of code, making them well-suited for the scale of the study. Furthermore, LLMs can execute and solve tasks based on

instructions without requiring additional training data [381], which was a critical requirement in this case, given the absence of a labeled dataset specifically tailored for detecting green architectural tactics in software repositories. To this aim, I devised an LLM-based mechanism aiming at detecting the application of green architectural tactics within software code. The approach employs prompt engineering, where carefully crafted input prompts guide the LLM to generate outputs that accurately reflect the presence of tactics [280, 402].

Based on the objectives of the work, I developed a prompt to instruct the LLM in achieving two primary objectives: (1) *'Identify Existing Architectural Green Tactics'*—the LLM was directed to identify the 15 sustainable tactics listed in Table 6.2 and determine their presence in the code; (2) *'Discover Additional Architectural Green Tactics'*—the LLM was tasked with detecting any sustainable tactics not included in Table 6.2. Finally, for each tactic in Table 6.2, I described the rationale of LLMs to detect tactics with a version of the code. To conduct the entire prompt development, I followed the recommendations of the PRIMES framework [79].

To accomplish these objectives, I structured the prompt into the following five key components, as illustrated also in Figure 6.2:

'objective'. The first part of the prompt defines the overarching goal, instructing the LLM to examine the source code to detect both direct and indirect sustainable practices implemented within the code changes analyzed, with a particular emphasis on green architectural tactics for machine learning;

'task description'. I outlined two specific tasks for the LLM. The first task requires the LLM to detect the 15 sustainable tactics from Table 6.2 and verify their presence in the code. The second task is to uncover new sustainable practices not previously documented.

'additional instructions'. This part details information on how the LLM should analyze the code and generate the final output;

'reporting instructions'. The guidelines instruct the LLM to clearly present its findings, indicating the presence of each tactic and including relevant code snippets. If a tactic is not detected, the LLM should explicitly indicate its absence by associating a *"No"* with the tactic in the output;

'examples'. The final part provides examples of how to correctly format and report the results. The examples serve as templates, guiding the LLM in generating output consistent with the desired structure. An example is provided for each possible scenario.

I experimented the prompt with two LLMs, GPT-4o from OpenAI [53] and Claude 3 Haiku from Anthropic [115], to determine the most suitable model for the analysis. These models were chosen for their recognized accuracy in task resolution and their ability to effectively interpret complex, such as source code [52]. The development of the prompt was an iterative process involving multiple refinements to enhance its accuracy and effectiveness. The first two authors of the paper conducted this process through collaborative sessions, where initial versions of the prompt were tested on a subset of projects. Feedback from these tests informed adjustments to the instructions and examples provided to the LLM. The evolution of the prompt, including its various iterations, is documented in the appendix [237].

As a final note, I designed the green architectural tactics mechanism with flexibility in mind, separating the code into two distinct components: one that handles project analysis and another that interfaces with the LLMs. This modular approach allows for easy integration of newer, more accurate, and higher-performing LLMs in the future. As such, the mechanism I developed can be further extended to support future work in the field.

6.3.4 *Model Validation*

To evaluate the suitability of using an LLM to extract architectural green practices, I conducted a pilot test. This process involved testing the prompt

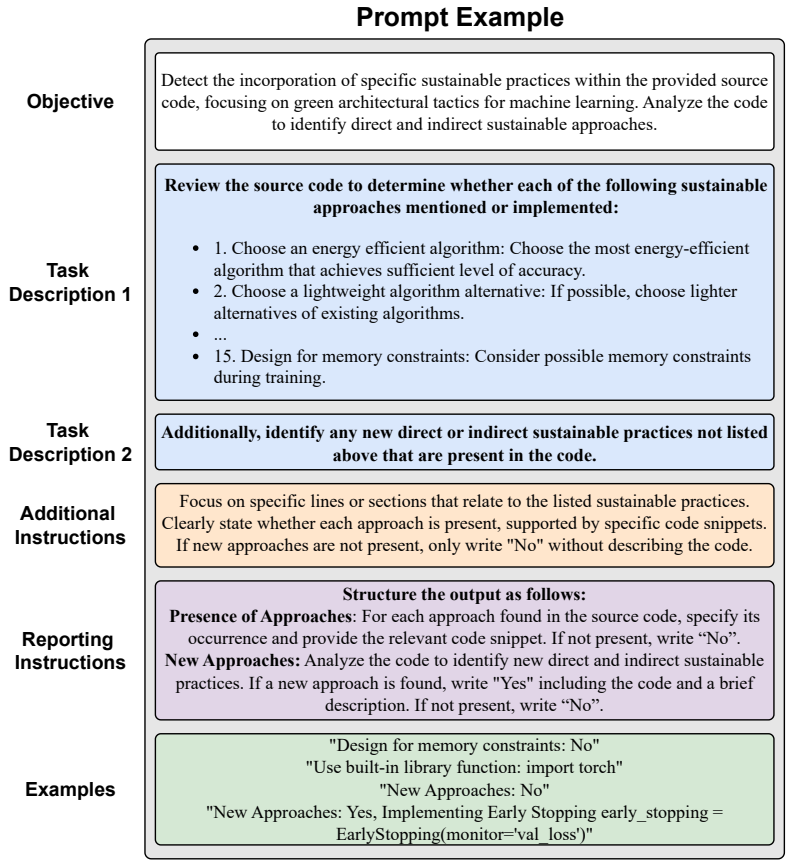


Figure 6.2: The prompt designed to extract architecture green tactics from software repositories.

with both GPT-4o and Claude 3 Haiku across a representative subset of projects. I specifically applied random sampling to ensure unbiased selection [15, 332], identifying 10 projects from the initial dataset, which included 86 Python files.

By analyzing the selected files, I manually constructed an oracle by classifying each file according to the presence or absence of sustainable tactics. This process involved the examination of the code to identify specific implementations of the green tactics considered in the scope of the work. Leveraging three years of research experience in sustainable software

engineering, the author compared each identified practice with those cataloged by Järvenpää et al. [182]. In doing so, the author used the exemplary code snippets provided in the catalog as a reference to determine whether a given piece of code implemented a sustainable tactic. It is important to note that this preliminary evaluation focused exclusively on practices already cataloged in the literature, as these could be reliably identified and manually classified using the well-defined criteria and examples provided by prior research [182]. In contrast, the identification and validation of previously undocumented tactics lack established definitions and examples, making it difficult to manually classify them with the same level of confidence. Consequently, I deferred the classification of new tactics to a subsequent phase of data analysis, which was guided by the insights provided by the LLM (see Section 6.3.5). Indeed, the prompt design not only allowed to identify potential new tactics but also to provide detailed explanations and examples that justified why these practices might qualify as sustainable. This approach allowed for a more informed and rigorous evaluation process in the later stages of the study.

To better illustrate the manual classification process, consider a Python file that implements a neural network model. If the code applies model pruning, where unnecessary neurons are removed to reduce computational load, this would align with the tactic of *'Consider energy-aware pruning'*. The author would then classify the file as containing this sustainable tactic. Conversely, if the code lacks such optimizations and does not exhibit any tactics from the catalog, it would not be classified as sustainable tactics.

Upon construction of the oracle, I executed both GPT-4o and Claude 3 Haiku with the prompt described in Section 6.3.3, collecting their responses without modifying the temperature or any other LLM parameters. Finally, I compared the output of the models with the oracle, computing the overall classification accuracy. Through this procedure, I could iteratively refine the prompt and evaluate its impact on the results. Additionally, using the prompt defined in Section 6.3.3 not only enabled us to assess the classification capabilities of the models from a quantitative perspective

but also offered insights into their performance in identifying additional green architectural tactics. Although this was not the primary focus of the evaluation, observing how each model handled the identification of new tactics provided valuable trends and considerations that informed the selection of the most suitable model for the study. As a result of this preliminary evaluation, both models demonstrated high accuracy, each with its own distinct strengths.

Specifically, GPT-4o achieved an accuracy of 95.58% in classifying green architectural tactics. Although the model was highly accurate, its responses were primarily analytical and lacked detailed explanations. When GPT-4o correctly identified a tactic, it efficiently returned the associated code but offered minimal context or reasoning behind the classification. This means that while GPT-4o was precise in its identifications, it often failed to provide insight into why a particular piece of code was classified as implementing a sustainable tactic. Additionally, GPT-4o did not identify any new sustainable practices beyond those already documented, which limits its effectiveness in discovering novel tactics. In contrast, Claude 3 Haiku exhibited a slightly higher accuracy of 97.91% in classifying tactics. Although it did not always provide the exact code snippets, often opting instead to describe the implemented tactic, it was notably successful in identifying additional sustainable tactics. However, Haiku sometimes repeated tactics that were already known, indicating a need for refinement in distinguishing genuinely novel practices. One of Haiku's key strengths was its ability to provide natural language explanations that clearly articulated the rationale behind its classifications. This feature significantly enhanced the understanding of why certain code segments were associated with specific sustainable practices, making it a valuable tool for users seeking deeper insights into the model's decision-making process. Finally, the two models have different token costs; Claude 3 Haiku has an input/output cost of millions of tokens of \$0.25/1.25 while GPT4o has a cost of \$5/15, making Claude 3 Haiku economical for analyzing a large number of large files.

Based on these findings, I selected Claude 3 Haiku for its higher accuracy,

its ability to identify additional sustainable tactics, and its lower cost. Although Haiku did not always return the specific code related to the tactic, I implemented a mechanism that allowed us to track the repository and file in which the tactic is detected. This ensures that, even if the code snippet is not provided explicitly, the tactic could still be located within the source file. As a final remark, it is worth reporting that, while I have opted to focus on a single LLM for consistency and cost-effectiveness, future research could investigate ensemble methods that exploits the strengths of multiple LLMs to further enhance the detection and explanation of green tactics.

6.3.5 *Data Analysis*

To ease the data analysis process, I developed a Python script [237] that systematically traverses the files in each repository and records key information. For each project, it generates a `csv` file, with each row corresponding to an analyzed file. The `csv` file captures (1) the file name and (2) the output generated by Claude 3 Haiku.

To address $\mathbf{RQ}_1$, the implemented script processed Haiku's responses by segmenting the first part of the output—specifically the classification of existing tactics—into separate columns, with each column representing one of the tactics considered in the study. After collecting all responses, the data was aggregated into a single file that tracked the occurrence of each tactic. The script calculated the total frequency of each tactic across all projects and used this information to address the first **RQ**, providing an overview of how green architectural tactics are adopted in the considered ML-enabled systems.

To address $\mathbf{RQ}_2$, I processed Haiku's responses to isolate the output provided for the second task—specifically, the identification of additional tactics not covered in the original catalog. Given the non-deterministic nature of LLMs, which can give rise to hallucinatory or spurious suggestions, I applied a manual validation process to mitigate this risk. Each candidate tactic was examined to ensure that it represented a consistent and actionable environmental sustainable practice. The manual analysis

removed responses containing hallucinations and optimized the accuracy and reliability of the tactics included in the rest of the paper. I first filtered out any responses where the model did not suggest new tactics, focusing only on instances where the model provided potential new sustainable practices. This could be done automatically, as the prompt required the model to answer *"No"* in cases where no additional tactics were found. From this filtering step, I obtained 1,374 responses, which were analyzed by means of iterative content analysis sessions [202]. More specifically, I manually reviewed each response to determine whether the identified tactics were novel or pre-existing. Each tactic was initially assigned one of three possible grades: *'No'*, *'Maybe'*, or *'Yes'*. Tactics classified as *'No'* were those already included in the existing catalog, unrelated to ML systems, or not considered sustainable. Tactics classified as *'Maybe'* refer to those with ambiguities due to insufficiently detailed descriptions that make it difficult to assess their applicability. These tactics may also address potential environmental sustainability factors, such as energy efficiency, that necessitate further interpretation or contextual understanding and would benefit from a discussion among the authors. Finally, tactics classified as *'Yes'* were clearly described and aligned with sustainable practices for ML-enabled systems.

This first content analysis session was supervised by the second author of the article, who took an active role in guiding the process, providing oversight and ensuring that the classification criteria were consistently applied. Together, the first and second authors also discussed the suitability of the tactics labeled as *'Maybe'*, deliberating on whether these tactics should be reclassified or excluded based on their potential contribution to sustainable practices. Following this initial session, a subsequent content analysis iteration was conducted to further refine the classification. This iteration involved re-evaluating the *'Maybe'* tactics in light of additional insights gained during the discussions. The authors aimed to reach a consensus on all tactics, carefully discussing any disagreements to ensure that only practices clearly aligned with sustainable principles were included in the final set.

From Haiku's responses, I obtained 1,374 files containing new green tactics. However, of these, 964 were classified as *'No'* and were eliminated because they were either not green tactics or were tactics already in the initial list. 259 were classified as *'Yes'*, and 161 were classified as *'Maybe'*. Of the latter, after a discussion, 23 were eliminated, and 387 outputs were analyzed. Once the final set of tactics was established, I thoroughly documented each response. I validated them to understand whether the tactics were truly green and identify any new contributions by taking the following steps: (1) assigning a tactic title, (2) categorizing it into the most appropriate group among those defined by Järvenpää et al. [182], (3) providing a detailed description of the tactic's characteristics, (4) providing a file containing all identified tactics, (5) including relevant code examples illustrating their implementation, (6) performing triangulation looking for the new elicited tactics also in the existing literature and adding references where applicable, and (7) defining the associated quality attributes (QA) using definitions from Järvenpää et al. [182].

It is important to clarify that the references were used to support the validity of the environmental tactics. The tactics included in the new catalog were not present in the previous catalog and are, therefore, identified as new contributions. The presence of a source does not undermine their novelty in this context; rather, it provides supporting evidence from other work that may illustrate or relate to their environmental nature. The process to reproduce the work is in the online appendix [237].

6.4 ANALYSIS OF THE RESULTS

In this section, I report quantitative and qualitative insights from the repository mining study to address the **RQ**s.

6.4.1 RQ_1. *Adoption of Green Tactics in Practice*

Figure 6.3 shows the frequency of adoption of green tactics in the considered 168 ML projects. It shows the number of projects that implemented

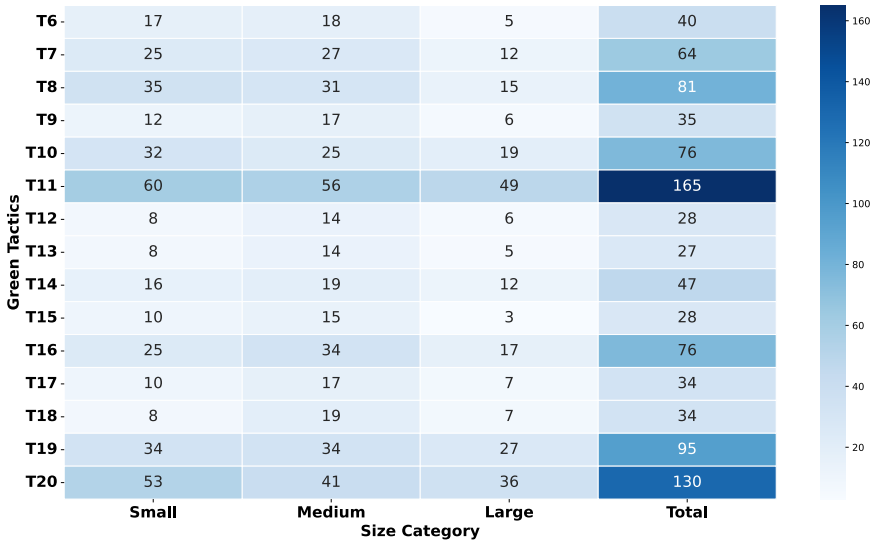


Figure 6.3: Frequency of Green Tactics in ML Projects.

each specific tactic, offering insights into the popularity of these strategies within real software projects across four size categories: *'Small'*, *'Medium'*, *'Large'*, and the full set of projects. The three size categories show distinct adoptions of tactics, with the category *'Medium'* adopting tactics the most, followed by small projects with greater adoption of tactics such as *'Decrease model complexity'*, *'Use dynamic parameter adaptation'*, *'Use built-in library function'*, and *'Design for memory constraints'*. Finally, I found that large projects exhibit lower adoption of all tactics. This may be attributed to several factors, including the higher complexity of larger projects, which can make integrating new tactics or optimizing for environmental sustainability more challenging and resource-intensive [132]. Moreover, large projects often prioritize stability and performance over experimentation, which can slow the adoption of sustainable practices [201].

Considering the totality of all projects, the most widely adopted tactic is the *'Use of built-in library functions'*, with 165 (98.2%) projects adopting it. This high adoption rate underscores the adoption of libraries and frameworks in software projects, which are optimized to improve software

performance compared to custom-made functions. Similarly, the *'Design for memory constraints'* tactic was adopted by 130 (77.4%) projects. This high adoption rate emphasizes the importance of optimizing memory usage for efficiency and preventing memory issues during ML project development.

Other frequently used tactics are the *'Use of checkpoints during training'* by 95 (56.5%) projects, which emphasizes the importance of managing training steps during model training, and *'Reducing model complexity'* by 81 (48.2%) projects, where models are simplified to achieve better efficiency and lower energy consumption. The tactics *'Use dynamic parameter adaptation'* and *'Consider transfer learning'* have been adopted by 76 (45.2%) projects. Finally, 64 (38.1%) projects used the *'Choose a lightweight algorithm alternative'* tactic.

However, certain tactics are less frequently implemented. For instance, *'Enhance model sparsity'* and *'Choose an energy efficient algorithm'* are employed in 47 (28%) and 40 (23.8%) projects, respectively. *'Consider knowledge distillation'* and *'Use quantization-aware training'* are each used in 34 (20.2%) projects, and *'Consider reinforcement learning for energy efficiency'* is adopted by 35 projects.

The least adopted tactics include *'Consider graph substitution'*, *'Set energy consumption as a model constraint'*, and *'Consider energy aware pruning'*, each used by only 27 (16.1%) -28 (16.7%) projects. Lower adoption rates of these tactics could indicate difficulties in their implementation or a lack of knowledge of these tactics.

> **RQ_1 — Adoption of Green Tactics in Practice**
>
> *Green tactics proposed in the literature are widely adopted in ML projects. However, some tactics are less commonly adopted, highlighting the need to raise awareness and provide better tool support to facilitate their broader integration.*

Table 6.3: New Green Tactics for ML-Enabled Systems. (*) Indicate energy efficiency is a secondary quality assurance (QA)

Cluster	Tactic	Target QA	#Projects	Source
Data-Centric	T31: Clean and Standardize Data	Performance*	5 (2.97%)	[380]
	T32: Consider Use Data Augmentation	Accuracy*	21 (12.5%)	[296, 311]
Model Optimization	T33: Ensure Gradient Stability	Performance*	5 (2.97%)	[387]
	T34: Use Adaptive Learning Rate Scheduling	Energy efficiency	12 (7.14%)	[320, 359]
Model Training	T35: Employ Parallel Training	Performance*	24 (14.28%)	[86, 175]
	T36: Use Early Stopping	Energy efficiency	71 (42.26%)	[214, 286]
Deployment	T37: Use Deployment-Specific Model Export Strategies	Energy efficiency	6 (3.57%)	[149]
Management	T38: Consider Model Reproducibility	Recoverability*	6 (3.57%)	[267, 279]
	T39: Dynamic Resource Management	Resource utilization*	6 (3.57%)	[66]

6.4.2 ***RQ$_2$.*** *Elicitation of Additional Green Tactics*

Table 6.3 catalogs the set of new tactics identified in **RQ**$_2$, along with their frequency in the ML projects considered in the study. As shown, the catalog contains nine green architectural tactics for ML-enabled systems, outlined below. Each tactic has been elicited through a semi-automated analysis supported by a LLM (Claude 3 Haiku), followed by a manual validation and refinement process aimed at ensuring precision and relevance. In this context, a tactic is included only if it represents a high-level design decision that contributes to energy or resource efficiency and exhibits architectural scope, consistent with the definition adopted in prior work [182]. To enhance clarity and facilitate traceability, each tactic is described following a common structured template composed of three elements:

- *Definition*: which explains the core idea and scope of the tactic;
- *Rationale and Sustainability Objectives*: which outlines the expected environmental sustainability benefits of the tactic, supported by empirical evidence or relevant literature;
- *Code-example*: which provides an implementation example illustrating the tactics adopted in real-world ML projects.

6.4.2.1 *T31: Clean and Standardize Data*

Definition: this tactic involves transforming raw, unprocessed data into a clean, consistent, and structured format. It includes steps such as noise removal, error correction, handling missing values, resolving inconsistencies, and enforcing standard formats and conventions [380].
Rationale and Sustainability Objectives: as highlighted by Zha et al. [380], clean and standardized data contribute to more efficient training by reducing the number of iterations required for convergence and minimizing the need for extensive hyperparameter tuning. These improvements can lead to lower computational demand and energy consumption, thus supporting environmental sustainability objectives. However, if not carefully applied, it may negatively affect fairness by removing rare or minority instances, which are important for representative learning.
Code-example: example of function to clean and standardize text by handling encoding issues [305], removing line breaks, and unescaping HTML entities:

```
def transform_clean_text(raw):
    decoded = raw.encode("ISO-8859-1").decode("utf-8")
    html_unescaped = Transformer.__html_parser.unescape(decoded)
    html_unescaped = re.sub(r"\r\n", " ",html_unescaped)
    return html_unescaped
```

6.4.2.2 *T32: Consider Use Data Augmentation*

Definition: This tactic consists of synthetically expanding the training dataset by generating new examples from existing data. Common meth-

ods include transformations such as rotation, flipping, noise injection, or more advanced generative models (e.g., GANs or VAEs) [296, 311].
Rationale and Sustainability Objectives: data augmentation improves model generalization by increasing data diversity without requiring new data collection or additional model complexity. According to Salehi and Schmeink [296], it enhances data efficiency and reduces energy consumption by lowering the number of training epochs needed to reach acceptable performance. This results in reduced compute time and fewer carbon emissions, making it an effective green strategy in data-constrained scenarios. However, it may introduce artifacts or noisy instances that negatively affect explainability or lead to unpredictable model behaviors if not well controlled. Careful calibration of augmentation techniques is therefore required.
Code-example: a basic augmentation [415] for audio data by randomly adjusting gain or swapping audio channels:

```
def _augment_gain(audio, low=0.25, high=1.25):
    gain = low + torch.rand(1) * (high - low)
    return audio * gain
def _augment_channelswap(audio):
    if audio.shape[0] == 2 and torch.FloatTensor(1).uniform_() <
     0.5:
        return torch.flip(audio, [0])
    return audio
```

6.4.2.3 *T33: Ensure Gradient Stability*

Definition: this tactic involves applying techniques such as gradient clipping or normalization to maintain stable gradient values during model training. It aims to prevent vanishing or exploding gradients, which are particularly problematic in deep or recurrent neural networks.
Rationale and Sustainability Objectives: unstable gradients can significantly slow convergence or cause divergence in deep learning models. Zhang et al. [387] provide theoretical and empirical evidence that gradient clipping improves training stability and allows faster convergence by enabling the optimizer to traverse non-smooth regions more effectively. As faster convergence reduces the number of required training epochs, this

tactic contributes to lowering energy consumption and computational cost. However, excessive clipping may hinder learning capacity or affect convergence to optimal minima, potentially impacting final model performance or explainability. Careful tuning is required to balance efficiency with learning quality.

Code-example: gradient clipping [416], which helps to stabilize the training process and prevent numerical issues:

```
trainer = pl.Trainer(..., gradient_clip_val = conf["training"]["
    gradient_clipping"])
```

6.4.2.4 *T34: Use Adaptive Learning Rate Scheduling*

Definition: this tactic involves dynamically adjusting the learning rate during training, often in response to validation performance or training epochs. Common strategies include decay schedules or cyclical learning rate (CLR) policies, where the learning rate oscillates between bounds to improve convergence and stability [320, 359].

Rationale and Sustainability Objectives: adaptive learning rate schedules accelerate convergence and improve model quality without extensive manual tuning. Smith [320] showed that cyclical schedules can achieve the same accuracy as fixed schedules in fewer iterations, reducing training cost. Wang et al. [359] found that CLR enables smaller batch sizes and faster convergence in neural machine translation, making it a practical choice in resource-constrained settings. These benefits contribute to reducing energy consumption and computational waste. However, poorly configured schedules might lead to instability or suboptimal convergence, affecting reproducibility and accuracy. Careful boundary selection is required.

Code-example: example of learning rate scheduling configuration for training [408]:

```
lr_config = dict(policy = 'step', warmup='linear', warmup_iters
    = 500, warmup_ratio = 1.0 / 3, ...)
```

6.4.2.5 *T35: Employ Parallel Training*

Definition: this tactic distributes the training of a machine learning model across multiple processing units (e.g., GPUs, TPUs, CPUs, or nodes in a cluster) to exploit hardware parallelism. Strategies include data parallelism, model parallelism, and pipeline parallelism [86, 175].
Rationale and Sustainability Objectives: parallel training accelerates convergence by processing larger datasets or deeper models faster. Dean et al.[86] showed that their DistBelief framework achieved up to 12× speedup using 81 machines, drastically reducing training time and enabling the use of larger models within a fixed budget. Huang et al.[175] introduced GPipe, demonstrating almost linear throughput scaling via pipeline parallelism, allowing models up to 83.9B parameters to be trained efficiently. Faster convergence and higher hardware utilization reduce wall-clock training time and energy consumption. However, it may increase implementation complexity and resource cost, and introduce reproducibility challenges due to hardware-specific behavior and synchronization differences. Robust orchestration is required to preserve training correctness.
Code-example: example of distributed training [407]:

```
from tensorflow.python.distribute.cross_device_ops import
    AllReduceCrossDeviceOps

cross_tower_ops = AllReduceCrossDeviceOps('nccl', num_packs=
    n_gpus)
strategy = tf.distribute.MirroredStrategy(
    devices=[f'/gpu:{i}' for i in range(n_gpus)],
    cross_device_ops=cross_tower_ops)
```

6.4.2.6 *T36: Use Early Stopping*

Definition: this tactic involves monitoring the model's performance on a validation set and halting training once the performance ceases to improve. The goal is to prevent overfitting and ensure the model is trained only for the necessary number of epochs.
Rationale and Sustainability Objectives: early stopping reduces unnecessary training by avoiding overfitting phases, directly lowering computa-

tional cost and energy use. This tactic minimizes redundant computation and offers one of the clearest energy-performance trade-offs in green AI[214, 286]. However, it may increase implementation complexity and resource cost, and introduce reproducibility challenges due to hardware-specific behavior and synchronization differences. Robust orchestration is required to preserve training correctness.
Code-example: example of early stopping [413]:

```
from pytorch_lightning.callbacks import EarlyStopping
early_stopping = EarlyStopping(monitor=monitor, mode=direction,
    min_delta=0.0, patience=100, strict=True, verbose=False,
    check_finite=True)
```

6.4.2.7 *T37: Use Deployment-Specific Model Export Strategies*

Definition: this tactic involves exporting trained models into optimized formats (e.g., ONNX, WebAssembly) that are tailored to the characteristics of the deployment environment, such as edge devices or embedded systems with constrained memory or processing power. The export format is chosen to minimize runtime overhead while maximizing compatibility and performance.
Rationale and Sustainability Objectives: deployment-specific export strategies optimize the inference phase by reducing overhead and improving runtime efficiency. Hampau et al. [149] conducted an empirical comparison of ONNX, Docker, and WebAssembly (WASM) deployments across multiple edge devices. Their results show that ONNX runtime offers the best trade-off between energy consumption and execution time, while WASM achieves the lowest memory footprint. These findings confirm that selecting the appropriate export format based on target hardware can significantly reduce energy usage and resource consumption in real-world deployments. However, the conversion process may introduce compatibility issues or require additional validation steps to ensure correctness, potentially affecting reliability or reproducibility if not well-managed.
Code-example: converting a model in ONNX [412]:

```
torch_onnx_export(decoder, args = tuple(input_list),
    export_params = True, input_names = input_names,
```

```
    output_names = output_names, dynamic_axes = dynamic_axes,
    opset_version = 12, do_constant_folding = True, verbose =
    verbose, ...)
```

6.4.2.8 *T38: Consider Model Reproducibility*

Definition: this tactic focuses on ensuring that ML systems can be reproduced under consistent conditions by explicitly controlling randomness, logging configurations, and structuring code for repeatability. Common practices include fixing random seeds, recording hyperparameters, tracking datasets and code versions, and modularizing model components [267, 279].

Rationale and Sustainability Objectives: improving reproducibility reduces the need for redundant experiments and trial-and-error iterations, thus saving time and computational resources. Qi et al.[279] show that re-engineering models into modular, reusable components can reduce inference time by 42.41% and weight usage by 89.89%, decreasing overhead in repeated experiments. Similarly, Pan and Rajan[267] report that decomposing CNNs into reusable modules lowers CO_2 emissions by up to 37× compared to training from scratch.

Code-example: example for track training progress [409]:

```
wandb = dict(init_kwargs = dict(project = "caltech", name = "
    convmlp_2x16", entity = "mlp", config = dict(work_dirs = "${
    work_dir}", total_step = "${runner.max_epochs}")), interval
    = 50)
```

6.4.2.9 *T39: Dynamic Resource Management*

Definition: this tactic refers to the dynamic allocation and adjustment of computational resources (e.g., CPU, GPU, memory) during model training and inference, based on real-time workload demands. It includes techniques such as auto-scaling, job preemption, elastic allocation, and topology-aware scheduling to optimize resource use throughout the ML pipeline [66].

Rationale and Sustainability Objectives: Chiang et al. [66] demonstrate that

dynamic resource management in ML pipelines significantly improves efficiency: their system, DynamoML, achieved a 60% increase in system throughput and a 70% reduction in training time compared to standard Kubernetes. These improvements reduce energy consumption by avoiding idle resource use and minimizing job execution time. Techniques like elastic GPU sharing and workload-aware scheduling contribute directly to energy savings and more sustainable ML operations. However, it may introduce complexity in orchestration, impact reproducibility due to resource variability, and require additional infrastructure (e.g., schedulers, controllers) to ensure consistent behavior across dynamic conditions.
Code-example: a shared memory management or efficient data management between processes or threads [414]:

```
sa_array = sa.create(name, data.shape, data.dtype) np.copyto(
    sa_array, data)
```

> **RQ$_2$ — Elicitaion of Additional Green Tactics**
>
> *I extracted nine new green tactics adopted by developers. Among them, 'Use Early Stopping' and 'Employ Parallel Training' are the most adopted.*

6.5 DISCUSSION AND IMPLICATIONS

Our study highlights several implications for the SE community and for society by minimizing the energy footprint.

On the Adoption of Green Tactics. When comparing the adoption of green tactics identified in previous studies, particularly the catalog of 30 tactics by Järvenpää et al. [182], I analyzed 168 open-source ML projects on GitHub using LLMs. Our analysis confirms that practitioners adopt many of these tactics. Specifically, tactics like *'Use of built-in library functions'* and *'Design for Memory Constraints'* have high adoption rates, suggesting that these tactics are well-supported and effectively integrated into current development processes. Conversely, other tactics such as *'Set Energy*

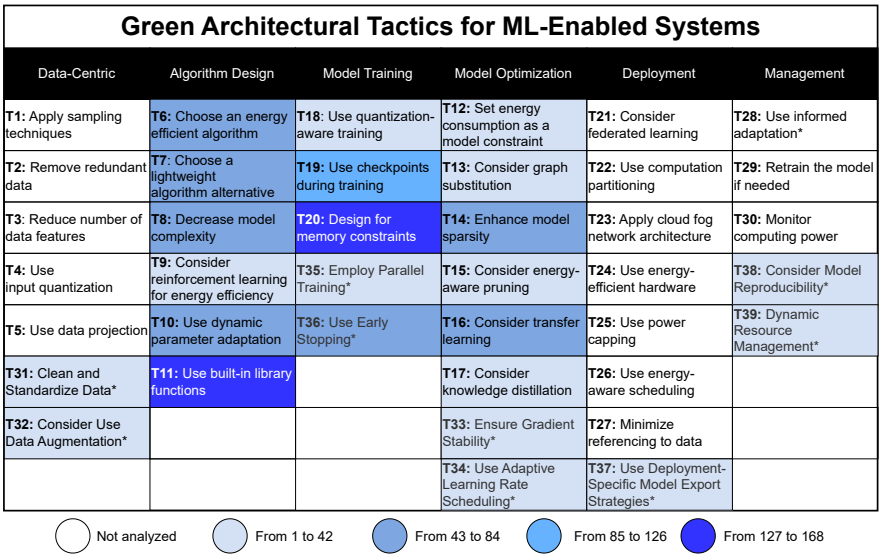


Figure 6.4: Catalog of 39 Green Architectural Tactics for ML-Enabled Systems, adopted from [182] and enhanced with use frequency of all tactics and the elicitation of nine new tactics. The symbol '*' indicates a new tactic found by our LLM-based mechanism. Colors indicate frequency using quartiles on 168 ML projects.

Consumption as a Model Constraint' and *'Consider Graph Substitution'* exhibit lower adoption rates in our study. Although our analysis is based on open-source projects, the results have broader implications for different ML contexts.

Many tactics are applicable to proprietary and enterprise-level systems, but some, such as energy-aware optimizations, may require adaptations for specific domains such as resource-constrained environments. Green tactics also hold potential for improving accessibility and affordability by reducing hardware and energy costs, but their adoption must be balanced against potential trade-offs in performance and usability, particularly in critical applications such as healthcare and autonomous systems. Adoption barriers, including organizational resistance, lack of awareness, and perceived costs, are particularly evident in larger projects where coordination and competing priorities complicate sustainability efforts. Addressing

these barriers will require better education, improved tooling, and incentives for adopting green tactics. Engaging directly with developers through interviews or analyzing project repositories could uncover additional insights into these challenges and inform strategies for overcoming them.

> **Implication #1.** Our results show that the adoption of green architectural tactics varies across different projects. This suggests a need for targeted strategies and tools tailored to address specific barriers. Future research should aim to explore the socio-technical reasons behind current adoption limitations.

On the Elicitation of Additional Green Tactics. When comparing the newly identified green tactics with those cataloged in existing literature, I find that developers are employing strategies not yet formally cataloged. This discovery has significant implications for both researchers and practitioners. The addition of these tactics expands the existing body of knowledge on green tactics [182, 350], increasing the total to 39 tactics. The new catalog is shown in Figure 6.4. The newly identified green tactics are marked with a '*' symbol. Furthermore, the figure illustrates the extent to which all the tactics analyzed in this study—both those previously elicited and the new ones—are employed within the systems considered. This updated catalog represents the main contribution of the work, (1) expanding the body of knowledge on green tactics by identifying previously unrecognized strategies and (2) providing insights into their practical adoption and prevalence within the systems analyzed.

Notably, some of these new tactics, such as *'Use Early Stopping'*, exhibit high adoption rates (found in 71 projects), suggesting their practical relevance and effectiveness. This opens new avenues for further research into the impact and efficiency of these tactics. Empirical studies could focus on quantifying the environmental benefits and energy savings of these tactics in different ML systems. There is also a need to explore how these tactics can be adapted to proprietary or large-scale industrial systems, which may face unique challenges. Examining the interaction between green tactics and other software QA, such as scalability, reliability, and maintainability, could provide insights into their broader applicability and impact in ML-

enabled systems. While our study emphasizes energy efficiency, other QAs also affect accuracy. Typically, achieving higher accuracy is viewed as a trade-off with energy consumption, where more training leads to increased energy use. A significant drop in accuracy to meet energy goals can make ML models unreliable for practical applications, underscoring the need to prioritize accuracy. However, recent research suggests that it is possible to reduce energy consumption without compromising accuracy [182, 350]. By understanding how different QAs interact, I can design systems that effectively balance energy efficiency, accuracy, and other requirements. By providing practical code examples, I aim to facilitate the integration of these tactics into more projects, potentially fostering wider adoption of green architectural tactics. This effort helps developers create more efficient systems and motivates the software engineering community to further embrace and enhance the use of emerging techniques. Finally, future research could focus on tactics not considered in our study to identify their adoption in real projects.

Implication #2. The discovery of nine new green architectural tactics highlights ongoing opportunities in sustainable software engineering. These findings encourage further exploration of their energy savings and integration into software quality frameworks, paving the way for studies and practical guidelines for developers. Importantly, offering practical code snippets for these tactics significantly lowers the barriers to adoption, making it easier for developers to integrate them into real projects. By demonstrating how to implement these tactics, I aim to enhance their accessibility and encourage wider adoption, ultimately contributing to the development of more sustainable ML-enabled systems.

Broader Impact. The results of our work have a broader impact, aligning closely with the principles of Green AI [304] and SDG Goals[5]. By emphasizing strategies to reduce resource consumption and CO_2 emissions, green architectural tactics contribute significantly to addressing the environmental impact of AI technologies while promoting efficient use of resources.

[5] https://www.globalgoals.org/goals/.

These strategies are particularly relevant in sectors that prioritize energy efficiency and environmental responsibility. Balancing energy efficiency with performance and usability remains a major challenge. Sectors such as healthcare and autonomous systems often demand uncompromised performance and reliability, creating potential trade-offs in implementing green tactics. Furthermore, while green architectural tactics promise to reduce hardware costs and resource requirements, achieving this balance requires careful consideration to ensure accessibility and affordability. These societal benefits demonstrate the importance of adopting sustainable tactics in ML development and underscore the need to explore these trade-offs further to guide informed decision-making.

To fully realize these benefits, collaboration between researchers and practitioners is essential. Researchers can focus on developing robust benchmarks and frameworks to evaluate the effectiveness of green tactics, while practitioners can integrate these strategies into different application areas. Working together, these stakeholders can ensure that green tactics mitigate environmental impacts and drive broader social progress, promoting a more sustainable and responsible future for AI technologies.

Implication #3. Our study highlights the need for interdisciplinary collaboration among researchers and practitioners to adopt the benefits of green tactics. This requires initiatives that combine technological advances with regulatory support, ensuring that green tactics lead to both environmental sustainability and societal benefits.

7

A METHODOLOGICAL FRAMEWORK FOR LLM-BASED MINING OF SOFTWARE REPOSITORIES

7.1 INTRODUCTION

Large Language Models (LLMs) have taken on a central role in Software Engineering (SE), reshaping established practices in software development, maintenance, and comprehension [103, 135, 166]. Recent work by Hou et al. [166] has systematically mapped how LLMs are used in SE, providing a comprehensive taxonomy of tasks supported by LLMs across the development lifecycle, from code generation [93, 374] and comprehension [252] to testing [275], and maintenance [274]. Their findings highlight the breadth of LLM applicability and their disruptive potential in automating and augmenting main SE practices.

While recent work has mapped where LLMs can be applied across Software Engineering tasks, less attention has been given to how LLMs are used as part of the software engineering research process itself. **Rather than dealing with the use of LLMs to assist software engineers, this article focuses on understanding and improving the methodological use of LLMs by SE researchers conducting empirical studies.** In particular, I focus on Mining Software Repositories (MSR) as a domain where LLMs are increasingly employed [166]. Recent contributions have begun to explore this methodological shift, examining how LLMs are used to support systematic reviews [73, 105] and qualitative analysis [19, 208]. Building on these efforts, the work provides an overview of how researchers operationalize LLMs in MSR and identifies approaches and threats in this new emerging research area.

Traditionally, MSR has relied on hand-crafted rules [84, 341], manual analysis [47, 62, 155], and classical machine learning methods [18, 244,

346] to extract insights from development artifacts such as commits, issues, and pull requests. These approaches typically focused on structured data and required extensive manual annotation and data preparation [24, 40]. Today, the advent of LLMs is reshaping how repository mining is approached. Thanks to their ability to process unstructured and heterogeneous sources of information, LLMs can synthesize knowledge across multiple artifacts. Unlike traditional MSR pipelines, which relied on task-specific models and explicit feature engineering, LLM-based workflows are increasingly driven by prompt design, model configuration, and iterative validation without the need to conduct extensive pre-processing steps [82, 135, 309]. In the survey of Hou et al. [166], the MSR topic is briefly acknowledged as a use case for LLMs; however, the study, like much of the existing literature, remains task-centric and overlooks how LLMs are operationalized in practice. Remarkably, using LLMs for MSR introduces a new set of threats. Researchers must design prompts tailored to heterogeneous artifacts, manage non-deterministic and context-sensitive model outputs, and validate results across large datasets without standardized protocols. Yet, these workflows are rarely described in sufficient detail. Decisions regarding data sampling, model selection, inference configuration, and evaluation strategies are often implicit, making replication difficult and methodological rigor inconsistent.

Recent studies have begun to build a methodological foundation for using LLMs in research. These efforts range from proposing high-level guidelines for evaluation and responsible use [297, 355], to empirically testing LLM capabilities for specific research activities like study replication and results validation [219, 358], and introducing preliminary frameworks for MSR tasks [80].

> ⚠ *While recent studies have introduced high-level guidelines and preliminary frameworks for using LLMs in empirical software engineering, they often remain abstract, focusing on evaluation principles or individual tasks. What is still missing is a systematic, bottom-up understanding of how researchers make concrete design decisions*

> *throughout the entire LLM-based MSR pipeline. This lack of grounded operational insight makes it difficult to assess whether current approaches effectively minimize threats to validity, and limits the ability to build reproducible and methodologically sound studies.*

As highlighted in **Gap #5**, there is still a lack of support for empirical methodological frameworks for LLM-based mining repository. This chapter aims to fill a gap by developing a methodological framework that captures how LLMs are used in MSR workflows. The framework is grounded in an empirical investigation that identifies methodological approaches, threats to validity, and the design choices employed to ensure rigor and reproducibility. Building on our previous work [80], which introduced a preliminary version of the PRIMES framework based on our experience, this study extends and generalizes those insights through a broader empirical investigation. Complementing prior efforts to propose LLM-oriented guidelines [297, 355] and reflect on the methodological implications of LLM usage in SE research [19, 105, 339], the study offers a domain-specific contribution focused on LLM-based repository mining.

To better understand how researchers employ LLM in MSR, I conducted a mixed-method exploratory study that combines a rapid review of 15 (out of initially considered 7,973) peer-reviewed articles [45] with a survey with 22 (initially targeted over 143) researchers experienced in LLM-based mining studies. This investigation revealed that the integration of LLMs into MSR is reshaping the set of activities that researchers conduct, changing methodological approaches and introducing new threats to validity that must be carefully addressed throughout the research process. Yet, in the absence of a structured and empirical framework, many studies risk falling short of the rigor required for reproducibility and validity. When researchers apply LLMs without clear objectives, well-designed prompts, or systematic validation, results may appear plausible but reflect shallow or inconsistent reasoning. Without methodological rigor, studies can unintentionally produce misleading findings, draw unsupported conclusions, or generate outputs that cannot be replicated. Until such frameworks become standard, there remains a high risk that LLM-based MSR research

continues to grow in volume but not in reliability. To address this gap, I refine and extend our initial framework [80] by introducing PRIMES 2.0, a structured methodological guide designed to support the responsible and rigorous use of LLMs in MSR studies. PRIMES 2.0 defines a sequence of six actionable stages across the research workflow, while explicitly highlighting the potential threats that may arise if these substages are neglected or poorly implemented. For each threat, the framework proposes concrete mitigation strategies, offering researchers practical guidance to ensure reproducibility, interpretability, and methodological soundness. Beyond a static checklist, I envision PRIMES 2.0 as an evolving resource that can guide current researchers and be iteratively improved as tools, models, and empirical needs evolve over time.

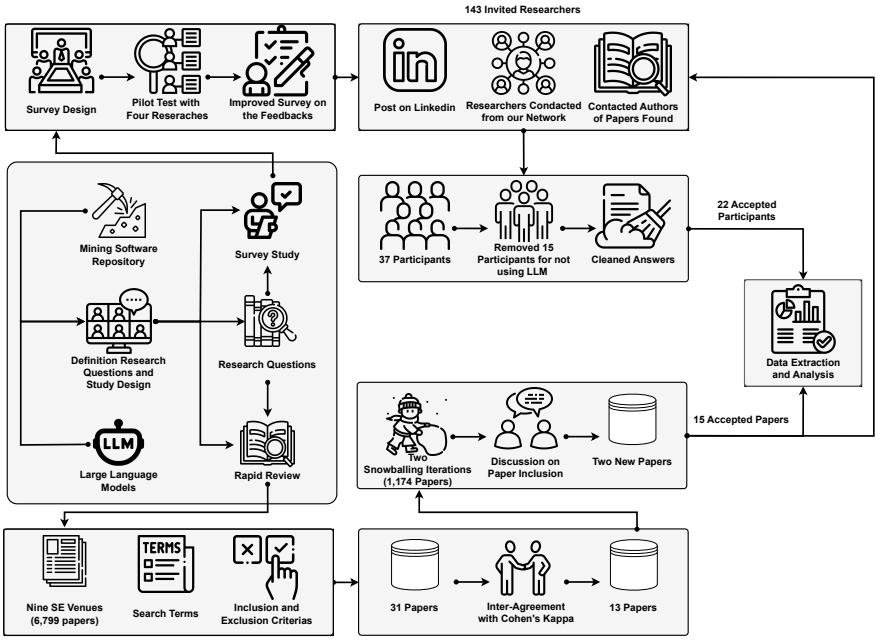


Figure 7.1: Overview of the Research Process.

7.2 RESEARCH PROCESS OVERVIEW

To define the research goal, I follow the Goal-Question-Metric (GQM) approach [39]. The *goal* of this study is to *analyze* the use of large language models (LLMs) *for the purpose of* data collection and analysis *with respect to* empirical rigor and threats to validity mitigation *from the perspective of* of SE researchers *in the context of* mining software repository (MSR). Researchers aim to understand recurring obstacles and usage patterns to inform future tools and research agendas and seek actionable insights to integrate LLMs into MSR workflows effectively.

To address this objective, I focus on two main research questions (**RQ**s). First, $\mathbf{RQ}_1$ aims to understand how LLMs are used in practice for MSR. This includes the specific patterns employed, prompting techniques, validation strategies, and configuration choices that determine how LLMs function in research scenarios [80, 166]. Understanding these usage patterns enables the identification of common practices, tool configurations, and validation strategies, which can support the development of practical guidelines, improve reproducibility, and foster more effective integration of LLMs into MSR workflows. Therefore, I ask:

> **$\mathbf{RQ}_1$—Large Language Model Approaches for Empirical Rigor**
>
> *What approaches are employed when using large language models for repository mining?*

Subsequently, $\mathbf{RQ}_2$ investigates the specific threats to validity encountered when applying LLMs to MSR. While LLMs are increasingly employed in various tasks, they also introduce new technical, methodological, and cost-related difficulties [166, 297]. By identifying these threats, I aim to highlight the issues that can inform future mitigation strategies and empirical validation efforts. Therefore, I ask:

> **$\mathbf{RQ}_2$—Large Language Model Threats**

> *What are the current threats to validity faced by researchers when they use large language models for repository mining?*

To answer these research questions, I adopted a *mix-method* exploratory study [186] consisting of two complementary empirical studies. Specifically, I conducted a rapid review to analyze existing published papers and a survey study to collect insights from researchers. Although the two studies differ in scope and methodology, they both contribute to answering the same research questions by capturing evidence from both the published literature and contemporary practitioner experience.

The first empirical study consisted of a *rapid review* of peer-reviewed literature [45], designed to systematically collect and synthesize existing studies describing how LLMs are applied in repository mining studies. I followed the guidelines of Kitchenham et al. [199] and scoped the search to the period between 2020 and 2025, as informed by Hou et al. [166]. The search targeted high-impact software engineering venues (e.g., ICSE, MSR, TOSEM), and applied inclusion and exclusion criteria focused on empirical applications of LLMs to repository mining studies. To increase coverage and identify additional relevant studies potentially missed during the initial query, I also conducted backward and forward snowballing on the references and citations of the selected papers [365]. This process led to the identification of 15 relevant studies. While informative, the limited number of retrieved articles, despite the growing relevance of LLMs in SE, indicated that the academic literature alone might not offer a sufficiently broad or up-to-date picture of current practices in the field.

To address this limitation, I complemented the review with a second empirical study: a structured *questionnaire survey study* aimed at collecting insights from researchers experienced in LLM-based mining of software repositories. The survey followed best practices for empirical research, as recommended by Kitchenham et al. [198] and Dillman et al. [90]. The survey included both closed and open-ended questions to capture respondents' prompting strategies, validation practices, tool usage, configuration choices, and perceived threats. I conducted a pilot test with four researchers to refine clarity and validity, then distributed the final

survey among academic and industry researchers working on MSR with LLMs. This allowed us to gather a more diverse set of quantitative and qualitative responses that extend the findings of the rapid review with the practitioner perspectives. The integration of these two methods improves the ability to triangulate evidence effectively. The review provides a comprehensive overview of established practices, drawing on an extensive published literature for a solid foundation. The survey investigates the current landscape, capturing current and novel insights from practitioners. This dual approach enriches current understanding and provides a more nuanced perspective on the use of LLMs in repository mining studies.

To ensure methodological rigor and reporting transparency, I followed the ACM/SIGSOFT Empirical Standards[1], particularly the guidelines under the *"Review Articles"*, *"Questionnaire Surveys"*, and *"Mixed Methods"* categories. In the remainder of this section, I describe each empirical study in detail, as illustrated in Figure 7.1.

7.2.1 *Rapid Review*

I conducted a *rapid review*, a methodology that streamlines or omits certain steps of a traditional SLR to synthesize evidence in a shortened timeframe [45]. The rationale for choosing this approach over a standard SLR is twofold. First, the application of LLMs in software engineering is a nascent and rapidly evolving field; a systematic literature review (SLR) would be time-consuming and risk becoming obsolete before publication, failing to capture the most current practices. Second, the goal is to provide a timely summary of this emerging landscape to identify current threats and inform the subsequent investigation. As a result, I focused on SE venues that publish articles on repository mining studies and articles that have been published up to 2025. A rapid review is expressly designed for such contexts. While reducing the number of possible articles that I may have omitted, they offer a compromise that prioritizes timeliness and relevance to a contemporary topic rather than exhaustiveness. Throughout the pro-

[1] https://github.com/acmsigsoft/EmpiricalStandards

cess, I still followed established guidelines by Kitchenham et al. [199] where applicable to maintain methodological rigor.

7.2.1.1 *Seed Set Search*

I constructed the search strategy following the established guidelines by Kitchenham et al. [199] for identifying primary studies in empirical SE. To define the search space, I adopted a *seed set approach*, a commonly used strategy in studies targeting emerging technologies (e.g., [83, 166]). The goal was to identify peer-reviewed research articles that (i) explicitly utilize LLMs in the SE context, and (ii) apply these models specifically to tasks related to MSR.

The search process was carried out in three structured steps:

Publication Period. In line with prior literature on LLM usage in SE [166], I scoped the review to the period between 2020 and 2025. This interval reflects the emergence and maturation of generative models (e.g., GPT-3, Claude 3 Haiku), and aligns with the recent growth of empirical studies leveraging LLMs in practice.

Target Venues. I selected nine high-quality SE venues known for publishing empirical research, as shown in Table 7.1, analyzing a total of 6,799 papers in the selected venues.

Table 7.1: Publication Venues.

Venue	Name
Journal	ACM Transactions on Software Engineering and Methodology (TOSEM)
Journal	IEEE Transactions on Software Engineering (TSE)
Journal	Empirical Software Engineering (EMSE)
Conference	International Conference on Software Engineering (ICSE)
Conference	Mining Software Repositories (MSR)
Conference	International Conference on Automated Software Engineering (ASE)
Conference	Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)
Conference	ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA)
Conference	Empirical Software Engineering and Measurement (ESEM)

These venues were chosen based on their relevance to empirical studies and their increasing inclusion of LLM-related research in recent years.

By concentrating on these sources, I ensured coverage of high-quality, peer-reviewed contributions relevant to the research questions.

Search Terms and Fields. I applied the search terms to the *title*, *abstract*, *conclusion* and *keywords* fields of each article [199]. The query was constructed using Boolean logic: I used the `OR` operator to include alternative expressions and synonymous terms within each thematic group, and the `AND` operator to connect conceptually distinct groups—ensuring that retrieved studies addressed both the use of LLMs and their application to software repository mining.

The terms were derived from the research questions and informed by recent terminology used in the literature by Hou et al. [166], who identified common descriptors in LLM-related software engineering studies. To address variations in terminology to address the goal on the repository mining studies, I narrowed the search to include acronyms, synonyms, and both general and specific terms. I grouped the terms into two main categories:

- **LLM-related keywords:** "Large Language Model", "LLM", "Generative AI", and names of specific LLMs such as "GPT", "Claude", "Gemini", "LLaMA".
- **Software repository mining keywords:** "Empirical Software Engineering", "Automated Data Collection", "Mining Software Repository", "Repository mining", "Open-Source Project", as well as names of specific open-source repositories such as "GitHub", "Hugging Face", and "StackOverflow".

This query structure was designed to maximize both *recall*, by covering lexical variation, and *precision*, by filtering for conceptual relevance. The resulting set of papers was subsequently screened using the predefined inclusion and exclusion criteria to identify the final set of studies analyzed in this review.

Table 7.2: Inclusion and exclusion criteria used in the selection process.

Inclusion Criteria (IC)
(IC1) The article explicitly states that at least one Large Language Model (LLM) is used.
(IC2) The study applies LLMs to software repository-related tasks (e.g., commit classification, data extraction).
(IC3) The study presents empirical results, practical experience, or methodological contributions.
Exclusion Criteria (EC)
(EC1) Duplicate articles or multiple versions of similar studies authored by the same group.
(EC2) Studies published in books, theses, monographs, keynotes, panels, or venues without a peer-review process.
(EC3) Tool demonstrations, editorials, or grey literature (e.g., technical reports, preprints).
(EC4) Articles not written in English.
(EC5) Articles that mention LLMs but do not describe their application to software repository analysis.
(EC6) Studies focused on LLM applications unrelated to repository mining (e.g., code generation, summarization, literature review).

7.2.1.2 *Exclusion and Inclusion Criteria*

To ensure the methodological rigor and relevance of the review, I defined a set of inclusion and exclusion criteria prior to the screening process. These criteria were informed by established practices in empirical software engineering [199] and aligned with the approach adopted by recent studies on LLMs in SE [166, 399]. Given the objective to analyze how LLMs are used for MSR, the criteria were designed to filter in only those studies that offer direct, documented evidence of such applications, while systematically excluding irrelevant or low-quality sources.

I included studies that (i) explicitly employ at least one Large Language Model (LLM), and (ii) apply that model to tasks directly related to software repositories, including mining activities, commit classification, quality analysis, or automated data extraction. I also required that selected studies provide empirical results or methodological guidance within the scope of LLM-based repository mining. In contrast, I excluded studies that lacked technical depth, methodological clarity, or contextual relevance. In particular, I excluded duplicate or overlapping publications from the same research, studies published in non-peer-reviewed articles (e.g., theses,

preprints, book chapters, editorials), and papers not written in English[2]. Furthermore, I excluded studies that merely referenced LLMs without providing a description of their application to repository mining, as well as studies focusing on software engineering techniques to improve LLMs, rather than on the application of LLMs to software engineering problems.

Unlike Hou et al. [166], I limited the scope to peer-reviewed articles to ensure consistency and scientific validity of the selected literature. The full list of inclusion and exclusion criteria applied during the review is summarized in Table 7.2.

The screening process was conducted by I, who applied the defined inclusion and exclusion criteria to the initial set of 6,799 papers. As a result, 31 primary studies were selected.

7.2.1.3 *Study Selection Validation.*

Following the initial screening process, the first and second authors independently reviewed the 31 selected papers to assess their eligibility for final inclusion. To quantify their agreement, I employed Cohen's Kappa coefficient [70], using a three-point agreement scale: 0 indicating disagreement, 0.5 indicating the need for further discussion, and 1 indicating full agreement. A Cohen's Kappa value of 0.75 or higher was considered acceptable, indicating substantial agreement. In the first round, the resulting Kappa value was 0.57, indicating moderate agreement. To resolve discrepancies and ensure consistent application of the inclusion criteria, the authors conducted a consensus meeting to discuss all cases of disagreement and clarify ambiguous interpretations. A second round of evaluation was then performed, which resulted in a perfect agreement score (Cohen's Kappa = 1.00).

As a result of this process, 18 papers were excluded, leaving a final set of 13 primary studies. To enhance the coverage of relevant literature, I subsequently conducted a backward and forward snowballing phase on the reference lists and citations of the included papers.

[2]These two last ECs apply when considering snowballing, see next subsection

7.2.1.4 *Snowballing.*

To complement the seed set and ensure broader coverage of relevant literature, I conducted a backward and forward snowballing process, following the guidelines of Wohlin et al. [365]. The process was conducted using *Google Scholar*,[3] an academic search engine that facilitates the analysis by allowing rapid access to both referenced and citing works.

The snowballing was executed in two levels. In the first level, the references and citations of the 13 included studies were reviewed, resulting in a total of 991 papers screened (876 Backward and 115 Forward). Two papers were identified as potentially relevant in this phase. The second level involved applying the same process to the outputs of the first-level snowballing, during which an additional 183 papers were reviewed (174 Backward and 9 Forward); however, no further papers met the inclusion criteria. The two papers identified were reviewed in a joint meeting by the first and second authors. After discussion, both papers were judged to meet the inclusion criteria and were approved for inclusion in the final dataset. These additions brought the total number of included primary studies to 15.

7.2.1.5 *Rapid Review Data Extraction and Analysis.*

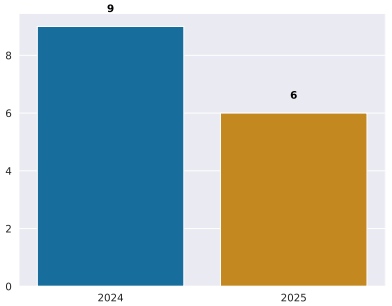


Figure 7.2: Papers published over time.

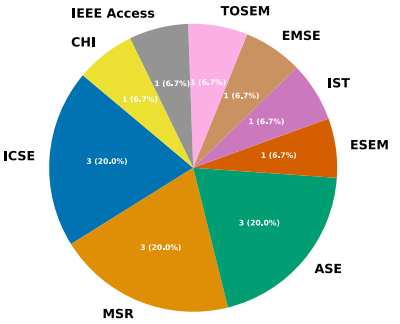


Figure 7.3: Papers venues.

[3] Link to *Google Scholar*: https://scholar.google.com/

To support the synthesis of results and ensure consistent evidence collection across studies, I designed a structured data extraction protocol. I performed the data extraction using a predefined spreadsheet, and the second author verified the consistency and completeness of the extracted fields. Each extracted attribute was defined prior to analysis and aligned with the research questions.

I first extracted general metadata to support the descriptive and bibliographic analysis of the selected corpus. As shown in Figure 7.2, the distribution by year of the selected studies reflects the recent emergence of LLM usage in software engineering: no papers were published before 2024, nine studies in 2024 and six published in 2025. This confirms that the integration of LLMs into repository mining is a very recent phenomenon.

I also analyzed the types of venues in which the studies appeared. As illustrated in Figure 7.3, the corpus includes 10 conference papers and five journal articles. This balance reflects both the exploratory nature of the topic, often first reported in conference venues, and its growing academic maturity, as evidenced by the presence of journal publications.

For $\mathbf{RQ_1}$, which investigates the practices and approaches adopted for applying LLMs to MSR, I collected methodological and technical details about the application of LLMs. Specifically, I recorded the primary focus of the study (e.g., issue classification, Self-Admitted technical debt detection), the software repositories analyzed (e.g., GitHub, Hugging Face), and the LLM employed, distinguishing between proprietary (e.g., GPT, Claude) and open-source LLMs (e.g., LLaMA, Mistral). I also extracted detailed information about the prompt engineering strategies used, including the prompting pattern (e.g., zero-shot, few-shot, chain-of-thought), techniques, and structural elements (e.g., objective, task). Additionally, I captured the activities performed using LLMs, whether the authors followed or referred to existing LLM usage guidelines, and how the output of the model was validated, such as through manual annotation, ground truth comparison, or statistical agreement. Evaluation metrics (e.g., Accuracy, F1-score, API cost) and configuration parameters (e.g., temperature, max tokens, top-k) were also recorded to characterize reproducibility.

To address **RQ$_2$**, which focuses on threats and limitations in using LLMs for MSR, I extracted all information explicitly describing encountered barriers or risks. This included threats related to model behavior (e.g., hallucinations, instability, or high cost), methodological concerns (e.g., reproducibility or scalability), and any threats to validity reported by the articles. I paid particular attention to issues related to the trustworthiness and generalizability of LLM-generated outputs, which are often cited concerns in LLM4SE studies.

7.2.2 *Questionnaire Survey*

Simultaneously, I conducted a *questionnaire survey study* to complement the evidence gathered through the rapid review and capture insights from researchers actively working with LLMs in software repository mining. Following established guidelines for empirical surveys in software engineering [90, 198], I designed the instrument to ensure clarity, reliability, and methodological rigor. The questionnaire survey approach was particularly suited to the objective of exploring how LLMs are used in practice, given the fast-evolving nature of the field and the limitations of relying solely on published literature to capture emerging threats and practices.

7.2.2.1 *Design of the Online Questionnaire Survey*

In designing the questionnaire survey study, I adhered to the guidelines outlined by Dillman et al. [90] and Kitchenham et al. [198]. Dillman et al.[90] tailored design method offers a structured framework for improving both response rates and the overall quality of questionnaire survey data, emphasizing principles such as clear question formulation, personalized communication, and respondent engagement. In parallel, Kitchenham et al.[198] empirical guidance for conducting surveys in software engineering informed the decisions regarding questionnaire structure, sampling strategies, and mitigation of bias.

To uphold ethical standards and ensure participant anonymity, I provided an informed consent statement at the beginning of the survey. The

questionnaire survey design was reviewed and approved by the Ethical Committee of the University of Salerno. I deliberately avoided collecting any personally identifiable information such as name, email, gender, or age. Instead, I focused on collecting role-specific and organizational context data to characterize respondents' experience levels. To further increase the reliability of the dataset, the questionnaire survey instrument included embedded attention check questions to identify and later exclude responses provided carelessly or without engagement.

To validate the questionnaire survey, I employed an iterative pilot-testing process aimed at ensuring clarity, completeness, and accurate estimation of response time. The pilot was conducted with four participants, each having at least one year of experience working with LLM technologies in a software engineering context. These individuals, distinct from those participating in the final study, were selected for their domain relevance and ability to offer constructive critique. Their feedback was used to refine the questionnaire by eliminating redundant items and correcting typographical issues.[4] I also measured the average time required to complete the survey, which was about 10 minutes, which is well below the goal of balancing coverage and burden on respondents.

7.2.2.2 *Structure of the Questionnaire Survey*

The questionnaire survey was designed as a single instrument comprising five thematic sections, with two sections mapped to the research objectives. The structure balanced closed-ended questions, enabling quantitative analysis, with open-ended questions to capture deeper insights into participants' practices and threats. The full list of survey questions is provided in the online appendix [236].

1. **Consent and Background Information:** The questionnaire survey began with an informed consent form outlining the study's purpose, confidentiality measures, and voluntary participation conditions.

[4] More specifically, some redundant questions were removed, and typos that were identified were corrected.

Participants were required to confirm their consent before proceeding. The section then gathered information on the respondent's current role in research or industry, experience with empirical software engineering, and previous involvement in LLM-based repository mining. Respondents also specified which LLMs and software repositories they had worked with (e.g., GPT-4, Claude, GitHub, StackOverflow). An attention check was included to filter careless responses. To ensure that only relevant respondents proceeded, I included a screening question: "Have you conducted studies on the use of LLMs for software repository analysis?" Participants who answered "No" were automatically redirected to the end of the questionnaire survey, and their responses were not recorded.

2. **Practices in LLM-based Repository Mining:** This section, aligned with RQ_2, focused on how participants apply LLMs in practice. It covered prompting techniques (e.g., zero-shot, few-shot, chain-of-thought), types of tasks performed, prompt structure, evaluation metrics, automation of pipeline components, and whether established guidelines or best practices were followed. Participants were also asked to describe how they validate LLM outputs and how they configure model parameters (e.g., temperature, top-k).

3. **Threats and Limitations:** This section addressed RQ_1 and focused on practical and methodological obstacles encountered when using LLMs. Participants rated the frequency of various threats, such as hallucinations, high cost, reproducibility issues, and scalability limitations, and were asked to identify the most significant threats they had faced. Open-ended questions invited them to elaborate on mitigation strategies and threats to validity they had observed.

4. **Perceptions and Future Needs:** This section asked participants to rate, using a five-point Likert scale, the potential usefulness of structured guidelines for improving LLM-based repository mining. Respondents were also asked to suggest tools, resources, or forms of

support they believed would be most beneficial in improving their current practices.

5. **Final Reflections:** The questionnaire survey concluded with an open-text question inviting participants to share any additional experiences, insights, or observed bad practices related to the use of LLMs in software repository contexts. The final screen displayed a short note of appreciation for their time and contribution.

7.2.2.3 *Questionnaire Survey Sample and Procedure*

For participant selection, I adopted a convenience sampling strategy [15, 148], which is commonly employed in SE research when targeting specialized populations. This non-probabilistic approach relies on the availability and accessibility of participants, and although it limits the generalizability of findings, it is particularly effective when seeking input from domain-specific experts.

I targeted researchers with experience in applying LLMs to software repository mining tasks. Specifically, the inclusion criteria required participants to (i) have conducted or supervised empirical studies involving LLMs for repository analysis, and (ii) be currently active in software engineering research. These criteria ensured that participants could provide informed and contextually grounded responses. To reach this audience, I employed a three-front recruitment strategy:

- I disseminated a call for participation through professional social media channels, primarily *LinkedIn*[5], targeting academic and practitioner communities interested in LLMs and mining software repositories.

- I directly invited researchers from our professional networks, focusing on groups known to be actively engaged in LLM-based software engineering research.

[5] https://www.linkedin.com/feed/update/urn:li:activity:7326243068523769856/

- Finally, I contacted the corresponding authors of the primary studies identified during the rapid review (see Section 7.2.1).

Participation in the questionnaire survey was voluntary and anonymous. Respondents were required to provide informed consent before beginning the questionnaire, and no personally identifiable information was collected. The questionnaire survey was hosted on Google Forms and remained open for a period of two months. To ensure data reliability, the survey included an attention check question, and only responses from participants who reported direct experience with LLM-based repository mining were retained for analysis.

Starting from a candidate pool of **143** researchers identified through the recruitment strategies, I distributed the questionnaire survey using an open invitation strategy, targeting individuals with relevant expertise in LLM-based repository mining. The goal was not to achieve statistical representativeness, but rather to gather diverse and informed perspectives from researchers directly engaged in this emerging area. Responses were collected iteratively until no substantially new themes emerged across open-ended responses, a principle aligned with theoretical saturation [300], where additional data no longer yields new insights.

In total, **37** valid responses were collected and retained for analysis. Among the 37 participants, only one participant answered the question about attention-check (see next subsection) incorrectly and was removed from the survey. In total, **22** respondents confirmed their experience with the application of LLMs in repository mining and have passed the eligibility check embedded in the questionnaire survey.

To provide context for the questionnaire survey findings, participants held a variety of research positions, with the majority being PhD students (9 participants, 40.91%). The rest of the sample included Associate Professors (4 participants, 18.18%), Assistant Professors or Lecturers (3, 13.64%), and Full Professors (2, 9.09%). Other roles were represented by single participants each (4.55%), including Academic Researchers (non-faculty), Postdoctoral Researchers, Industry Researchers, and one respondent who identified as both a PhD Student and an Academic Researcher

(non-faculty). Participants reported varying levels of experience in Empirical Software Engineering. The most common range was 3–5 years (9 participants), followed by 6–9 years (5 participants). A smaller number of respondents reported less than 2 years (4 participants) or more than 10 years (4 participants) of experience in the field. When asked about their prior experience conducting LLM-based studies in MSR, most participants reported having conducted 2–3 studies (11 participants), followed by those who had conducted a single study (8 participants). A smaller group (3 participants) indicated having conducted more than 3 studies.

7.2.2.4 *Questionnaire Survey Data Extraction and Analysis*

After collecting the questionnaire survey responses, I performed a data cleaning process to ensure the reliability and validity of the dataset. I manually reviewed each submission to assess completeness, internal consistency, and attentiveness. This step is essential in survey studies, especially in online environment, where inattentive or low-effort responses can compromise data quality [198, 323]. Additionally, I included an attention-check question in the questionnaire survey to identify disengaged participants. The flagged responses were jointly reviewed by the first and second authors, and final inclusion decisions were made by consensus, as recommended in qualitative questionnaire survey reliability procedures [76]. Once the dataset was validated, I analyzed the responses aligned with the research questions. Responses to closed-ended questions were analyzed using descriptive statistics, including frequency distributions and visual aggregations, to identify patterns across participant practices, tool usage, prompt configurations, and validation strategies. This analysis enabled a structured understanding of how LLMs are currently being used in MSR. Open-ended responses were reviewed collaboratively by the first and second authors using qualitative content analysis [202]. Through iterative reading and topic-driven grouping, I summarized responses into thematically consistent categories. This method supports the systematic description of unstructured textual data and is well-established in empirical software engineering for interpreting practitioner feedback [76, 294].

The results from these two complementary analyses were then mapped to the research questions.

RQ$_1$ was addressed through questions in Section 2. These questions explored the types of prompts used, the specific LLMs and repositories involved, the activities performed, parameter settings, validation methods, automation practices, and the metrics employed. **RQ$_2$**, which focuses on the threats that researchers encounter when using LLMs for repository mining, I gathered insights through questions in Sections 3 and 4 of the survey. These questions asked participants to describe their most significant threats and the strategies they employed to overcome them.

7.3 ANALYSIS OF THE RESULTS

This section presents the synthesized findings from the mixed-method study, combining a rapid review of 15 peer-reviewed articles with a questionnaire survey of 22 experienced researchers. By triangulating evidence from published literature and practitioner experience, I address the research questions on the approaches and threats in LLM-based mining of software repositories. Although all responses were collected anonymously, for readability, I assign participant identifiers (R1–R22) when quoting or referring to individual answers.

7.3.1 *Context of the Analyzed Studies: Research Topics and Platforms*

Before detailing the methodological approaches, I first characterize the context from which the evidence is drawn. The analysis of the literature and questionnaire survey responses reveals that LLM-based MSR is being applied to a consistent set of research themes. The most predominant theme in the rapid review papers is **Code Quality, Maintenance & Security**, which was the focus of 6 out of 15 studies, covering topics like detecting green architectural tactics [82], code smells [316], code-comment inconsistencies [393], self-admitted technical debt [309], security vulnerabilities [369], and notebook executability issues [258]. The second most common

theme is **Developer Communication & Sentiment**, addressed by five studies that analyzed emotions [180, 193], sentiment [306, 392], and figurative language [54] in developer discourse. This is followed by **Repository Intelligence & Metadata**, with three studies focusing on creating datasets [185], analyzing software failures from news [14], and building repository chatbots [3]. Finally, one study focused on **Issue Management**, specifically on issue report classification [71].

This distribution of research topics is strongly mirrored in the activities reported by the survey respondents. A majority of practitioners (**13 out of 22**) focus on topics related to **Code Quality, Maintenance, and Security**, with respondents listing activities such as "Code smell detection, technical debt", "MSR for security", "defect prediction", and "Architecture reconstruction". **Issue Management** was also a common focus, mentioned by **seven** respondents with tasks like "issue report classification" and "bug localization".

The platforms analyzed for these studies are detailed in Figure 7.4. The results show that, while some practices are consistent, others highlight emerging trends. In both the published literature and the practices reported by the questionnaire survey respondents, **GitHub** emerges as the predominant source for MSR data. However, a notable divergence appears with model-sharing platforms. **Hugging Face** and **PyTorch Hub** are frequently used by researchers in the questionnaire survey, reflecting a growing interest in analyzing ML models and their ecosystems, but they were not mentioned as primary data sources in the reviewed literature. Platforms for developer communication and issue tracking, such as **Stack Overflow** and **Jira**, appear with low frequency in both data sources. Finally, other specialized sources like **Google Play**, **Gerrit**, and **Etherscan** were used exclusively in the reviewed literature for task-specific analyses.

7.3.2 *RQ_1: What approaches are employed when using large language models for repository mining?*

The first research question aimed to investigate the methodological approaches adopted by researchers when using LLMs in MSR studies. To further improve the clarity and interpretability of the findings for **RQ_1**, I organized the identified practices in stages of the LLM-based research process. This staging builds upon the earlier conceptualization, which initially included four phases of PRIMES [80]. However, it required refinement to better align with empirical evidence from the literature and questionnaire survey data. The updated structure introduces two entirely new stages and refines three existing ones. 'Stage 0: Strategic Planning & Preparation' is newly introduced to explicitly account for preliminary tasks, such as defining study objectives, selecting methodological guidelines, curating datasets, and integrating automation and tooling for scalable validation workflows. These activities were previously implicit or considered in other stages but emerged as distinct and recurrent decisions in both the reviewed studies and practitioner reports. *"Stage 1: Creation of Prompts for Piloting"* remains conceptually consistent with the original framework. In *"Stage 2: Prompt Validation"*, I refined the title to better reflect the process of including the human in the loop. *"Stage 3: Large Language Models Setup"* is a newly introduced stage, reflecting the configuration of LLMs, consideration of managing ensemble strategies, documenting LLM parameters, and ensuring reproducibility. *"Stage 4: LLM Comparison"* updates an earlier stage of PRIMES and focuses on the comparative evaluation of candidate models to select the most suitable one for the main study. Finally, *"Stage 5: Execution, Validation & Synthesis"* is a new, comprehensive stage covering the large-scale execution on the full dataset, the subsequent validation of outputs through error correction and post-processing, the final analysis of results, and the crucial practice of publishing replication packages. Table 7.3 summarizes this structure, mapping each stage to the methodological approaches identified in the study. Although presented sequentially for clarity, the overall process is highly iterative in practice, with

researchers frequently revisiting earlier stages as new insights or failures emerge during execution.

Table 7.3: Summary of the stages and methodological approaches of LLM4 MSR research. Categories marked with (*) are newly introduced in this version; those marked with (†) represent revisions of categories from the original PRIMES framework [80].

Stage	Description	Identified Approaches
Stage 0: Strategic Planning & Preparation *	Defining research goals, considering established standards, automating and tracking the workflow, and preparing the dataset before model interaction.	**A1: Use of Existing Guidelines**
		A2: Goal and Task Definition
		A3: Dataset Preparation and Curation
		A4: Automation & Track the Process
Stage 1: Creation of Prompts for Piloting	Designing the initial prompt structure, specifying output formats, and selecting an LLM to perform pilot testing.	**A5: Selecting Prompting Strategy**
		A6: Select Large Language Models
Stage 2: Prompt Validation †	Evaluating and refining the prompt using pilot runs, agreement metrics, and annotated samples.	**A7: Iterative Prompt Refinement and Pilot Testing**
		A8: Evaluation Metrics
Stage 3: Large Language Models Setup *	Selecting, configuring, documenting, comparing multiple LLMs used in the study, and considering the adoption of ensemble strategies.	**A9: LLM Reporting Configuration**
		A10: Consider LLMs Ensemble
Stage 4: LLM Comparison †	Benchmarking LLMs using validated prompts to select the optimal LLM based on predefined metrics on an oracle.	**A11: Output Validation Methods**
		A12: Benchmarking and Comparative Analysis
Stage 5: Execution, Validation & Synthesis †	Executing the chosen LLM at scale, validating outputs through post-processing and error correction, synthesizing final results, and publishing a replication package.	**A13: Large-Scale Execution**
		A14: Result Analysis and Synthesis
		A15: Open-Source Replication Packages

STAGE 0: STRATEGIC PLANNING & PREPARATION * This phase lays the foundations for the study by aligning the research objectives, methodological choices, and resource organization. It ensures that

subsequent activities are based on a coherent framework that promotes consistency, scalability, and reproducibility.

⚙ **A1: Use of Existing Guidelines.** Adhering to established methodological guidelines is a fundamental approach to ensuring rigor in LLM-based MSR and, more broadly, in empirical software engineering research. The rapid review suggests that adherence to formal guidelines is not yet a common practice in published work. Only two studies (out of 15) explicitly cited external guidelines, both referring to best practices from OpenAI [3, 306]. One study [180] explicitly derived its prompt design from existing studies in the literature. The majority of papers (13) developed their methodologies without referencing established standards.

In the questionnaire survey, I asked participants whether they follow any guidelines or best practices when using LLMs. Out of 22 responses, the largest group, comprising 10 participants (45.5%), stated they were unaware of such guidelines or responded with 'NA', highlighting a gap in awareness or accessibility of structured guidance. Only 2 respondents (9.1%) reported following established frameworks or formal guidelines, including references to the LLM4SE guidelines [355] and the PRIMES framework [80]. A larger portion, 7 participants (31.8%), reported adherence to informal best practices, such as those recommended by OpenAI or Google, or general principles of prompt engineering and research ethics. Finally, 3 respondents (13.6%) described partial or exploratory adoption, indicating either inspiration from related work or intentions to follow formal practices in future studies.

While a minority of researchers follow formal frameworks or guidelines, a large group adopts informal strategies, and many researchers continue to operate without structured methodological support. This underscores the need for better dissemination and adoption of industry-specific LLM guidelines in empirical software engineering.

⚙ **A2: Goal and Task Definition.** Another essential methodological step, preceding any technical implementation, is the strategic definition of research objectives. This initial planning phase, which directly guides

prompt creation and the overall study design, was consistently identified as the starting point in the data. Survey respondents explicitly listed activities like *"Define the goal of the analysis"* (R1) and *"Define Task"* (R13) as the first actions in their workflows. This approach ensures that all subsequent methodological choices, from data curation to model selection, are aligned with a clear and well-defined research goal.

In line with these goals, studies typically begin by identifying and collecting a diverse range of software artifacts that match the targeted research questions. Published work analyze code-level data, such as entire code repositories to identify architectural tactics [82], individual classes to detect code smells [316], smart contract source code for security vulnerabilities [369], and code comments to find self-admitted technical debt [309] or inconsistencies [393]. Other works focus on developer communication and process artifacts, including issue reports for classification [71] and discussions in commits or pull requests for sentiment and emotion analysis [54, 180, 392]. The scope extends even to non-traditional artifacts like computational notebooks for executability analysis [258], model cards from platforms like Hugging Face for metadata extraction [185], and external sources like online news articles to analyze software failures [14].

A3: Dataset Preparation and Curation. Following the definition of research objectives, a critical preliminary step is the preparation and curation of the dataset. The review of the literature shows that studies consistently perform a data collection and preprocessing phase before engaging with any LLM.

This involves identifying and gathering relevant software artifacts. The scope of these artifacts is broad, ranging from granular code-level data to project-wide communications and even external sources. For instance, studies have focused on individual code comments to identify Self-Admitted Technical Debt [309], specific code snippets containing known smells [316], and function-level code with its accompanying comments to detect inconsistencies [393]. Other research analyzes artifacts at a

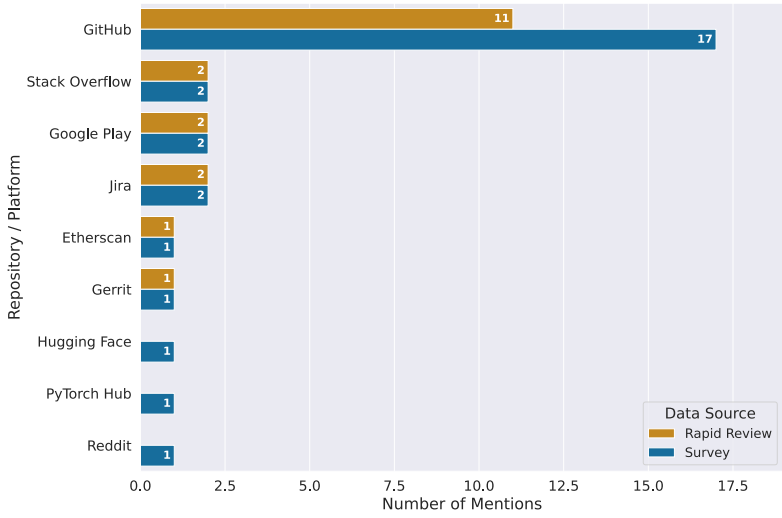


Figure 7.4: Frequency of software repositories and platforms analyzed in the rapid review (N=15 papers) and survey (N=22 respondents).

larger scale, such as entire code repositories to find green architectural tactics [82] or smart contracts to detect security vulnerabilities [369]. A significant portion of studies also targets developer communication, analyzing issue reports from bug tracking systems [71], pull request discussions [392], and general developer messages to understand emotions [180] or figurative language [54]. The spectrum of artifacts also includes more specialized data, such as computational notebooks to assess their executability [258], model cards from platforms like Hugging Face [185], and even news articles reporting on software failures [14]. Beyond simple collection, this phase often includes significant data curation efforts, such as cleaning noisy text, filtering for relevant samples, and preprocessing the data (e.g., through truncation or dataset balancing) to create a high-quality input corpus suitable for the LLM [309].

Beyond simple collection, this phase often includes significant data curation efforts, such as cleaning noisy text, filtering for relevant samples, and preprocessing the data (e.g., through truncation or dataset balancing) to create a high-quality input corpus suitable for the LLM [309].

This practice is strongly confirmed by the questionnaire survey respondents, who identify it as important part of their workflow. Researchers explicitly listed activities such as *"data collection, pre-processing"* (R17) and *"Input collection"* (R2) as essential steps that precede prompt engineering. This stage ensures that the data fed to the LLM is clean, relevant, and structured, which is a prerequisite for designing effective prompts and obtaining reliable results.

⚙ **A4: Automation & Track the Process.** To manage the complexity of the entire workflow, from request to validation, researchers create a pipeline using customized automation tools. This practice is consistently observed in the literature and the questionnaire survey. Many studies report the development of bespoke systems, such as FAIL [14], PonziSleuth [369], and RustT4 [393], or novel mining mechanisms [82, 185] to orchestrate the analysis. This is corroborated by the survey respondents, who describe *"Python scripts, and I create an automated pipeline that runs all the phases"*(R2) to drive the entire research workflow, the same mechanisms are used in literature [82, 316]. A primary target for automation is the handling of inconsistent LLM outputs. Researchers frequently build scripts to validate and parse responses, often employing fallback mechanisms. For example, one practitioner detailed a common strategy: *"The output validation and extraction are performed by parsing the output automatically (if it has the requested structure); otherwise, I use regular expressions to find an answer"* (R15). This approach is documented in the literature, where regular expressions are used to reliably extract data from malformed model outputs [3]. Beyond parsing, some researchers leverage automation for the creative process of prompt engineering itself, including using LLMs for *"automatic prompt engineering"* (R7) or applying *"meta prompting when generating prompts"* (R3) to improve their quality iteratively.

This custom tooling often integrates with other libraries and analysis techniques. For instance, questionnaire survey respondents report combining their scripts with standard repository mining libraries, such as *"PyDriller or simple GraphQL queries"* (R3). Similarly, the literature de-

scribes pipelines that integrate LLMs with static analysis [369, 393] or automated execution frameworks, such as Papermill [258]. This investment in custom automation is critical not only for managing complexity but also for enhancing scientific rigor. By codifying the entire experimental workflow, these automated pipelines effectively act as an executable specification of the research method. They ensure reproducibility by capturing the exact versions of the dataset, the final **prompts**, the complete LLM configuration (including model version, temperature, and seed), and the logic for output parsing and validation. This makes the entire research process more transparent and replicable by researchers.

STAGE 1: CREATION OF PROMPTS FOR PILOTING This phase defines the prompt structures, specifying output formats, and selecting initial models for piloting. It enables early iterations to align the model's behavior with the research goals and constraints.

⚙ **A5: Selecting Prompting Strategies.** The design of prompts is one of the most important methodological aspects. In the reviewed literature, several distinct strategies have been employed to train LLMs. These strategies are also highlighted by Hou et al. [166] as follows:

- **Zero-shot prompting**, where the model is expected to perform a task without any explicit training or examples, relying solely on the provided instructions [280]. This was the most common pattern, used in 12 of the 15 reviewed studies [14, 54, 71, 180, 193, 258, 306, 309, 316, 369, 392, 393], underscoring a reliance on the out-of-the-box capabilities of powerful models.
- **Few-shot prompting**, which involves providing a limited number of examples within the prompt to help the model learn the task and generalize to similar cases [32]. This was the second most common technique, used in six studies to improve performance with in-context examples [54, 71, 193, 306, 309, 392].
- **Chain-of-Thought (CoT) prompting**, an advanced technique that guides the model to break down a problem into intermediate rea-

soning steps, enhancing its ability to tackle complex tasks [361]. This emerging strategy was identified in three reviewed studies [14, 71, 369].

Beyond selecting a single prompting strategy, an identified approach in the review is the **use of multiple prompts** to achieve a research goal. A common pattern is **Prompt Chaining** [6], where a complex task is decomposed into a sequence of simpler prompts. For instance, one prompt might first classify an artifact (e.g., an emotion), and a second prompt then uses that output to perform a more detailed analysis (e.g., extract the cause of the emotion) [180, 369].

The survey data shows similar findings. **Zero-shot prompting** remains the most widespread technique (used by 20 of 22 respondents), often serving as a baseline. However, practitioners frequently adopt more complex strategies to improve reliability. For instance, the concept of prompt chaining is mirrored in practice, with one researcher describing how they *"split the prompts into multiple prompts, making the LLM go into the repository gradually"* (R5) to build context incrementally. Furthermore, to ensure predictable and parsable outputs, researchers are increasingly adopting **Structured Chain-of-Thought (SCoT) Prompting** [212], which was used by 18 of 22 respondents. The motivation is to gain control over the output, as one respondent aims to *"tailor the prompts with detailed descriptions to generate most consistent (at least correct) results"* (R11). This is followed by **Few-shot prompting** (17 of 22) and standard **Chain-of-Thought prompting** (13 of 22), where researchers guide the model by asking it to, for example, *"explain the classes involved and their interaction"* (R5) before providing a final answer. This suggests that while zero-shot is a common starting point, researchers in MSR actively employ a range of sophisticated prompting methods to control model behavior and ensure reliable results.

[6] Chain complex prompts for stronger performance https://docs.anthropic.com/en/docs/build-with-claude/prompt-engineering/chain-prompts

⚙ **A6: Select Large Language Models.** With a prepared dataset, the next approach is the selection of the LLM itself. A wide array of LLMs, both proprietary and open-source, were utilized in the literature.

- **Proprietary Models:** The landscape is dominated by OpenAI's GPT series. **GPT-4** and its variants (e.g., GPT-4o, GPT-4-turbo) were used in nine studies [3, 14, 82, 180, 185, 193, 306, 392, 393], and **GPT-3.5-turbo** was used in seven studies [14, 71, 180, 185, 306, 316, 369]. This highlights a reliance on high-performing, commercially available models. Other models like Anthropic's **Claude** series [82] and Google's **Gemini** family [193] also appeared, suggesting diversification.
- **Open-Source Models:** There is a growing adoption of open-source alternatives. Smaller, encoder-only models (sLLMs), such as **BERT**, **RoBERTa**, and **CodeBERT**, were frequently used as baselines or for fine-tuning in four studies [54, 306, 309, 392]. Larger, decoder-only open-source models like **LLaMA** [258, 369, 392], **Flan-T5** [180, 309], **Mistral** [193, 369], **Vicuna** [392], and **Qwen** [193] were also employed, indicating a trend towards leveraging models that offer greater transparency and customization.

This trend of using a diverse set of models is directly reflected in the practices of the survey respondents. Confirming the dominance of proprietary solutions, GPT-4 was the most-used LLM, employed by 68.2% of researchers (15 out of 22). Close behind, LLaMA 3 was reported by 11 respondents (50%), highlighting a strong adoption of open-weight models, particularly Meta's LLaMA series. This strong adoption is likely driven by several factors that directly address key threats identified in the study. Open-weight models provide a practical solution to the high financial costs and scalability issues associated with proprietary APIs, two of the most frequently reported threats by the survey respondents. Furthermore, running models on local or controlled infrastructure provides researchers with greater control over the experimental environment, thereby enhancing reproducibility by mitigating the model drift that is common in commercial APIs. The ability to fine-tune these models on

domain-specific data, combined with the competitive performance of recent releases, makes them a compelling alternative for research.

GPT-3 remains widely used, cited by nine participants (40.9%), likely due to legacy system integration or accessibility. Claude 3 (27.3%) and Gemini 2.5 Pro (22.7%) were also frequently selected, reflecting the growing competitiveness of Anthropic and Google's offerings in the LLM ecosystem. Other models showed more niche adoption:

- Qwen and Gemma were each mentioned three times;
- Mixtral, Deepseek v1, and DeepSeek V3 mentioned two times;
- while Claude 3.5 Sonnet, Gemini 1.5 Pro, Gemini 1.5 Flash, Gemini 2.0 Flash, GPT4-o, CodeQwen, GPT-o4-mini, Deepseek-coder, DeepSeek, codegeex, codellama, starling-lm, phi, starcoder, mistral, and zephyr were each mentioned one time.

Finally, custom models, including those trained internally, were reported by three participants, indicating that while most researchers rely on pre-trained foundation models, a subset is investing in domain-specific or fine-tuned solutions tailored to their specific needs.

These results underscore a dual trend: the continued dominance of powerful general-purpose models like GPT-4, and a diversification toward open-weight and customizable alternatives.

Most importantly, the process of choosing among these models is often empirical, as articulated by one researcher whose workflow includes an explicit step to *"compare the final prompt on multiple LLMs, and select the final LLM"* (R1). This phenomenon is also highlighted in literature [71, 82, 185, 316, 369]. This indicates a shared landscape where researchers leverage commercial models and open-source alternatives for transparency, often making their final selection based on comparative evaluation.

STAGE 2: PROMPT VALIDATION † This phase focuses on evaluating and refining the prompt through small-scale tests. It introduces human-

in-the-loop validation and agreement measures to ensure the prompt produces reliable and interpretable outputs.

A7: Iterative Prompt Refinement and Pilot Testing. A crucial approach that bridges initial prompt design and full-scale analysis is the practice of iterative refinement through pilot testing. Rather than immediately applying a designed prompt to an entire dataset, **researchers execute it on a small but representative sample with a single LLM**. This pilot sample often serves as a "gold standard" or "oracle" for validation. For instance, studies in the review created validation sets ranging from a few dozen instances, such as 31 software failure incidents [14] or 50 model cards selected via stratified sampling [185], to a set of 10 diverse projects [82]. This hands-on, experimental loop is clearly described by survey respondents, with one detailing their process as a cycle of *"Define Task... Experiment with Prompts... Redefine Prompt (go back to step 2, repeat)"* (13) and another characterizing it as a phase of *"trial and errors on a subset"* (7) while other refined the prompt asking directly to the LLMs used *"I've started with a base prompt that I improved by asking ChatGPT directly"* (R4) or *"Prompts were sometimes refined leveraging LLMs"* (R16).

This practice is consistently reported in the literature, where researchers conduct pilot sessions to iteratively refine the wording, structure, and examples of prompts based on observed LLM behavior and output quality [82, 185, 316]. The core of this approach involves manually inspecting the pilot results to identify issues like hallucinations, incorrect formatting, or misinterpretations, and then adjusting the prompt accordingly. A detailed workflow provided by one respondent includes: *"Initial prompt drafting; Validation of the Draft's Results; Prompt Refinement; Prompt Execution on a small subset"* (R3).

While refining prompts using a single LLM may introduce bias when results are later compared across multiple models, this remains a practical and effective strategy for aligning prompt behavior with the

research goals. This iterative validation ensures that the prompt is robust and reliable before being used for large-scale execution, thereby saving time, reducing costs, and minimizing the risk of collecting low-quality data.

A8: Evaluation Metrics. To quantify the results of the validation process, researchers employ a well-established set of metrics, largely drawn from traditional classification, statistical comparison, and natural language generation tasks. The activity is reported as *"Prompt Execution on a small subset, Results evaluation, Refinement(if necessary)"* (R2). The reviewed literature consistently features standard classification metrics such as **Accuracy**, **Precision**, **Recall**, and **F1-score** [309, 316, 393]. These are often complemented by statistical tests to compare model or prompt performance, such as **McNemar's test**, often reported with **Odds Ratio** and **Absolute Risk Difference** [306, 316], or the **Wilcoxon signed-rank test** [392].

For tasks involving text generation or assessing semantic similarity, a different set of metrics is used. These include **BLEU** for evaluating generated text [180], **cosine similarity** for comparing sentence embeddings [54], and the Area Under the Curve (**AUC**) for evaluating classifier performance [392]. Inter-rater reliability for manual validation is commonly measured with **Cohen's Kappa** [71, 309]. Furthermore, some studies adopt highly task-specific metrics, such as the **executability ratio** for notebook restoration [258] or qualitative **taxonomy agreement** for analyzing failure reports from news [14]. The survey data shows a strong consensus on the use of these core metrics. The most frequently reported metric was **Precision**, mentioned by 16 out of 22 respondents, followed closely by **F1-score** (15 respondents), **Accuracy** (14 respondents), and **Recall** (14 respondents). These were often used in combination to provide a comprehensive view of model performance. Beyond standard classification, a subset of researchers considered more resource-oriented measures. **API Cost** was reported by four respondents, often to compare the efficiency of different strategies, while **inference time** and the

number of tokens used were also cited. Other specific metrics mentioned by participants included *BLEU*, *Mojo distance*, *Exact Match*, and *cosine similarity*, aligning with the diverse toolkit found in the literature.

Finally, to move beyond purely quantitative performance, researchers leverage qualitative and explanatory methods. These include **thematic analysis** for interpreting subjective outputs, as reported by the survey respondents, and the use of Explainable AI (XAI) techniques like **SHAP explanations** to understand the drivers of model predictions [306]. These findings suggest a growing need to complement quantitative evaluations with interpretability and cost-awareness, especially as LLMs are applied to increasingly diverse MSR use cases.

STAGE 3: LARGE LANGUAGE MODELS SETUP * This phase involves selecting, configuring, and documenting the LLMs used in the study. It includes decisions about model versions, parameter settings, and ensemble strategies to ensure reproducibility.

A9: LLM Reporting Configuration. Once a model is selected, the next step is to define its operational parameters to ensure reproducibility and control its behavior. To enhance reproducibility, details on LLM parameter settings were often provided in the literature. For inference-based tasks, the `temperature` parameter was the most frequently configured, commonly set to zero or a low value to ensure deterministic and factual outputs [3, 14, 71, 306, 309, 392]. Parameters controlling response length, such as `max_tokens` or `max_new_tokens`, and overall context window size were also frequently specified to manage cost and output structure [71, 185, 309, 316]. For studies involving model adaptation, researchers reported fine-tuning hyperparameters, including the number of `epochs`, the `learning rate`, and the `batch size` to optimize performance on specific SE datasets [306, 309, 392]. On the other hand, other studies do not de-

fine the specified parameters in the article or use default parameters [54, 82, 180, 258].

The configuration practices reported by the survey respondents align closely with these findings, particularly in terms of managing determinism and cost. The most common practice mentioned was controlling the `temperature` (nine respondents). A majority of respondents who configure parameters reported setting the temperature to a low value or zero to ensure deterministic and reproducible outputs. One researcher provided a task-dependent heuristic: using a *"low temperature for classification, higher temperature for generation."* Controlling `max_tokens` was the second most cited strategy, used both to manage API costs and to ensure concise outputs for specific tasks. As one participant noted, for classification tasks, *"putting a tailored constraint in the max token can help"* limit costs. Several respondents (six respondents) reported not altering any default settings. This suggests that for some researchers, default behavior is considered sufficient, or that configurability is limited by tooling or time constraints.

Beyond direct tuning, a few participants mentioned relying on guidance from literature, whitepapers, or official API documentation to inform their choices. A smaller group reported using more systematic approaches, *"I use reference from the literature, otherwise I will proceed with an ablation study or a configuration experiment"* (R2) or *"I use grid search to find the optimal value for a parameter"* (R15). Overall, these responses reveal a wide range of practices, from using default settings to a more deliberate, literature-based configuration. Although not all studies or researchers systematically adjust parameters, those who do recognize the value of improving reproducibility.

A10: Consider LLMs Ensemble. Researchers may consider adopting an **ensemble of LLMs**, which involves combining the outputs of LLMs potentially differing in architecture, training data, or behavior,

to enhance robustness and accuracy. Instead of relying on a single model, this approach leverages model diversity to mitigate individual limitations and capture a broader range of correct outputs. For extracting structured metadata from model cards, this technique proved to be advantageous [185]. Additionally, respondents also corroborate this approach: *"sometimes multiple LLMs are needed when conducting multiple tasks at the same time"* (R10). To further improve the reliability and stability of LLM outputs, researchers employ **ensemble prompting**, where a single prompt is executed multiple times on the same input to mitigate randomness. The final result is then determined by aggregating the outputs, often through a majority vote [393] or by averaging performance scores across runs [71, 369], especially when deterministic accuracy cannot be ensured through a single run. This practice was highlighted by a survey respondent saying: *"run experiments multiple time and use a vote criterion to determine the accuracy"* (R18). This is complemented by the use of **output filtering and parsing**. This involves developing scripts or rules to validate the structure of the LLM's response. For example, researchers use regular expressions to extract valid data from malformed outputs [3] or implement post-processing steps to resolve inconsistencies and normalize the data [185]. If an output fails these checks, it is often discarded. As one researcher (R15) explained, if automated parsing fails, *"the output is considered null"*. Together, these techniques improve the reliability of the collected data.

STAGE 4: LLM COMPARISON † This phase focuses on the systematic comparison and selection of the final LLM for the study. It involves benchmarking the performance of different candidate models using the validated prompts and evaluation metrics on a representative data sample to make an evidence-based decision.

A.12: Benchmarking and Comparative Analysis. A common methodological practice is the use of benchmarking and comparative analy-

sis to justify model selection. Rather than selecting a single model a priori, many reviewed studies evaluated multiple LLMs to identify the best-performing or most cost-effective option for their specific task. This often involved running the same prompts and dataset through different proprietary models (e.g., comparing GPT-4 against Claude 3 Haiku [82]), different open-source models (e.g., LLaMA vs. Mistral [369]), or a mix of both [180, 193]. Some studies conducted extensive benchmarks, comparing over 20 different models to analyze trade-offs between performance and resource requirements [71], while others performed ablation studies to isolate the impact of different model choices within a larger system [369].

This practice of multiple model selection is described by the survey respondents, whose workflows often include an explicit model evaluation phase. One researcher described this step as needing to *"compare the final prompt on multiple LLMs, and select the final LLM"* (R1). This process based on performance metrics and also on practical constraints. For instance, another respondent highlighted the trade-off between cost and resources, explaining that they rely on *"local setups of LLMs (Llama, Qwen)"* because larger proprietary models are often too slow or expensive to run on their available hardware (R13). The use of ablation studies, noted in the literature, is also confirmed in practice, with one researcher stating they would proceed with an *"ablation study or a configuration experiment"* (R2) to justify their choices.

STAGE 5: EXECUTION, VALIDATION & SYNTHESIS † The final stage of the framework encompasses the main data generation, validation, and synthesis activities. It includes the large-scale execution of the selected LLM on the entire dataset, ensuring the reliability of the generated data through post-processing and error correction, analyzing and interpreting the final results, and publishing replication packages to ensure transparency and reproducibility.

A13: Large-Scale Execution. Once both the prompt and the LLM have been validated and finalized, typically through pilot testing and comparative evaluations, the next step is **large-scale execution**. This approach consists of the application of the finalized experimental setup to the entire target dataset. It represents the transition from methodological adjustments to the large-scale collection of the study's raw empirical data.

This involves applying the finalized prompt to every data point, which demands significant engineering effort. This is corroborated by the survey, where a majority of respondents (14 out of 22) reported that **"high costs"** and **"scaling to large datasets"** are threats they face 'Often' or 'Always'. Researchers must manage concerns such as API rate limits, batch processing, hardware requirements, and overall cost [82, 369]. One respondent articulated this difficulty clearly, stating, *"When scaling to large datasets, the inference time and hardware requirements are a serious problem to take into account"* (R15). The custom automation pipelines, discussed in Approach (**A14**), ensure a smooth and efficient execution in this phase. Studies in the review demonstrate this large-scale application on thousands of artifacts, such as news articles [14] or entire software repositories [82]. Therefore, large-scale execution is not merely a final 'run', but a distinct engineering approach where researchers must balance speed, cost, and reliability to generate the full raw dataset for their study.

A14: Result Analysis and Synthesis. The final approach in the workflow is the analysis and synthesis of the validated LLM-generated data. This phase transitions from data generation to knowledge creation, where the structured outputs are interpreted to yield empirical findings that address the study's original research questions. The specific techniques used depend on the nature of the data and the research goals. For quantitative data, such as classifications or counts, this often involves descriptive statistics to analyze the frequency and distribution of categories [82, 316]. For more qualitative or unstructured outputs, researchers apply techniques like thematic analysis or clustering (e.g., using DBSCAN)

to identify emergent patterns and group related concepts [180]. This final approach is a consistent part of the process described by survey respondents, who listed activities like *"result analysis"* (R17) and *"qualitative analysis through thematic analysis"* (R7) as part of their workflow. It is in this phase that the raw, machine-generated data is transformed into human-interpretable insights. For example, after using an LLM to identify thousands of software failures from news articles, one study performed a final analysis to identify trends in failure recurrence and severity over time [14]. This highlights that the LLM is often a powerful intermediate tool for data extraction and classification, but the ultimate research contribution comes from the subsequent analysis and synthesis performed by the researcher.

⚙ **A15: Open-Source Replication Packages.** An important approach for ensuring transparency and reproducibility, adopted by all 15 studies in the review, is the creation and public release of a comprehensive replication package. This approach ensures that the entire research process, from data collection to analysis, is verifiable and reusable by the community.

The analysis of these packages reveals a common set of platforms used for dissemination. **GitHub** is the most prevalent choice, used in 9 of the 15 studies for hosting source code and versioned artifacts [54, 180, 185, 193, 258, 309, 369, 392, 393]. This is often supplemented by long-term archival services, such as **Zenodo** (3 studies) [3, 14, 316] and **Figshare** (3 studies) [71, 82, 306], to ensure the persistent availability of the data.

The contents of these packages consistently feature artifacts for replication. For instance, the **source code** for the analysis is provided in 14 of the 15 packages, while the **datasets** used (including raw data, oracles, and final results) are also included in 14 studies. The provided code ranges from data processing scripts and Jupyter notebooks [309] to complete, installable tools that replicate the study's functionality [369, 393]. Most important for this line of research, 6 of the 15 studies explicitly include the exact **LLM prompts** or prompt templates, component for reproduc-

ing generation-based results [14, 71, 82]. This approach of providing comprehensive, publicly accessible artifacts is fundamental to the verifiability of LLM-based MSR, allowing other researchers to inspect the methodology, validate the findings, and build upon the work.

> **RQ_1 — Large Language Model Approaches for Empirical Rigor**
>
> *As a result of **RQ_1**, I identified a set of 15 methodological approaches used when applying LLMs for MSR. These include strategies for prompt design, model selection, iterative validation, and automation. These findings offer future researchers a detailed overview of the technical and methodological choices that characterize current LLM-based MSR studies. Researchers can design their studies effectively and align with emerging best practices in the field.*

7.3.3 *RQ_2: What are the current threats to validity faced by researchers when they use large language models for repository mining?*

The analysis reveals a consistent set of significant threats, triangulated between the published literature and the experiences of the survey respondents. In the following section, I discuss each threat and the strategies for mitigating them, summarized in Table 7.4.

T1: Hallucinations, Misinterpretation, & Output Inaccuracy. A predominant threat is the propensity of LLMs to generate factually incorrect information (i.e., hallucinations) or misinterpret the input data, an issue reported across a majority of the reviewed studies [3, 14, 54, 82, 180, 185, 193, 258, 306, 309, 316, 392]. This finding is strongly corroborated by the survey, where half of the respondents (11 out of 22) report experiencing hallucinations 'Often' or 'Always', with seven encountering them 'Sometimes'. In contrast, three respondents classified the problem as occurring 'Rarely', and just one had 'Never' experienced it. This issue was frequently cited as the most significant barrier, with one researcher

Table 7.4: Summary of Threats, Mitigation Strategies, and Employed Techniques.

Threat	Mitigation Strategy	Techniques Employed
❗ **T1: Hallucinations, Misinterpretation, & Output Inaccuracy**	**(M1) Grounding via Hybrid Approaches**	Static Analysis / AST Checks [369, 393]
		Knowledge Graph (KG) Querying [3]
		Retrieval-Augmented Generation (RAG) [14]
	(M2) Structured & Multi-Step Prompting	Chain-of-Thought (CoT) [14, 369]
		Context Enrichment (Schemas, Taxonomies) [3, 82, 180]
	(M3) Output Stabilization through Aggregation:	Majority Voting / Averaging Results [71, 369, 393]
		Vote Criterion for Accuracy (R18)
❗ **T2: Prompt Engineering & Sensitivity**	**(M4) Systematic Iteration & Oracle-Guided Refinement**	Pilot Testing & Comparative Evaluation [71, 82, 316]
		Iterative Workflow (R3, R7, R13)
	(M5) Adoption of De-Facto & Emerging Standards	Modular Component-wise Evaluation [193]
		Structured Templates & Guidelines [3, 82, 306]
	(M6) Meta-Prompting & Automated Assistance	Meta-Prompting (R3, R4, R7) [82, 185, 316]
❗ **T3: Contextual Understanding & Domain Specificity**	**(M7) Injecting Domain Knowledge**	Grounding in SE-specific Taxonomies [82, 180]
		Fine-tuning on Domain-specific Data [54]
	(M8) Hybrid Program Analysis	Pre-processing with Static/Taint Analysis [185, 369, 393]
	(M9) Intelligent Context Management	Automated Code Slicing [369]
		RAG & Rule-based Truncation [14, 71]
		Intelligent Chunking/Summarization (R17)
❗ **T4: Scalability, Efficiency, & Cost**	**(M10) Strategic LLM Selection**	Cost-Performance Analysis [82, 185, 309]
		Use of Local/Open-Source Models (R13)
		Model Quantization (R15)
	(M11) Input & Output Token Optimization	Input Reduction via Code Slicing [369]
		Output Capping with 'max_tokens' (R3, R19)
	(M12) Multi-Stage Hybrid Pipelines	Low-cost Pre-filtering (e.g., embeddings) [14]
❗ **T5: Reproducibility & Output Stability**	**(M13) Deterministic Parameter Configuration**	Setting 'temperature = 0' [14, 71] (R4, R7, R17)
	(M14) Stabilizing Stochastics through Aggregation	Averaging Results of Multiple Runs [71, 369]
	(M15) Transparent Reporting & Version Pinning	Documenting Specific Model Versions [14, 71]
❗ **T6: Output Formatting & Parsing**	**(M16) Prompt-Based Formatting Instructions**	Requesting Strict JSON Format [185, 369]
		Providing Few-shot Examples of Format [82]
	(M17) Robust Post-Processing & Fallback Parsing	Fallback Parsing with Regular Expressions [3] (R11, R15)
		Flexible Text Normalization [180]
	(M18) Manual Correction or Exclucion	Manual Fixing of Malformed Outputs [316]
		Discarding and Reporting Loss Rate [71]
❗ **T7: Dataset Noise, Imbalance, & Pre-training Leakage**	**(M19) Input Dataset Preparation & Cleaning**	Multi-Annotator Reconciliation [180]
		Manual Error Analysis to Correct Labels [309, 392]
	(M20) Bias Mitigation through Prompting & Analysis	Prompt-based Safeguards (e.g., persona) [193]
		Use of Interpretability Methods (R22)
	(M21) Acknowledgment of Data Leakage	Acknowledging Threat to Construct Validity [71, 185]
❗ **T8: Validation Complexity**	**(M22) Oracle Construction for Output Evaluation:**	Multi-Annotator Workflow & Reconciliation [180, 393]
		Reporting Inter-Rater Reliability [14]
	(M23) Scalable Validation via Statistical Sampling	Validating on a Statistically Significant Sample (R10, R16)
❗ **T9: Lack of Standardized Tooling & Infrastructure**	**(M24) Developing Custom Automated Pipelines**	Creating custom Python scripts & pipelines (R2)
		Fallback parsing with regular expressions (R15)
	(M25) Integrating Libraries & APIs	Combining MSR libraries (e.g., PyDriller) with LLM APIs (R3)
	(M26) Adopting Local Inference Engines	Using local frameworks like 'ollama' or 'llama.cpp' (R9, R15)

describing how the LLM *"approximates the information it returns and invents mock outputs in accordance with the little information it gets"* (R5). The literature further details this problem through examples of vague or generic outputs, as well as failure to distinguish between nuanced concepts [3, 82, 258, 316].

Mitigation Strategies. To prevent and reduce inaccurate outputs, researchers could employ three strategies:

- **(M1) Grounding via Hybrid Approaches:** A primary strategy is to ground the LLM in verifiable external data rather than relying solely on its internal knowledge. This is achieved by pairing the LLM with deterministic components, such as using **static analysis** to extract code-level facts [185, 369, 393]. This is confirmed by survey respondents who integrate traditional MSR libraries like *"PyDriller or simple GraphQL queries"* (R3) into their LLM pipelines. Other hybrid techniques include structuring information into a **Knowledge Graph (KG)** that the LLM can query [3], or adopting **Retrieval-Augmented Generation (RAG)** to supply relevant context from long documents [14, 185].
- **(M2) Structured & Multi-Step Prompting:** Prompts are engineered to enforce logical consistency. This includes using **Chain-of-Thought (CoT)** prompts that require the model to outline its reasoning steps [14, 71, 369], a practice echoed by researchers who structure their workflows to first ask the LLM to *"explain the classes involved and their interaction"* before delivering a final result (R5). Practitioners also employ **multi-step prompting**, described by one as splitting *"the prompts into multiple prompts... to build... knowledge"* (R5) incrementally. Furthermore, prompts are enriched with explicit context like KG schemas [3], code context [309], detailed extraction rules [393], or pre-defined taxonomies [82, 180], with the goal of providing *"detailed descriptions to generate most consistent (at least correct) results"* (R11). These strategies are also presented in the Approaches (**A5, A10**).

- **(M3) Output Stabilization through Aggregation:** To improve robustness and filter out random errors, many studies employ an aggregation strategy based on multiple executions. Specifically, this involves executing the same prompt on the same model several times and then aggregating the outputs—often through a **majority vote** on the outcomes [309, 393], or by **averaging performance scores across the runs** [71, 369], to determine the most reliable final result. This practice, which focuses on stabilizing the output of a single model configuration rather than combining different models, was corroborated by one of the survey respondents, who noted to *"run experiments multiple times and use a vote criterion to determine the accuracy"* (R18). These strategies are also presented in the Approaches (**A5,10**).

T2: Prompt Engineering & Sensitivity. Crafting effective and reproducible prompts remains a significant hurdle, a problem exacerbated by a wider lack of awareness of methodological guidelines and standards in the field. This threat was noted in eight studies [54, 71, 180, 193, 309, 316, 392, 393] and was a clear theme in the survey. The difficulty is not merely technical but systemic: the rapid review found that only two studies cited external guidelines for their process [3, 306]. This is reflected in the survey, where 11 of the 22 respondents do not follow any established practices, with some stating they are *"Not aware of such standard guidelines"* (R16).

This absence of a shared foundation makes prompt engineering a persistent operational bottleneck, encountered 'Often' or 'Always' by over a quarter of the respondents (six out of 22). Researchers describe the process as *"a long process of trial and error"* (R3), with one identifying their most significant threat as the *"Difficulty in designing the prompt as there are no clear guidelines on how to create them and which are more effective depending on the context"* (R1). This aligns with findings from the literature where LLM performance was found to be highly sensitive

to subtle changes in phrasing and structure, leading to instability or performance degradation [54, 71, 393].

Mitigation Strategies. To mitigate prompt design threats, researchers are combining experimentation with the ad-hoc adoption of emerging best practices and could employ three strategies:

- **(M4) Systematic Iteration & Oracle-Guided Refinement:** The most common mitigation is to embrace what respondents describe as a *"long process of trial and error"* (R3) by turning it into a systematic, iterative process. Researchers conduct pilot tests on a small subset of data to refine prompts over several cycles [82, 316]. This process follows a clear workflow of *"Initial prompt drafting; Validation...; Prompt Refinement"* (R3) within an iterative loop where researchers *"... and redefine prompt (go back... repeat)"* (R13). This refinement is not random; it is often guided by objective feedback from a high-quality **manual oracle** (a ground-truth dataset) or an existing dataset to make evidence-based decisions about which prompt variations perform best before large-scale deployment [82, 185, 316]. This strategy is also presented in the Approach (**A7**). However, performing this refinement using a single LLM may introduce model-specific biases, leading to prompts that are inadvertently optimized for that model's behavior. To mitigate this risk, researchers are increasingly testing prompts across **multiple LLMs** during Stage 4, as suggested in the approach (**A12**), to ensure generalizability and reduce overfitting to a particular LLM.
- **(M5) Adoption of De-Facto & Emerging Standards:** In the absence of formal standards, researchers seek guidance from available sources. This includes adopting modular and structured frameworks proposed in the literature, such as deconstructing prompts into standardized components (e.g., *Persona*, *Task Instruction*) for systematic evaluation [82, 193]. Practitioners also turn to de-facto standards from model providers like *"OpenAI and Google"* [3, 306] or adopt nascent academic guidelines like the *"PRIMES frame-*

work" (R6). In parallel, several communities are creating prompting guides to help researchers and practitioners[7]. These practices align with approaches (**A1,A5**), which cover the selection of prompting strategies.

- **(M6) Meta-Prompting & Automated Assistance:** A complementary strategy, identified primarily through the survey, is the use of **meta-prompting**, where an LLM is employed not to perform the target task, but to help refine the prompt itself. Respondents describe this as a process to *"refine the promt with an LLM"* (R1) or what one explicitly calls *"meta prompting when generating prompts"* (R3). This technique turns the model into a creative assistant that can suggest rephrasings or structural improvements, a workflow some term *"automatic prompt engineering"* (R7). Another researcher described their method as *"asking ChatGPT multiple times for help until coming up with a solution"* (R4), effectively leveraging the LLM to accelerate the prompt engineering cycle. While this was a clear practice among several survey respondents (R1, R3, R4, R7, R19), it is not yet a widely documented mitigation strategy in the formal literature I reviewed, suggesting it is an emerging, practitioner-driven technique. This is also highlighted in the approach (**A7**).

T3: Contextual Understanding & Domain Specificity. LLMs often struggle with the deep contextual understanding required for specialized SE tasks, a finding evident in literature focused on detecting context-heavy code smells [316], interpreting SE-specific metaphors [54], or understanding complex panic conditions in code [393]. This threat is closely linked to the technical limitations of current models, a point emphasized by the survey participants. Over a third of respondents (8 out of 22) encounter technical limitations like context windows 'Often' or 'Always'. An additional five face them 'Sometimes', while seven report

[7] An example of such resources is the Prompting Guide https://www.promptingguide.ai/prompts

them as a 'Rare' occurrence, and only two claimed to have 'Never' encountered such issues. One researcher identified this as their primary obstacle, explaining that the context window *"makes it difficult to provide enough context for accurate reasoning, especially when the model needs to understand dependencies across files or track code evolution"* (R17). To mitigate this, practitioners report adopting strategies such as *"chunking the input intelligently or summarizing or filtering code segments before sending them to the model"* (R17).

Mitigation Strategies. To overcome these limitations, researchers focus on augmenting the LLM with domain-specific knowledge and employing sophisticated techniques to manage the finite context window. Researchers could employ three strategies:

- **(M7) Injecting Domain Knowledge:** To fix this problem, researchers explicitly provide the LLM with domain-specific knowledge. This is often done by grounding the prompt in established **SE-specific taxonomies** and definitions, such as providing a catalog of green architectural tactics [82], formal emotion hierarchies [180], or detailed descriptions of code smells [316] and technical debt categories [309]. A more resource-intensive approach involves **fine-tuning** a model on domain-specific data. This is confirmed as a practice by the survey respondents (R22) and is documented in the literature for tasks like interpreting SE-specific metaphors [54], analyzing sentiment [392], and identifying self-admitted technical debt [309]. This mitigation is related to approaches **(A2, A5)**, which support task definition and prompt design.

- **(M8) Hybrid Program Analysis:** Instead of providing raw data and expecting the LLM to understand its complex relationships, studies use a hybrid approach where traditional program analysis preprocesses the context. Techniques like **static taint analysis**, filtering based on **Abstract Syntax Trees (ASTs)** [185, 369, 393], and **def-use analysis** [258] are used to identify critical data flows or code structures. This practice is corroborated by survey respondents

who use tools like *"PyDriller"* (R3) to *"filter code segments before sending them to the model"* (R17). The output of this analysis is then fed to the LLM, effectively simplifying the reasoning task. This mitigation is related to approach **(A4)**.

- **(M9) Intelligent Context Management:** To work within the LLM's finite context window, a challenge one respondent called their *"most significant"* because it *"makes it difficult to provide enough context for accurate reasoning"* (R17), researchers adopt several intelligent context management strategies. The literature highlights techniques like **automated code slicing** [369], rule-based **truncation** [71, 309], and, for large textual documents, **Retrieval-Augmented Generation (RAG)** [14, 185]. These formal techniques are corroborated in practice, with the same respondent describing their solution as needing to *"chunk the input intelligently or summarize or filter code segments"* (R17) before sending them to the model. These strategies are also reflected in methodological practices (**A3, A4**).

T4: Scalability, Efficiency, & Cost. The practical application of LLMs at scale introduces concerns about computational and financial costs, a theme present in several reviewed papers [82, 185, 369]. These concerns were among the most severe and frequently reported threats in the survey. The majority of researchers, 14 out of 22 (63.6%), encounter issues with both **High Costs** and **Scaling to Large Datasets** 'Often' or 'Always'. For high costs, seven respondents reported the issue occurring 'Sometimes' and only one 'Rarely', with none claiming to have 'Never' faced it. Similarly, for scaling issues, four encountered them 'Sometimes' and three 'Rarely', with just one respondent reporting it was 'Never' a problem. Respondents consistently described this as a primary barrier, with one noting: *"When scaling to large datasets, the inference time and the hardware requirements are a serious problem to take into account"* (R15). Another simply stated that financial cost is *"the most significant factor"* (R20).

Mitigation Strategies. To make large-scale analysis feasible, researchers could employ three strategies:.

- **(M10) Strategic LLM Selection:** Rather than defaulting to the largest and most expensive LLM, a key mitigation is to strategically select one that balances performance and cost. Studies in the literature conduct comparative analyses of different LLMs (e.g., GPT-4 vs. Claude Haiku), explicitly choosing the most cost-effective option that meets the required accuracy threshold [82, 185]. This includes recognizing that state-of-the-art models may only offer marginal performance gains for a significant increase in cost [309]. The survey reveals the practical side of this strategy, where researchers use local, open-source models to avoid API fees (R13) and apply techniques like **model quantization** to run models on less powerful hardware (R15).

- **(M11) Input & Output Token Optimization:** Since cost is tied to token count, a primary focus is on optimization. For input, literature highlights using **program analysis**, such as building an AST, to programmatically **slice code** and provide the LLM with only the minimal necessary context [369]. Complementing this, the survey respondents emphasize controlling the output, a practice also seen in the literature where parameters controlling the output length (e.g., 'max_new_tokens' or 'max_tokens') are explicitly limited [309]. A common cost-saving measure is to strictly limit this parameter to prevent unnecessarily long responses. Respondents confirm this practice, stating its purpose is to *"contain costs"* (R19) or to set the output to the *"bare minimum to have a fixed response"* (R4). As another researcher summarized, this strategy *"has helped to limit the usage (and cost) of the models"* (R3).

- **(M12) Multi-Stage Hybrid Pipelines:** To avoid using expensive LLMs for simple tasks, researchers design multi-stage pipelines that act as a cost-saving filter. A common pattern is to use a fast, low-cost method, such as leveraging vector embeddings for an initial

analysis [14]. The powerful and expensive LLM is then utilized in a second stage to perform a deeper analysis on the much smaller subset of candidate data that has passed the initial filter, thereby optimizing the use of costly resources.

T5: Reproducibility & Output Stability. The reproducibility of results is threatened by a combination of factors, including the inherent stochasticity of LLMs, the "black-box" nature of proprietary models, and the continuous, often undocumented, evolution of the models themselves. The threat of **temporal drift**, where proprietary models can yield different results over time despite being called by the same name, makes long-term replication particularly difficult, a threat explicitly noted in the literature [369]. These issues are frequently reported across the reviewed studies [14, 54, 71, 82, 309, 393].

While this is a well-documented concern, the survey suggests it is perceived as a less frequent operational hurdle compared to other threats. Only three out of 22 respondents (13.6%) encounter reproducibility issues 'Often' or 'Always'. The vast majority experience it less frequently, with twelve reporting it 'Sometimes' and seven 'Rarely'. Notably, no respondent claimed to have 'Never' faced this threat. However, the difficulty in achieving stable results remains a practical concern. To combat output instability, a mitigation strategy mentioned by numerous respondents is to set the model's **temperature** parameter to zero. As one researcher stated, their goal is to set parameters *"such that the responses are deterministic"* (R4). This suggests that practitioners are actively configuring models to mitigate the risks associated with stochasticity, a key component of the broader reproducibility problem.

Mitigation Strategies. To address the threats of non-determinism, improve the stability of their findings, and improve reproducibility, researchers could employ three strategies:

- **(M13) Deterministic Parameter Configuration:** The most widely adopted mitigation, reported in both the literature [14, 71] and

the survey, is to configure the LLM for deterministic output. This is almost universally achieved by setting the **'temperature'** parameter to '0'. Numerous survey respondents (R1, R7, R12, R17, R19) confirmed this practice, with one specifying they use a lower temperature *"when I need more deterministic and consistent outputs"*(R17). This strategy is also highlighted in the approach **A.9**.

- **(M14) Stabilizing Stochastics through Aggregation:** When non-deterministic outputs are used (i.e., 'temperature > 0'), researchers ensure the stability of the final reported result by aggregating multiple runs. Instead of reporting the outcome of a single, random run, studies perform the same query multiple times and then aggregate the results, either by taking a **majority vote** on the outcomes [393] or by **averaging performance scores** across the runs [71, 369]. This approach reduces variance and ensures the overall experimental result is stable and reproducible. The practice is confirmed by a survey respondent who, to handle uncertainty, will *"run experiments multiple time and use a vote criterion to determine the accuracy"* (R18). This approach is also described in **A10**.
- **(M15) Transparent Reporting and Version Pinning:** A crucial step for ensuring reproducibility and mitigating model drift is the meticulous documentation of the experimental setup. This includes transparently reporting all key parameters (temperature, max tokens, etc.) and, critically, pinning the specific model version used (e.g., 'gpt-3.5-turbo-0125' instead of the generic 'gpt-3.5') [14, 71]. This practice acknowledges that models evolve and helps other researchers replicate the study conditions as closely as possible. Furthermore, it is worth noting that replication may be easier with older versions of open-source LLMs than with commercial LLMs, as they may no longer be available. This solution is described in **A.9**.

❗ **T6: Output Formatting & Parsing.** Ensuring that LLMs produce output in a consistent, machine-parsable format has been identified in the

literature as a recurring threat, often requiring manual correction or the development of post-processing scripts [316]. This issue is a practical annoyance confirmed by the survey respondents' descriptions of their automation workflows. Researchers described developing custom scripts to handle inconsistent outputs. For example, one practitioner's pipeline involves a script for *"checking for structural consistency, format issues, and unexpected characters"* (R6). Another detailed a common fallback strategy for parsing: if the output has the requested structure, it is parsed automatically, *"otherwise I use regular expressions to find an answer. If it does not work, the output is considered null"* (R15). These accounts highlight the hidden post-processing effort required in LLM-based mining.

Mitigation Strategies. To mitigate this threat, researchers could employ three strategies:

- **(M16) Prompt-Based Formatting Instructions:** This consists of explicitly defining the desired output structure within the prompt itself. Many studies instruct the LLM to return its response in a structured, machine-readable format like JSON [185, 369]. To further improve compliance, prompts are often augmented with clear reporting instructions or even few-shot examples that demonstrate the exact output format required [82]. This mitigation aligns with methodological practices **(A5)**, which emphasize structured prompting and format specification.
- **(M17) Robust Post-Processing & Fallback Parsing:** Recognizing that LLMs do not always produce outputs that strictly follow the requested format, researchers develop resilient post-processing pipelines. These pipelines often implement a multi-level parsing strategy, a workflow one respondent detailed as parsing *"the output automatically (if it has the requested structure); otherwise i use regular expressions to find an answer"* (R15). Other researchers confirm using *"regular expression for output data extraction"* (R11) as a fallback mechanism [3]. Further techniques include flexible text processing to normalize minor variations (e.g., treating 'Confused'

and 'Confusion' as equivalent) [180]. This mitigation reflects the approach **(A4)**.

- **(M18) Manual Correction or Exclusion:** When automated parsing proves insufficient, researchers adopt one of two strategies. The first is manual correction, where malformed outputs are inspected and adjusted by hand to ensure inclusion in the dataset [316]. The second, more scalable alternative is to exclude non-compliant outputs from the analysis entirely, with the discard rate reported transparently as a limitation of the method [71].

T7: Dataset Noise, Imbalance, & Pre-training Leakage. The quality of the data used to prompt LLMs and the data on which they were pre-trained are critical, as highlighted in the literature through threats like annotation noise [306, 309], dataset imbalance [193, 392], and data leakage from pre-training corpora [71, 185]. These threats were also raised by the survey respondents. In their open-ended responses, practitioners explicitly identified *"Bias due to data leakage"* (R19) and the general need for *"Data quality control"* (R22) as key limitations. This highlights a shared awareness of the 'garbage in, garbage out' principle, where the quality of both the input prompts and the LLM's training data fundamentally constrains the validity of the results.

Mitigation Strategies. To address data-related threats, researchers focus on rigorous curation of ground-truth datasets, prompt-based techniques to steer model behavior, and and could follow three mitigations:

- **(M19) Input Dataset Preparation & Cleaning:** To ensure that the LLM operates on accurate and unbiased data, researchers carefully prepare and clean the input datasets before running the model [82]. This typically involves using multiple human annotators and conducting reconciliation sessions to resolve disagreements and build a consistent gold standard [180]. In addition, several studies perform manual error analysis on LLM outputs to identify and

retrospectively correct mislabeled examples in the original dataset [309, 392].

- **(M20) Bias Mitigation through Prompting & Analysis:** While controlling the bias inherent in pre-trained models is difficult, one novel mitigation strategy from the literature attempts to steer the model at inference time. This involves embedding safeguards directly into the prompt, such as assigning the LLM an "unbiased" persona, to guide it away from generating biased or socially harmful content [193]. Complementing this, one survey respondent suggested a more analytical approach: to *"apply interpretability methods to explain the LLM behavior"* (R22), which can help uncover and understand the sources of bias in a model's decisions. This approach is also highlighted in (**A.6**)
- **(M21) Acknowledgment of Data Leakage:** Data leakage from the LLM's pre-training corpus—particularly in proprietary closed weight models, is often an unresolvable threat. In these cases, researchers cannot verify whether the model has seen the target data during training, which undermines construct validity. As a result, the most appropriate mitigation is to explicitly acknowledge this risk in the study's design and limitations. This includes transparently stating the potential for data leakage and advising readers to interpret findings with appropriate caution [71, 185]. Doing so reinforces the study's integrity and supports informed interpretation, even when the threat cannot be fully eliminated.

T8: Validation Complexity. The literature notes that validating LLM outputs is complex and requires significant manual effort to establish a ground truth, especially for nuanced tasks [193, 258]. This point is one of the most strongly supported findings from the survey, highlighting validation as a critical bottleneck in practice. A significant majority of the respondents (13 out of 22, or 59.1%) report that validation is 'Often' or 'Always' a complex and time-consuming process. An additional five

respondents encounter this 'Sometimes', and four find it to be a 'Rare' issue. Notably, no researcher claimed to have 'Never' faced validation complexity. This complexity was frequently mentioned as the most significant threat. One researcher attributed it to *"the need for a statistically significant oracle, which required manual curation of the dataset"* (R6). Another one added a crucial perspective on the difficulty of automating this step, stating: *"I still could not find reliable sources to demonstrate that validating LLM results with LLMs is scientifically sound"* (R21). These insights underline the current necessity of human-in-the-loop validation in empirical SE studies involving LLMs.

Mitigation Strategies. Researchers mitigate validation complexity with two primary human-centric strategies that trade rigor for scale.

- **(M22) Oracle Construction for Output Evaluation:** To rigorously validate LLM predictions, researchers construct high-reliability oracles using structured, multi-annotator workflows. This "gold standard" approach typically involves: (i) having at least two researchers independently label the same data; (ii) a reconciliation session to resolve disagreements; and (iii) reporting inter-rater reliability metrics (e.g., Cohen's Kappa) to demonstrate the ground truth's consistency [14, 180, 393]. However, survey respondents say that *"complex or time-consuming validation is due to the need for a statistically significant oracle, which requires manual curation of the dataset."* (R6). Even if I have described these approaches (**A7, A11**), future research could develop annotated datasets to improve the use of LLM4MSR.

- **(M23) Scalable Validation via Statistical Sampling:** To address the validation burden in large-scale studies, researchers adopt a strategy based on statistical sampling. Instead of validating every LLM output, a randomly selected, statistically representative subset is manually reviewed to estimate overall model accuracy. This trade-off enables performance assessment while minimizing cost and effort. As noted by the survey respondents, researchers often *"use*

sampled data and statistically validate the outputs" (R10), or more succinctly, *"only sample statistical sample for evaluation"* (R16) when full validation is not feasible.

T9: Lack of Standardized Tooling & Infrastructure. In addition to methodological threats, researchers frequently encounter a lack of standardized tooling tailored for LLM-based MSR studies. The current landscape consists primarily of low-level APIs and general-purpose libraries, requiring significant engineering effort to operationalize studies. This gap emerged clearly in the survey responses, where participants repeatedly cited the need for "tools" (R3), *"workflow tools to automate the process"* (R7), and *"better frameworks/API"* (R4). As a result, researchers are often forced to build bespoke solutions from scratch—an effort described as *"tremendously time consuming"* (R4), particularly when dealing with issues such as GPU memory and library incompatibilities. These infrastructure issues divert time and resources away from core research objectives, introducing significant delays in the research pipeline.

Mitigation Strategies. In response to the absence of mature automation tools, researchers adopt engineering-intensive mitigation strategies. I identified three mitigation strategies that directly correspond to the approach (**A4**), where researchers establish the necessary infrastructure to execute their work.

- **(M24) Developing Custom Automated Pipelines:** The most direct mitigation is to build the missing tools themselves [82]. Researchers report creating custom *"Python scripts, and... an automated pipeline that runs all the phases"* (R2) to manage everything from data extraction and prompt execution to output validation. These pipelines often include custom logic for parsing inconsistent outputs, such as using regular expressions as a fallback when structured formatting fails (R15), thereby codifying the entire experimental workflow for transparency and reproducibility.
- **(M25) Integrating Libraries & APIs:** Instead of building silos, researchers act as integrators, stitching together traditional MSR li-

braries with LLM APIs. For instance, they combine data extraction using established tools like *"PyDriller or simple GraphQL queries"* (R3) with calls to various LLM provider APIs, creating a functional but often brittle end-to-end workflow.

- **(M26) Adopting Local Inference Engines:** To overcome the financial and accessibility barriers of proprietary API-based infrastructure, a growing mitigation is to set up local inference environments. Researchers use frameworks like *"llama.cpp"* (R9) or *"ollama"* (R15) to run open-source models on their own hardware. While this reduces API costs and increases control, it trades one threat for another, requiring significant effort in system configuration and hardware management.

> **RQ_2 — Large Language Model Threats**
>
> *As a result of **RQ_2**, I identified nine threats that researchers encounter when using LLM-based MSR. These threats include hallucinations, prompt sensitivity, validation complexity, limited reproducibility, and high computational cost. To address these threats, I analyzed the literature and survey responses to derive 26 mitigation strategies. These include techniques such as hybrid validation workflows, structured prompt engineering, cost-performance-aware model selection, and intelligent context management. By mapping each threat to a corresponding strategy, the findings provide a structured basis for guiding future researchers in selecting informed methodologies.*

7.4 DISCUSSIONS AND IMPLICATIONS

This study offers several implications for different stakeholders involved in integrating LLMs into MSR studies. In the following, I outline the key implications for the SE research community, with the goal of supporting researchers and practitioners in designing more rigorous studies by addressing possible threats to validity by design. By synthesizing current

practices and threats, the results provide actionable insights that can guide future empirical work involving LLMs and contribute to advancing the methodological robustness and reproducibility of MSR studies.

7.4.1 *Understanding Emerging Approaches in LLM-based MSR*

The findings from **RQ**$_1$ shed light on how SE researchers are currently operationalizing LLMs for MSR, offering a resource for those aiming to assess the gap between high-level methodological guidelines and the actual empirical practices adopted in recent LLM-based MSR studies. These results enrich the existing literature in a way that has not yet been fully captured by prior literature. Hou et al.[166] provide an extensive mapping of LLM applications in SE and identify mining insights from platforms such as GitHub and StackOverflow as an application area. However, their work does not examine how these mining tasks are practically implemented in research workflows. Similarly, Fan et al. [103] and Zheng et al.[399] focus on what LLMs can do, offering taxonomies of tasks and capabilities, but they do not analyze how these models are used in practice. These findings fill this gap by uncovering the decisions and adjustments that researchers make when integrating LLMs into software repository analysis, which are implicit or omitted in publications. The practicalities of prompt formulation, model configuration, and validation are foundational to the success of such studies, yet remain under-discussed.

This absence of methodological transparency has been explicitly noted in recent work. Sallou et al. [297] warn of the threats posed by the uncritical use of LLMs in SE, highlighting the need for greater reflection on assumptions and potential biases. Wagner et al. [355] respond by calling for evaluation guidelines to ensure the rigor of LLM-based empirical studies. Trinkenreich et al. [339] similarly emphasize the growing role of LLMs in SE research, urging the community to take a more systematic approach to experimentation. Lastly, De Martino et al. [80] create a preliminary framework for utilizing LLMs in MSR studies, focusing on creating, refining, and validating prompts that enhance LLM output, particularly in the con-

text of data collection in empirical studies. While these papers articulate broader concerns, this study provides concrete empirical evidence of the kinds of methodological practices that are emerging, as well as the lack of standardized ways to report or reuse them.

The contrast between the operational depth observed in this analysis and the limited methodological framing in prior work suggests that the community is undergoing a quiet methodological shift, one that is not yet reflected in common standards or publication norms. To support the future of LLM-based MSR, and general SE studies, it will be necessary to move beyond task taxonomies and performance benchmarks and begin investing in shared methodological assets such as prompt libraries, configuration templates, validation heuristics, and reporting guidelines that can enable reproducible, transparent, and scalable research described in this approach **A15**.

Implication #1. The findings from $\mathbf{RQ}_1$ reveal a growing methodological maturity in how LLMs are employed for MSR, but this evolution remains under-documented and fragmented. To support robust and reproducible research, I call for community efforts to consolidate prompting practices, configuration guidelines, and validation heuristics into standardized, reusable methodological resources. Lastly, these approaches could be generalized to other areas of software engineering research.

7.4.2 *Revisiting Technical and Methodological Threats To Validity in MSR*

The findings from $\mathbf{RQ}_2$ provide a deeper understanding of the obstacles researchers face when integrating LLMs into MSR. These threats span from the technical to the methodological and are not always acknowledged explicitly in the literature. In this analysis, researchers reported difficulties in selecting suitable LLMs for their tasks, tuning them effectively under resource constraints, and validating their outputs in the absence of ground truth data. These issues were particularly pronounced when dealing with under-documented APIs, hallucinated outputs, and inconsistent results

between runs. Moreover, researchers often had to make trade-offs between automation and manual oversight, balancing efficiency against control and interpretability. Many of these decisions were shaped by the limitations of current tooling and the lack of standardized evaluation practices. To enrich the practical relevance of these findings, one of the key contributions of this work is that I explicitly annotated each threat with its associated mitigation strategies. No study has classified the threats posed by LLMs to MSR studies and defined mitigation strategies to support research. These include, for example, the risk of unstable outputs due to non-deterministic decoding settings, which researchers addressed by enforcing fixed seeds and low-temperature configurations. Similarly, the threat of prompt misalignment, where minor changes in phrasing affect outcomes, was mitigated through iterative pilot testing. By capturing these concrete mitigation strategies, this analysis provides actionable guidance for future researchers. It offers a framework for reasoning about and mitigating the methodological fragility of LLM-based usage in MSR studies. Furthermore, these threats are likely to be relevant to areas other than MSR. Although this empirical study focuses on MSR, threats have been identified that relate to the general use of LLMs outside MSR (e.g., hallucinations), and these threats, mitigations, and approaches have direct implications for the broader use of LLMs.

These findings align with recent discussions on how the software engineering community addresses threats to validity. Lago et al. [207] emphasize that these threats are often underreported or treated as formalities, urging a move beyond standardized checklists to better understand their emergence and management in research. This investigation focuses on how technical and methodological threats occur in studies involving LLMs in MSR. While some identified issues, like construct misalignment, fit traditional validity categories, others, such as unstable decoding behaviors and prompt sensitivity, arise specifically from LLMs. These insights underscore the need for both taxonomical clarity and practical understanding of how researchers navigate uncertainty and adapt their methods. This

work complements Lago et al. [207] agenda by offering a domain-specific perspective on threats to validity in LLM-based research.

Additionally, these threats reinforce and refine observations previously made in the literature. Sallou et al. [297] have warned that the adoption of LLMs in SE is often accompanied by a lack of critical reflection, with risks stemming from opacity, reproducibility issues, and fragile assumptions. Wagner et al. [355] propose evaluation guidelines to improve the quality of empirical studies involving LLMs. These findings empirically validate these concerns: even in well-structured studies, I found limited reporting on validation protocols, replication attempts, or the rationale behind LLM selection. Hou et al. [166], in their SLR, identified multiple research opportunities related to LLM integration, scalability, and data management, but their analysis stops short of detailing the technical and methodological frictions that occur in practice.

These insights suggest that the integration of LLMs into MSR research is not only a matter of LLM capability, but also of process maturity. Methodological fragmentation, lack of reusable assets, and poor tooling support create barriers to scalability, reproducibility, and generalization. This is particularly concerning in a field like MSR, where the quality of insights depends on the consistency and traceability of the mining process. The threats identified in $\mathbf{RQ}_2$ reveal that SE researchers, in the pioneering LLM-based MSR studies (LLMs gained popularity in November 2022), have been left to make ad hoc decisions, constrained by limited computational resources, unstable APIs, and inadequate documentation.

To advance the field, I propose that issues such as "Contextual Understanding & Domain Specificity" (T3), "Validation Complexity" (T8), and "Reproducibility & Output Stability"(T5) should be viewed not only as technical challenges but also as methodological research problems. Addressing these issues requires coordinated efforts from the community. This could involve creating open-source datasets with gold-standard annotations to tackle dataset noise (T7) and validation complexity (T8), developing streamlined validation protocols as a direct response to validation uncertainty, designing model comparison toolkits specifically for SE tasks to

address output inaccuracy (T1) and output instability (T5), and establishing evaluation frameworks that extend beyond traditional metrics like accuracy or F1 score.

Implication #2. The findings from $\mathbf{RQ}_2$ show that the main barriers to progress in LLM-based MSR are not purely technical, but methodological. By mapping threats and mitigation strategies for each identified threat, this study offers actionable knowledge that can inform the design of more stable and reproducible LLM-based pipelines. The MSR and general SE fields now need shared protocols, lightweight validation strategies, and infrastructure for prompt and model management to support robust, scalable, and replicable research practices.

7.4.3 *Implications for the Mining Software Repositories Community*

The rise of LLMs for software repository analysis, and especially for data collection and analysis, represents both an opportunity and a challenge for the MSR community. LLMs promise to enhance traditional mining activities, such as classification and entity extraction, beyond the limits previously achievable with rule-based or shallow learning techniques. However, there is still a lack of a common methodological foundation, resulting in fragmented practices and reproducibility issues that hinder the proper conduct of such studies.

From a community development perspective, this situation calls for a change in the way LLM-based studies are designed, reported, and reviewed. Researchers are no longer mere consumers of pre-trained models, but are actively shaping the behavior of LLMs through prompts, tuning, and post-processing. This change implies that the MSR community, to continue being a reference about open science and reproducibility, must now include in its pipeline-oriented discipline new artifacts as first-order research contributions: prompt design, validation strategies, and reproducibility protocols. In fact, this study shows that many SE researchers are already engaged in this type of methodological work.

Looking ahead, I believe that in the MSR and SE communities, I should actively invest in creating methodological guidelines to support the design and evaluation of LLM-based studies. These guidelines should address essential aspects such as prompt formulation, configuration control, reporting practices, and validation protocols. Similar to past community efforts in dataset curation and shared activities, these resources could foster more robust and transparent research, enabling others to replicate results, compare techniques, and build on previous work in a cumulative manner.

Furthermore, as LLMs become increasingly central to empirical SE, the community will face fundamental challenges related to reproducibility. Unlike traditional mining pipelines, LLM-based workflows often involve stochastic outputs and dependence on rapidly evolving model versions, all of which pose a threat to long-term replicability. Ensuring the reproducibility of research studies will require not only technical solutions (e.g., versioned models, deterministic decoding), but also the extension of more detailed documentation and open sharing of prompts, code, and inference artifacts as described in (**A15**).

More broadly, the approaches in $\mathbf{RQ}_1$ and the challenges in $\mathbf{RQ}_2$ extend not only to the MSR community but also to the SE community. As LLMs are increasingly used to generate explanations, summaries, and retrieve artifacts, SE researchers need to critically examine what constitutes evidence, how it should be evaluated, and what ethical responsibilities arise from automating human-level reasoning tasks.

Implication #3. The integration of LLMs into MSR research opens up new opportunities to go beyond extracting data specific to certain tasks and move toward comprehensive experimental design. To respond to this change, the community must develop a shared infrastructure, define methodological standards, and explicitly address challenges related to reproducibility and validation. This will ensure that LLM-based MSR studies are not only innovative, but also rigorous, transparent, and sustainable over time.

7.4.4 *PRIMES 2.0: An Empirical-Based Methodological Framework for LLM-Based Mining of Software Repositories*

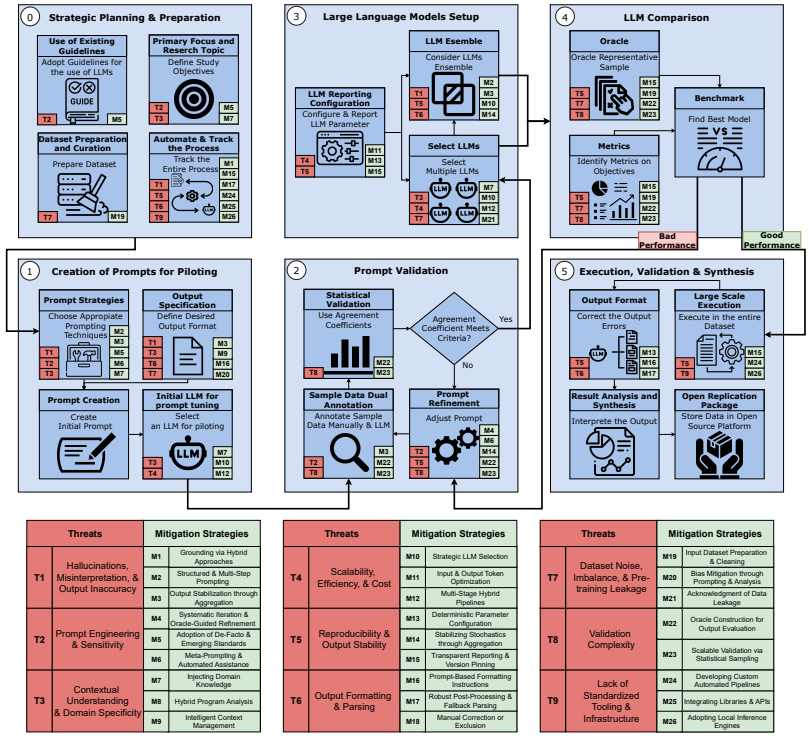


Figure 7.5: PRIMES 2.0: An Empirical-Based Methodological Framework for LLM-Based Mining of Software Repositories.

Building on the results of $\mathbf{RQ}_1$ and $\mathbf{RQ}_2$, I propose an evolved version of the initial framework, PRIMES [80], aimed at supporting the design and execution of LLM-based MSR studies. The original PRIMES framework had some limitations, as it was based on only two previous studies[46, 82] and did not fully address the complexities and threats encountered in the full research pipeline. While the initial version of PRIMES offered a conceptual framework for configuring prompts and models, this study highlighted the need for enhancements in two key areas: methodological improvement

and the integration of strategies for validating and mitigating potential threats.

The new PRIMES 2.0, shown in Figure 7.5, addresses the limitations identified in earlier frameworks by incorporating findings from this mixed-method study. PRIMES 2.0 is organized into six stages, comprising 23 methodological substeps, each mapped to specific threats (T1–T9) and corresponding mitigation strategies (M1–M26), providing prescriptive and adaptive support throughout the lifecycle of LLM-based MSR studies.

PRIMES 2.0 functions as both a design tool and a diagnostic instrument. It empowers researchers to make informed decisions at each stage of the process, from selecting representative repositories and designing reproducible prompts to conducting large-scale evaluations and interpreting results. Unlike traditional linear methodologies, PRIMES 2.0 supports iterative refinement loops, allowing prompts, model configurations, and validation strategies to evolve based on empirical feedback and pilot results. This adaptability is particularly important in MSR contexts, where prompt behaviors can vary significantly depending on the repository context, historical depth, and task-specific semantics. PRIMES 2.0 builds upon and extends previous guidelines tailored for LLM-based software engineering research by offering a structured, staged framework that spans from "strategic planning & preparation" to "execution, validation & synthesis". It is a six-stage process that formalizes best practices for both the design and execution of empirical studies involving LLMs.

Wagner et al. [355] offer a rich set of practical recommendations aimed at improving the transparency, reproducibility, and interpretability of LLM-based SE studies. These include documenting model versioning and decoding parameters, prompt structure and rationale, evaluation criteria, and annotation agreement practices, all of which are also reflected in the studies I analyzed.

These findings corroborate these guidelines and reinforce the emerging consensus on best practices for LLM-based research. The contribution builds on this work by explicitly mapping such practices to recurring methodological threats observed in the field, and by embedding them into

a structured, multi-step protocol. In this sense, PRIMES 2.0 complements previous efforts by operationalizing and organizing these recommendations into concrete stages, mitigation strategies, and validation techniques (e.g., dual annotation, LLM-as-judge loops, inter-model comparisons), thus enabling more fine-grained methodological traceability.

Additionally, PRIMES 2.0 incorporates sustainability aspects from both environmental and methodological perspectives. Expanding on the reflections introduced in the original framework [80], PRIMES 2.0 incorporates sustainability into several stages of the empirical workflow. Specifically, I address the high computational cost and resource consumption associated with large-scale LLM usage by introducing scalable validation techniques and model reuse strategies. For example, rather than evaluating every prompt-model combination exhaustively, PRIMES 2.0 promotes validation via representative sampling and focuses only on configurations that show initial evidence of reliability. The framework encourages reuse of models through local inference setups with open-weight LLMs and supports parameter stabilization to avoid redundant executions and uncontrolled variability across runs. Moreover, PRIMES 2.0 reinforces reproducibility and traceability by requiring researchers to document prompt templates, versioned model configurations, and output logs. These artifacts are designed to be reusable across studies, reducing unnecessary repetition and promoting knowledge sharing. In line with prior concerns about private models and pretraining leakage [297], the framework prioritizes transparent model selection and encourages the use of open-source resources. By embedding these practices directly into the methodological design, PRIMES 2.0 responds to the need for energy-aware, ethically grounded, and methodologically sound LLM-based research in software engineering.

In addition, the work by Sallou et al. [297] raises awareness of construct, internal, and external validity threats, especially for LLMs trained on public datasets. Their guidelines focus on data leakage detection, reproducibility via prompt logs and date stamps, and the use of metamorphic testing. PRIMES 2.0 concretely operationalizes these principles by (i) introducing a dual-annotation process for prompt validation, (ii) recommending meta-

morphic input variants during benchmarking, and (iii) requiring execution metadata and output archiving across all stages. It also reinforces traceability through integrated LLM versioning and automates and tracks the entire process, aligning with the concerns highlighted by Sallou et al. [297]. Although PRIMES 2.0 was developed from empirical studies within the MSR community, its structure and practices are not unique to this domain. While originally designed to support mining-specific tasks, the framework formalizes methodological decisions such as prompt refinement, model comparison, and output validation that are broadly relevant across empirical SE. Several components of PRIMES 2.0 are applicable to other SE contexts where LLMs are utilized for analysis, synthesis, or labeling, including test generation, code summarization, and requirements classification. In these domains, researchers face similar methodological threats, such as prompt sensitivity, output reproducibility, and lack of validation infrastructure, which PRIMES 2.0 directly addresses through its threat-driven design. I envision PRIMES 2.0 as a foundation for future guidelines, protocols, and toolkits that support not only MSR studies but also the broader empirical SE community. Its modular and extensible structure encourages researchers to report both outcomes and the processes that led to them. I invite the community to explore, apply, and extend PRIMES 2.0 to assess its applicability across diverse empirical settings.

Implication #4. PRIMES 2.0 operationalizes the insights from the empirical study into a flexible and threat-aware framework that supports rigorous design, refinement, and validation of LLM-based MSR studies. It provides a methodological foundation consisting of mitigation actions for building reproducible, transparent, and adaptive empirical pipelines.

PART IV

SYNTHESIS, THREATS AND CONCLUSIONS

8

SUMMARY OF THE MAIN FINDINGS

This chapter first answer the four high-level research questions (RQ_A–RQ_D) introduced in Chapter 1. It then elaborates on the findings of the empirical studies presented in Chapters 2,3,4, 5,6, and 7, highlighting how these results open new research gaps and directions for advancing the engineering of sustainable ML-enabled systems. The discussion is structured around four thematic areas: (1) how sustainability is conceptualized within the broader landscape of non-functional requirements in ML-enabled systems, (2) how sustainability can be evaluated in light of trade-offs with other qualities such as fairness, energy efficiency, and maintainability, (3) how automated tools and large language models can support the detection, classification, and mining of sustainability-related practices in ML repositories, and (4) how LLMs themselves can be effectively and sustainably leveraged in repository mining, leading to empirically grounded frameworks that guide their adoption in software engineering research.

8.1 RQ_A: HOW IS SUSTAINABILITY CONCEPTUALIZED FOR RESEARCHERS AND PRACTITIONERS WITHIN THE BROADER LANDSCAPE OF NON-FUNCTIONAL REQUIREMENTS IN ML-ENABLED SYSTEMS?

The first research question examined how sustainability is conceptualized in the context of ML-enabled systems, from the academic research perspective and that of practitioners who build and maintain these systems. The findings revealed that sustainability remains an *emerging and contested quality attribute*, lacking a universally agreed-upon definition within the field of software engineering.

From the **research perspective**, Chapter 2 [J1] presented a systematic literature review on NFRs in ML-enabled systems. The study revealed that sustainability has been predominantly framed through its *environmental dimension*, with an emphasis on energy-efficient AI, carbon footprint reduction, and green software practices. In contrast, the *social and economic dimensions* of sustainability received limited attention. Moreover, most works have articulated sustainability in terms of *high-level principles* (e.g., responsible AI, fairness, efficiency), but they have lacked concrete methods and operational metrics for practitioners. As a result, sustainability was often treated as a *post hoc concern* rather than a requirement that is proactively integrated into the lifecycle of ML-enabled systems. This analysis highlighted a fundamental misalignment between theoretical discourse and engineering needs.

From the **practitioners' perspective**, Chapter 3 [J2] reported the results of a mixed-method study involving interviews and a large-scale survey. Practitioners described sustainability primarily through two complementary lenses: (i) as a *trade-off*, where efficiency may be partially sacrificed to reduce environmental and societal impact, and (ii) as an *ethical responsibility*, aimed at minimizing harm and considering long-term consequences for future generations. Quantitative results confirmed that practitioners valued all three sustainability dimensions—environmental, social, and economic—although the environmental dimension was consistently ranked as the most critical. Beyond conceptualizations, practitioners identified *concrete sustainability practices*, including 19 approaches grouped into five thematic categories. They also reported practical metrics and acknowledged systemic challenges such as lack of awareness, insufficient incentives, absence of dedicated frameworks, and limited educational resources.

Bridging these two perspectives highlights an important research gap (**RGap$_1$**): the absence of a shared conceptual foundation that unifies the *principle-driven academic framing* of sustainability with the *practice-driven operationalization* employed by engineers. While researchers em-

phasize sustainability as a broad socio-technical concern, practitioners demand actionable guidelines, standardized metrics, and tool support.

Answer RQ$_A$

Sustainability in ML-enabled systems is conceptualized inconsistently. Researchers emphasize high-level environmental principles and broad socio-technical framings, whereas practitioners adopt a pragmatic perspective focused on efficiency, maintainability, and ethical responsibility. This misalignment calls for integrated definitions, taxonomies, and frameworks that connect abstract principles with actionable engineering practices and evaluation metrics.

8.2 RQ$_B$: HOW CAN WE EVALUATE SUSTAINABILITY IN ML-ENABLED SYSTEMS, ACCOUNTING FOR TRADE-OFFS BETWEEN DIFFERENT NON-FUNCTIONAL REQUIREMENTS?

Assessing sustainability in ML-based systems requires explicit recognition of its interaction with other NFRs. From a practitioner's perspective (Chapter 3), sustainability is never pursued in isolation, but considered alongside other qualities such as fairness, energy, and cost. Professionals emphasized the importance of metrics, including energy consumption, CO_2 emissions, return on investment, and employee satisfaction, which enable them to capture sustainability in everyday engineering decisions. Their insights highlight how sustainability is perceived both as a measurable outcome and as a trade-off with other system qualities.

From the research perspective (Chapter 4), empirical benchmarks confirmed the multi-dimensional nature of these trade-offs. By applying six bias mitigation algorithms across multiple datasets, the study revealed that improvements in fairness (social sustainability) frequently come at the cost of higher energy consumption and carbon emissions (environmental sustainability), as well as longer training time and reduced accuracy (economic sustainability). While some algorithms demonstrated competitive

trade-offs, the overall results underscore that optimizing one dimension of sustainability often worsens another.

Answer RQ$_B$

Sustainability in ML-enabled systems can only be evaluated through a multi-dimensional lens that explicitly accounts for trade-offs with other NFRs. Practitioners rely on concrete operational metrics, whereas empirical evidence suggests that improvements in fairness, energy, and accuracy often conflict with sustainability goals. This calls for integrated evaluation frameworks that can balance environmental, social, and economic dimensions, rather than optimizing sustainability in isolation.

8.3 RQ$_C$: HOW CAN AUTOMATED TOOLS AND LLMS SUPPORT THE MINING AND IDENTIFICATION OF SUSTAINABILITY PRACTICES IN ML-ENABLED REPOSITORIES?

Automated tools enable the large-scale identification and analysis of ML-enabled repositories, providing the foundation for sustainability-oriented mining studies. Chapter 5 introduced *MARK*, a rule-based classifier designed to distinguish between producer, consumer, and hybrid ML projects. Validated on more than 4,600 repositories, MARK demonstrated high accuracy in identifying producers and significantly reduced the manual effort otherwise required for dataset construction. Although rule-based heuristics struggled with hybrid or unconventional projects, the resulting dataset provided a reliable and reproducible basis for subsequent analyses.

Building on Chapter 5, Chapter 6 applied LLMs to the MARK dataset to investigate the presence of green architectural tactics. LLMs confirmed several sustainability tactics already known in the literature and, more importantly, uncovered additional undocumented tactics by analyzing Python files. LLMs have demonstrated their ability to perform tasks automatically with high performance. At the same time, LLM-based mining has introduced risks of variability, hallucinations, and computational over-

load, which have required mitigation through prompt refinement, expert validation, and structured methodological guidance.

👉 Answer RQ$_C$

Automated tools and LLMs provide complementary support for mining sustainability practices in ML-enabled repositories. Rule-based classification with MARK enables the scalable and reproducible construction of datasets of ML projects, reducing manual effort while ensuring high accuracy in identifying producers. Building on this foundation, LLMs are able to go beyond predefined taxonomies by confirming known green architectural tactics and uncovering previously undocumented ones directly from source code and related artifacts. The combination of these approaches offers both the interpretability and reliability of rule-based methods and the contextual sensitivity of LLMs. However, to be effective, this integration requires robust validation strategies, prompt refinement, and methodological frameworks to mitigate risks of variability, hallucination, and computational overhead.

8.4 RQ$_D$: HOW CAN LLMS BE EFFECTIVELY AND SUSTAINABLY LEVERAGED IN REPOSITORY MINING SCENARIOS, CONSIDERING THE PERSPECTIVE OF RESEARCHERS INTERESTED IN MINING ML REPOSITORIES?

Leveraging LLMs in repository mining offers new opportunities but also raises challenges that require methodological rigor. Chapter 6 showed that while LLMs can successfully identify known and undocumented sustainability tactics, their use is hindered by variability in outputs, hallucinations, and high computational costs. These limitations emphasize that researchers cannot rely on LLMs in an ad hoc fashion but must adopt structured approaches that ensure reproducibility and validity.

To address this need, the study proposed the *PRIMES 2.0* framework in Chapter 7, which provides methodological guidance for integrating LLMs

into repository mining studies. PRIMES 2.0 structures the process into stages that cover prompt creation and refinement, empirical validation, model selection and comparison, and the evaluation of trade-offs among accuracy, cost, and energy. This framework not only improves reproducibility and robustness but also supports a more sustainable use of LLMs by explicitly accounting for their computational footprint.

👉 Answer RQ$_D$

LLMs can be effectively and sustainably leveraged in repository mining when structured methodological frameworks guide their adoption. PRIMES 2.0 demonstrates the need of rigorous prompt design, empirical validation, model comparison, and explicit evaluation of trade-offs for mitigating variability, hallucination, and computational overhead. PRIMES 2.0 enables researchers to move beyond ad hoc use and employ LLMs in a reproducible, reliable, and sustainability-aware manner.

9 THREATS TO VALIDITY

This chapter summarizes the threats to validity that apply across the studies presented in this thesis. I distinguish between (i) *common threats* that cut across all methodological approaches, (ii) *quantitative-specific threats*, and (iii) *qualitative-specific threats*. This organization enables the synthesis of recurring issues while still accounting for threats tied to particular study designs. The categories discussed below map onto the four classic dimensions of construct, internal, external, and conclusion validity [368], but are organized thematically to better reflect the diversity of methods used in this thesis.

9.1 COMMON THREATS TO VALIDITY

Across the studies, several recurring threats emerged. Although they manifested differently depending on whether the study was a literature review, a survey, a mining experiment, or a tool-based analysis, they share common problems and mitigation strategies.

SCOPE DELIMITATION AND SELECTION BIAS. *Problem.* Design decisions regarding where to search, what to include, and whom to sample risked omitting relevant evidence and skewing coverage. This was most evident in literature reviews, where NFRs were often referred to by specific terms (e.g., *fairness*, *robustness*) rather than the generic label NFRs, risking under-coverage (Chapter 2). Similarly, in surveys and interviews, participant selection could bias the range of perspectives.

Mitigation. To mitigate this risk, I employed a hybrid systematic review combining Kitchenham's protocol [199] with Wohlin's venue-based guidelines [367] and snowballing [365], consulting multiple databases (ACM

DL, Scopus, IEEE Xplore). In other studies, I complemented in Chapter 3 the small-sample interviews with a large-scale survey (203 participants) following practices from empirical SE [210, 231, 268, 301], pre-screened respondents for heterogeneity, and inserted attention-check questions to filter careless responses [15]. In mining-based analyses (Chapter 5 and Chapter 6), I selected well-established datasets such as NICHE [364] and Gonzalez et al. [136], applying preprocessing to exclude toy projects and, in Chapter 6, selecting only active projects.

RESEARCHER SUBJECTIVITY IN MANUAL STEPS. *Problem.* Manual activities such as study screening, quality assessment, qualitative coding, oracle construction, and error analysis carry risks of subjective interpretation and inconsistencies.

Mitigation. To minimize these risks, I defined inclusion and quality assessment criteria a priori and validated them on held-out samples, refining them through consensus sessions (Chapter 2 and Chapter 7). In other cases, multiple authors coded data independently, measured inter-rater reliability using Cohen's Kappa [70], and resolved disagreements in meetings (Chapters 3, 5, 7). For qualitative data, I relied on Krippendorff's content analysis framework [202] and grounded theory practices to guide coding and synthesis. Author expertise also played an important role in mitigating subjectivity: for example, the first and second authors had complementary experience in NFRs, ML, and SE4AI (Chapters 2, 5, and 6), which reduced the risk of misinterpretation. Processes and artifacts were fully logged in appendices to ensure transparency and replicability.

MEASUREMENT IMPRECISION AND INSTRUMENTATION EFFECTS. *Problem.* Instruments used in surveys, experiments, or LLM-based analyses may introduce inaccuracies, whether through response bias, background variability in hardware/software, or stochastic model behavior.

Mitigation. In survey design, I combined open definitions, Likert-scale items, and ranking questions to reduce acquiescence and social desirability bias [137], following Dillman's tailored design method [90] and

Ciolkowski et al. [68] recommendations for SE surveys . In experiments, I stabilized energy/performance measurements with warm-up runs, controlled background processes, and pauses between executions, while acknowledging CodeCarbon's known underestimation [375] (Chapter 4), in line with established protocols in green software engineering [75, 114, 291]. In LLM-based analyses, I refined prompts iteratively, tested outputs for determinism across runs, and validated results against manually constructed oracles (Chapter 6).

PROXY CONSTRUCTS AND CONSTRUCT BOUNDARIES. *Problem.* Some constructs could only be captured through proxies that partially represent the intended phenomena, such as stars/forks for project popularity, subsets of sustainability metrics for the broader concept of sustainability, or heuristic rules for project classification.

Mitigation. To address this, I adopted proxies widely recognized in prior work [29, 57], applied transparent heuristics with known trade-offs (precision vs. recall) in project role classification (Chapter 5), and explicitly bounded claims to the defined constructs. For example, I selected nine sustainability metrics across social, environmental, and economic dimensions [63, 65, 114, 375] in Chapter 4, balancing coverage with interpretability. In Chapter 2, I restricted synthesis to challenges explicitly tied to NFRs, excluding unrelated ML concerns.

REPRODUCIBILITY AND TRACEABILITY. *Problem.* Multi-step pipelines are sensitive to seeds, dependency versions, and API evolution, potentially hindering replication.

Mitigation. To ensure reproducibility, I documented processes in detail, released appendices and replication packages (Chapters 4, 5, 6), and reported configurations and parameters comprehensively.

GENERALIZABILITY AND TRANSFERABILITY. *Problem.* Findings may be bound to particular datasets, participants, languages, or algorithms, limiting external validity.

Mitigation. I explicitly limited claims to the studied contexts, scaled up small-sample interviews with large surveys (Chapter 3), employed diverse datasets and models while clearly stating the attribute scope (Chapter 4), and justified the Python focus due to its widespread adoption in ML (Chapters 5, 6). Replications in other contexts remain a future direction.

USE OF METHODOLOGICAL GUIDELINES AND STANDARDS. *Problem.* Without systematic procedures, studies risk unclear constructs, inappropriate design, and insufficient rigor.

Mitigation. To avoid this, I consistently adopted recognized standards: Kitchenham [199] and Wohlin [367] for literature reviews; Dillman [90], Ciolkowski [68], and Kitchenham [198] for surveys; Krippendorff [202] and grounded theory for qualitative coding; Wohlin's guidelines on controlled experimentation [368]; and the PRIMES framework for LLM-based mining [79]. These standards were adapted to each study's goals and documented to ensure transparency.

9.2 QUANTITATIVE-SPECIFIC THREATS

Quantitative studies introduced additional risks related to measurement, statistical inference, and automation.

Metric selection trade-offs. Operationalizing sustainability with nine metrics (Chapter 4) balanced interpretability with feasibility but excluded other indicators.

Statistical conclusion validity. Violations of normality, small samples, and multiple comparisons posed risks of Type I/II errors. In Chapter 4 and 5, these were mitigated with Shapiro–Wilk tests [308], nonparametric tests (Friedman [159, 254], Kruskal–Wallis [72], Mann–Whitney [243]), post-hoc Nemenyi analysis, and Holm–Bonferroni corrections [160]. Multiple repetitions stabilized outcomes.

Heuristic edge cases. MARK's heuristic classification, while efficient for large-scale mining, struggled with hybrid or non-standard projects, maintaining precision at the expense of recall (Chapter 5).

Oracle construction. Manually built oracles provided reliable ground truth but could not capture all project variations; this was mitigated with statistically significant samples and complementary qualitative inspection.

Parameter dependence. Default hyperparameters and toolkit settings ensured comparability with prior work [64, 65] but may affect outcomes under different configurations (Chapter 4 and Chapter 6).

LLM non-determinism. Stochastic outputs in tactic extraction introduced variability (Chapter 6); prompt refinement, pilot tests, and oracle comparisons reduced but did not eliminate this threat. Importantly, I also accounted for the risk of *hallucinations* in LLM outputs, ensuring that only tactics with clear evidence in primary studies were retained.

9.3 QUALITATIVE-SPECIFIC THREATS

Qualitative studies presented risks in construct framing, participant sampling, and subjective interpretation.

Conceptual clarity. Ambiguous definitions could bias responses; to mitigate this, I grounded sustainability in Brundtland's definition [35] and McGuire et al.'s sustainability dimensions for software processes [242] (Chapter 3).

Social desirability bias. Respondents might rate all sustainability dimensions as equally important; combining open definitions, Likert items, and ranking questions reduced acquiescence bias [137].

Small samples. Limited interview participants (8/35) risked restricted perspectives; this was countered with a large-scale survey (203 participants) to increase breadth (Chapter 3).

Subjectivity in coding. Interpretation of qualitative data is inherently subjective; to address this, I applied systematic content analysis [202], had multiple coders review responses, measured Cohen's Kappa agreement, and reconciled differences in consensus meetings (Chapters 3, 7).

Interpretation of open responses. Open-ended survey data risked superficial or inconsistent coding; triangulation with rapid review results and collaborative category development mitigated this (Chapter 7).

Sampling strategies. Purposive and convenience sampling limited statistical representativeness but ensured domain expertise, aligning with recommendations for SE research [15]. *AI-generated responses.* A specific threat in survey-based research is contamination by AI-generated text; in Chapter 3, I explicitly mitigated this by applying the ZeroGPT tool to detect and filter synthetic responses.

10

CONCLUSION

The research presented in this dissertation has focused on advancing the sustainability of ML-enabled systems by combining empirical software engineering methods, automated classification techniques, and large language models (LLMs). Addressing critical challenges in the conceptualization, evaluation, and automation of sustainability as a non-functional requirement (NFR), this thesis aims to bridge the gap between academic perspectives and practitioners' needs, demonstrating that sustainability cannot be treated as an afterthought but rather as an integral quality attribute in ML engineering.

The work began with an exploration of how sustainability is conceptualized across research and practice. The systematic literature review (Chapter 2) revealed that researchers predominantly frame sustainability in terms of environmental concerns—energy efficiency and carbon footprint—while leaving social and economic dimensions less explored. In contrast, the practitioner survey (Chapter 3) highlighted a more pragmatic view, where sustainability is tied to efficiency in training and inference, maintainability of ML pipelines, and ethical responsibility. This divergence revealed the absence of a shared conceptual foundation, pointing to the need for integrated definitions and taxonomies that align high-level principles with actionable practices.

The thesis then turned to the evaluation of sustainability in practice. Insights from practitioners showed the importance of operational metrics, including energy, CO_2, ROI, and employee satisfaction, while empirical benchmarks (Chapter 4) quantified the trade-offs between sustainability and other NFRs. The results demonstrated that fairness, accuracy, and efficiency improvements often increase energy usage or reduce maintainability, confirming that sustainability must be assessed through a multidimensional lens where competing qualities are explicitly balanced.

To address the lack of automated support, Chapter 5 introduced *MARK*, a rule-based classifier that categorizes ML repositories into producers, consumers, and hybrids. Validated on over 4,600 projects, MARK reduced the manual effort required for dataset construction and improved construct validity in repository mining. Building on this dataset, Chapter 6 explored the use of LLMs to identify sustainability-related practices, with a focus on green architectural tactics. LLMs confirmed existing tactics and uncovered undocumented ones, revealing their potential for discovering implicit sustainability signals across artifacts such as code, documentation, and issues. However, they also introduced risks of variability, hallucinations, and computational overhead, highlighting the necessity for structured methodological guidance.

To this end, Chapter 7 proposed *PRIMES 2.0*, a methodological framework that systematizes the use of LLMs in repository mining. By structuring prompt design, empirical validation, model comparison, and trade-off evaluation, PRIMES 2.0 ensures reproducibility, rigor, and sustainability awareness. This framework marks a step forward in moving from ad hoc experimentation with LLMs to systematic, reliable, and responsible adoption in empirical software engineering research.

This dissertation makes several key contributions. First, it provides a comprehensive analysis of sustainability as a non-functional requirement in ML-enabled systems, comparing how researchers and practitioners conceptualize it. Second, it offers empirical evidence of the trade-offs between sustainability and other NFRs, promoting sustainability as a multidimensional and negotiated quality attribute. Third, it introduces *MARK*, a rule-based tool that automates the classification of ML repositories, enabling the scalable and reproducible construction of datasets. Fourth, it demonstrates the potential of LLMs for mining sustainability practices, showing their ability to both confirm existing tactics and uncover new ones beyond predefined taxonomies. Finally, it proposes *PRIMES 2.0*, a methodological framework for LLM-based repository mining that ensures rigor, reproducibility, and awareness of sustainability in empirical studies.

While the findings are significant, several limitations must be acknowledged. The practitioner survey is limited by its sample size and potential for bias in self-reported perceptions. The empirical benchmarks, although comprehensive, cannot capture all ML use cases and algorithms. Rule-based approaches, such as MARK, are limited by the heuristics they encode, which restricts their adaptability to unconventional projects. LLMs, while powerful, remain prone to hallucinations, variability, and high computational costs. These challenges were mitigated through triangulation, validation, and methodological rigor, but they highlight areas that future research should address.

The results of this thesis open several promising avenues for further research. One avenue concerns the development of shared conceptual foundations and taxonomies of sustainability that encompass environmental, social, and economic dimensions, aligning with engineering practice. Another priority is the design of multidimensional evaluation frameworks that integrate quantitative metrics, qualitative evidence, and decision-support mechanisms to capture sustainability trade-offs. A third direction is the combination of rule-based classifiers with LLMs to leverage the scalability of heuristics with the contextual reasoning of neural models. Future work should also investigate the environmental and economic footprint of LLM-based mining itself, assessing energy, cost, and reproducibility trade-offs. Finally, establishing community benchmarks, datasets, and reporting standards for sustainability-focused empirical studies remains essential for ensuring comparability and rigor.

Sustainability is an increasingly central but complex issue in ML engineering. This thesis clarifies how it is conceptualized, demonstrates how it can be evaluated in light of trade-offs, and develops automated, LLM-based methods to support its identification in practice. By combining empirical evidence and methodological innovation, this work bridges the gap between theory and practice, providing both theoretical clarity and practical tools. Ultimately, it contributes to the design of ML-based systems that are not only effective and fair but also sustainable, reliable, and socially responsible.

LIST OF PUBLICATIONS

The complete list of publications is reported below. Titles in **bold** are discussed in this dissertation.

INTERNATIONAL JOURNAL PAPERS

J01 - **De Martino V.** and Palomba F. (2025). **Classification and Challenges of Non-Functional Requirements in ML-enabled systems: A Systematic Literature Review.** Information and Software Technology (IST), 107678.

J02 - **De Martino V.**, Voria G., Troiano C., Catolino G. and Palomba F. (2025). **Examining the Impact of Bias Mitigation Algorithms on the Sustainability of ML-enabled Systems: A Benchmark Study.** Journal of Systems and Software (JSS), 112458.

J03 - **De Martino V.**, Recupito G., Giordano G., Di Nucci D., Ferrucci F., and Palomba F. (2025). **Into the ML-universe: An Improved Classification and Characterization of Machine-Learning Projects.** Journal of Systems and Software (JSS), 112471.

J04 - **De Martino V.**, Lambiase S., Pecorelli F., Van Den Heuvel W. J., Ferrucci F, and Palomba F. (2025). **Sustainability of Machine Learning-Enabled Systems: The Machine Learning Practitioner's Perspective.** Transaction on Software Engineering and Methodology (TOSEM).

J05 - **De Martino V.**, Martınez-Fernández, S. and Palomba F. (2025). **Sustainable Practices in ML-enabled Systems: An LLM-based Repository Mining Study.** Under Major Revision at Empirical Software Engineering (EMSE).

J06 - **De Martino V.**, Castaño J, Palomba F, Franch X. and Martınez-Fernández S. (2025). **A Methodological Framework for LLM-Based Mining of Software Repositories.** Under Revision at Transaction on Software Engineering and Methodology (TOSEM).

J07 - Pentangelo V., Di Dario D, **De Martino V.**, Dello Buono M., and Lambiase S. (2025). Towards AI-Powered Automatic 3D Scene Generation for the Metaverse: A Comparative Analysis of Manual and Photogrammetry Techniques. Under Review at Cluster Computing.

INTERNATIONAL CONFERENCE PAPERS

C01 - Della Porta A., **De Martino V.**, Recupito G., Iemmino C., Catolino G., Di Nucci D., and Palomba F. (2024). Using large language models to support software engineering documentation in waterfall life cycles: Are we there yet? CEUR Workshop Proceedings, vol. 3762, pp. 452–457.

C02 - Novielli N., Oliveto R., Palomba F., Calefato F., Colavito G., **De Martino V.**, Della Porta A., Giordano G., Guglielmi E., Lanubile F., Quaranta L., G. Recupito G., Scalabrino S. , Spina A., and Vitale A. (2024) QualAI: Continuous Quality Improvement of AI-based Systems. RCIS Workshops.

C03 - Novielli N., Oliveto R., Palomba F., Calefato F., Colavito G., **De Martino V.**, Della Porta A., Giordano G., Guglielmi E., Lanubile F., Quaranta L., G. Recupito G., Scalabrino S. , Spina A., and Vitale A. (2024). Continuous Quality Improvement of AI-based Systems: the QualAI Project. In Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (pp. 603-607).

C04 - Pentangelo V., Di Dario D., **De Martino V.**, Dello Buono M., and Lambiase S. (2024). Accelerating 3D Scene Development for the Metaverse: Lessons from Photogrammetry and Manual Modeling.

2nd International Conference on Intelligent Metaverse Technologies & Applications (iMETA), IEEE, pp. 190–197.

C05 - **De Martino V.**, Martínez-Fernández S., and Palomba F. (2025). Do developers adopt green architectural tactics for ML-enabled systems? A mining software repository study. IEEE/ACM 47th International Conference on Software Engineering in Society (ICSE-SEIS), pp. 135-139.

C06 - **De Martino V.**, Castaño J., Palomba F., Franch X., and Martínez-Fernández S. (2025). A Framework for Using LLMs for Repository Mining Studies in Empirical Software Engineering. IEEE/ACM 2nd International Workshop on Methodological Issues with Empirical Studies in Software Engineering (WSESE),pp. 6-11.

C07 - Recupito G., **De Martino V.**, Di Nucci D., and Palomba F. (2025). A First Look at the Lifecycle of DL-Specific Self-Admitted Technical Debt. IEEE 2nd Workshop on Software Quality Assurance for Artificial Intelligence (SQA4AI).

C08 - Palomba F., Voria G., Parziale A., Pentangelo V., Della Porta A., **De Martino V.**, Recupito G. and Giordano G. (2025). Teaching Software Quality Assurance for Artificial Intelligence: An Experience Report. 51st Euromicro Conference Series on Software Engineering and Advanced Applications (SEAA).

C09 - **De Martino V.**, Zadenoori MA., Franch X., and Ferrari A. (2026). Green Prompting Engineering: Investigating the Energy Impact of Prompt Design in Software Engineering. Submitted to IEEE/ACM 48th International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)

C10 - **De Martino V.**, Franch X., and Martínez-Fernández S (2026). Green Prompt Engineering: Guidelines and Research Agenda. Submitted to IEEE/ACM 48th International Conference on Software Engineering in Society (ICSE-SEIS).

C11 - Zadenoori MA., **De Martino V.**, Dąbrowski J, Franch X., and Ferrari A. (2026) Does Model Size Matter? A Comparison of Small and Large Language Models for Requirements Classification. Submitted to the 32nd International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ).

BIBLIOGRAPHY

[1] https://www.gartner.com/en/newsroom/press-releases/2024-05-16-gartner-identifies-the-top-five-strategic-technology-trends-in-software-engineering-for-2024. [Accessed 27-02-2025].

[2] Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher. "The impact of popularity bias on fairness and calibration in recommendation." In: *arXiv preprint arXiv:1910.05755* (2019).

[3] Samuel Abedu, Laurine Menneron, SayedHassan Khatoonabadi, and Emad Shihab. "RepoChat: An LLM-Powered Chatbot for GitHub Repository Question-Answering." In: *2025 IEEE/ACM 22nd International Conference on Mining Software Repositories (MSR)*. IEEE. 2025, pp. 255–259.

[4] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudík, John Langford, and Hanna Wallach. "A reductions approach to fair classification." In: *International conference on machine learning*. PMLR. 2018, pp. 60–69.

[5] Aniya Aggarwal, Pranay Lohia, Seema Nagar, Kuntal Dey, and Diptikalyan Saha. "Black Box Fairness Testing of Machine Learning Models." In: *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2019. Association for Computing Machinery, 2019, 625–635. ISBN: 9781450355728. DOI: 10.1145/3338906.3338937. URL: https://doi.org/10.1145/3338906.3338937.

[6] Roozbeh Aghili, Heng Li, and Foutse Khomh. "Studying the characteristics of AIOps projects on GitHub." In: *Empirical Software Engineering* 28.6 (2023), p. 143.

[7] Khlood Ahmad, Mohamed Abdelrazek, Chetan Arora, Muneera Bano, and John Grundy. “Requirements engineering for artificial intelligence systems: A systematic mapping study.” In: *Information and Software Technology* 158 (2023), p. 107176.

[8] Md Nafee Al Islam, Yihong Ma, Pedro Alarcon, Nitesh Chawla, and Jane Cleland-Huang. “RESAM: Requirements elicitation and specification for deep-learning anomaly models with applications to UAV flight controllers.” In: *2022 IEEE 30th International Requirements Engineering Conference (RE)*. IEEE. 2022, pp. 153–165.

[9] Afnan A Al-Subaihin, Federica Sarro, Sue Black, Licia Capra, and Mark Harman. “App store effects on software engineering practices.” In: *IEEE Transactions on Software Engineering* 47.2 (2019), pp. 300–319.

[10] Adam Alami, Mansooreh Zahedi, and Neil Ernst. “Are you a real software engineer? best practices in online recruitment for software engineering studies.” In: *Proceedings of the 1st IEEE/ACM International Workshop on Methodological Issues with Empirical Studies in Software Engineering*. 2024, pp. 52–57.

[11] Mohamed Abdullahi Ali, Ng Keng Yap, Abdul Azim Abd Ghani, Hazura Zulzalil, Novia Indriaty Admodisastro, and Amin Arab Najafabadi. “A systematic mapping of quality models for AI systems, software and components.” In: *Applied Sciences* 12.17 (2022), p. 8700.

[12] Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. “Software engineering for machine learning: A case study.” In: *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE. 2019, pp. 291–300.

[13] Amin Aminifar, Matin Shokri, Fazle Rabbi, Violet Ka I. Pun, and Yngve Lamo. “Extremely Randomized Trees With Privacy Preser-

vation for Distributed Structured Health Data." In: *IEEE Access* 10 (2022), pp. 6010–6027. DOI: 10.1109/ACCESS.2022.3141709.

[14] Dharun Anandayuvaraj, Matthew Campbell, Arav Tewari, and James C Davis. "FAIL: Analyzing Software Failures from the News Using LLMs." In: *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*. 2024, pp. 506–518.

[15] Sebastian Baltes and Paul Ralph. "Sampling in software engineering research: A critical review and guidelines." In: *Empirical Software Engineering* 27.4 (2022), p. 94.

[16] Teodora Baluta, Zheng Leong Chua, Kuldeep S Meel, and Prateek Saxena. "Scalable quantitative verification for deep neural networks." In: *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE. 2021, pp. 312–323.

[17] Mercy Bamiduro, Iffat Fatima, Patricia Lago, and Sophie Vos. "Challenges in Incorporating Sustainability Practices in the Software Lifecycle." In: *Software Business. ICSOB 2024*. Springer Nature, 2024.

[18] Abdul Ali Bangash, Hareem Sahar, Shaiful Chowdhury, Alexander William Wong, Abram Hindle, and Karim Ali. "What do developers know about machine learning: a study of ml discussions on stackoverflow." In: *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE. 2019, pp. 260–264.

[19] Cauã Ferreira Barros, Bruna Borges Azevedo, Valdemar Vicente Graciano Neto, Mohamad Kassab, Marcos Kalinowski, Hugo Alexandre D Do Nascimento, and Michelle CGSP Bandeira. "Large Language Model for Qualitative Research: A Systematic Mapping Study." In: *2025 IEEE/ACM International Workshop on Methodological Issues with Empirical Studies in Software Engineering (WSESE)*. IEEE. 2025, pp. 48–55.

[20] Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya Nori, and Antonio Criminisi. "Measuring Neural Net Robustness with Constraints." In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg,

I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper/2016/file/980ecd059122ce2e50136bda65c25e07-Paper.pdf.

[21] Barry Becker and Ronny Kohavi. *Adult*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5XW20. 1996.

[22] Christoph Becker, Ruzanna Chitchyan, Leticia Duboc, Steve Easterbrook, Birgit Penzenstadler, Norbert Seyff, and Colin C Venters. "Sustainability design and software: The karlskrona manifesto." In: *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. Vol. 2. IEEE. 2015, pp. 467–476.

[23] Rachel KE Bellamy, Kuntal Dey, Michael Hind, Samuel C Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, Aleksandra Mojsilović, et al. "AI Fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias." In: *IBM Journal of Research and Development* 63.4/5 (2019), pp. 4–1.

[24] João Helis Bernardo, Daniel Alencar Da Costa, Sérgio Queiroz de Medeiros, and Uirá Kulesza. "How do machine learning projects use continuous integration practices? an empirical study on GitHub actions." In: *Proceedings of the 21st International Conference on Mining Software Repositories*. 2024, pp. 665–676.

[25] Manal Binkhonain and Liping Zhao. "A review of machine learning algorithms for identification and classification of non-functional requirements." In: *Expert Systems with Applications: X* 1 (2019), p. 100001.

[26] Sumon Biswas, Md Johirul Islam, Yijia Huang, and Hridesh Rajan. "Boa Meets Python: A Boa Dataset of Data Science Software in Python Language." In: *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. 2019, pp. 577–581. DOI: 10.1109/MSR.2019.00086.

[27] Sumon Biswas and Hridesh Rajan. "Do the Machine Learning Models on a Crowd Sourced Platform Exhibit Bias? An Empirical Study on Model Fairness." In: *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2020. Association for Computing Machinery, 2020, 642–653. ISBN: 9781450370431. DOI: 10.1145/3368089.3409704. URL: https://doi.org/10.1145/3368089.3409704.

[28] Sumon Biswas and Hridesh Rajan. "Fair Preprocessing: Towards Understanding Compositional Fairness of Data Transformers in Machine Learning Pipeline." In: *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2021. Association for Computing Machinery, 2021, 981–993. ISBN: 9781450385626. DOI: 10.1145/3468264.3468536. URL: https://doi.org/10.1145/3468264.3468536.

[29] Hudson Borges, Andre Hora, and Marco Tulio Valente. "Understanding the factors that impact the popularity of GitHub repositories." In: *2016 IEEE international conference on software maintenance and evolution (ICSME)*. IEEE. 2016, pp. 334–344.

[30] Tejas S. Borkar and Lina J. Karam. "DeepCorrect: Correcting DNN Models Against Image Distortions." In: *IEEE Transactions on Image Processing* 28.12 (2019), pp. 6022–6034. DOI: 10.1109/TIP.2019.2924172.

[31] Jan Bosch, Helena Holmström Olsson, and Ivica Crnkovic. "Engineering ai systems: A research agenda." In: *Artificial intelligence paradigms for smart cyber-physical systems* (2021), pp. 1–19.

[32] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. "Language models are few-shot learners." In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[33] Bernd Bruegge and Allen H Dutoit. “Object–oriented software engineering. using uml, patterns, and java.” In: *Learning* 5.6 (2009), p. 7.

[34] Yuriy Brun and Alexandra Meliou. “Software fairness.” In: *Proceedings of the 2018 26th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering*. 2018, pp. 754–759.

[35] Gro Hariem Brundtland. “World commission on environment and development.” In: *Environmental policy and law* 14.1 (1985), pp. 26–30.

[36] Erik Brynjolfsson and Tom Mitchell. “What can machine learning do? Workforce implications.” In: *Science* 358.6370 (2017), pp. 1530–1534.

[37] Joy Buolamwini and Timnit Gebru. “Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification.” In: *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*. Ed. by Sorelle A. Friedler and Christo Wilson. Vol. 81. Proceedings of Machine Learning Research. PMLR, 2018, pp. 77–91. URL: https://proceedings.mlr.press/v81/buolamwini18a.html.

[38] Toon Calders and Sicco Verwer. “Three naive bayes approaches for discrimination-free classification.” In: *Data mining and knowledge discovery* 21 (2010), pp. 277–292.

[39] Victor R Basili1 Gianluigi Caldiera and H Dieter Rombach. “The goal question metric approach.” In: *Encyclopedia of software engineering* (1994), pp. 528–532.

[40] Fabio Calefato, Filippo Lanubile, and Luigi Quaranta. “A preliminary investigation of MLOps practices in GitHub.” In: *Proceedings of the 16th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. 2022, pp. 283–288.

[41] Coral Calero, Manuel F Bertoa, and Ma Ángeles Moraga. "A systematic literature review for software sustainability measures." In: *2013 2nd international workshop on green and sustainable software (GREENS)*. IEEE. 2013, pp. 46–53.

[42] Antonio Candelieri, Andrea Ponti, and Francesco Archetti. "Fair and green hyperparameter optimization via multi-objective and multiple information source Bayesian optimization." In: *Machine Learning* 113.5 (2024), pp. 2701–2731.

[43] Junming Cao, Bihuan Chen, Longjie Hu, Jie Gao, Kaifeng Huang, Xuezhi Song, and Xin Peng. "Characterizing the Complexity and Its Impact on Testing in ML-Enabled Systems: A Case Sutdy on Rasa." In: *2023 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE. 2023, pp. 258–270.

[44] Nicholas Carlini and David Wagner. "Towards Evaluating the Robustness of Neural Networks." In: *2017 IEEE Symposium on Security and Privacy (SP)*. 2017, pp. 39–57. DOI: 10.1109/SP.2017.49.

[45] Bruno Cartaxo, Gustavo Pinto, and Sergio Soares. "Rapid reviews in software engineering." In: *Contemporary Empirical Methods in Software Engineering* (2020), pp. 357–384.

[46] Joel Castaño, Rafael Cabañas, Antonio Salmerón, David Lo, and Silverio Martínez-Fernández. "How do machine learning models change?" In: *arXiv preprint arXiv:2411.09645* (2024).

[47] Joel Castaño, Silverio Martínez-Fernández, Xavier Franch, and Justus Bogner. "Exploring the carbon footprint of hugging face's ml models: A repository mining study." In: *2023 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE. 2023, pp. 1–12.

[48] Roger Creus Castanyer, Silverio Martínez-Fernández, and Xavier Franch. "Which design decisions in AI-enabled mobile applications contribute to greener AI?" In: *Empirical Software Engineering* 29.1 (2024), pp. 1–34.

[49] L Elisa Celis, Lingxiao Huang, Vijay Keswani, and Nisheeth K Vishnoi. "Classification with fairness constraints: A meta-algorithm with provable guarantees." In: *Proceedings of the conference on fairness, accountability, and transparency*. 2019, pp. 319–328.

[50] Joymallya Chakraborty, Suvodeep Majumder, and Tim Menzies. "Bias in Machine Learning Software: Why? How? What to Do?" In: *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2021. Association for Computing Machinery, 2021, 429–440. ISBN: 9781450385626. DOI: 10.1145/3468264.3468537. URL: https://doi.org/10.1145/3468264.3468537.

[51] Joymallya Chakraborty, Suvodeep Majumder, Zhe Yu, and Tim Menzies. "Fairway: A Way to Build Fair ML Software." In: *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2020. New York, NY, USA: Association for Computing Machinery, 2020, 654–665. ISBN: 9781450370431. DOI: 10.1145/3368089.3409697. URL: https://doi.org/10.1145/3368089.3409697.

[52] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. "A survey on evaluation of large language models." In: *ACM Transactions on Intelligent Systems and Technology* 15.3 (2024), pp. 1–45.

[53] *ChatGPT API Keys*. https://openai.com/index/openai-api/. 2024.

[54] Preetha Chatterjee, Mia Mohammad Imran, and Kostadin Damevski. "Shedding Light on Software Engineering-specific Metaphors and Idioms." In: *ICSE'24: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. Association for Computing Machinery. 2024, pp. 1–13.

[55] Boyuan Chen, Mingzhi Wen, Yong Shi, Dayi Lin, Gopi Krishnan Rajbahadur, and Zhen Ming (Jack) Jiang. "Towards Training Reproducible Deep Learning Models." In: *Proceedings of the 44th International Conference on Software Engineering*. ICSE '22. Association for Computing Machinery, 2022, 2202–2214. ISBN: 9781450392211. DOI: 10.1145/3510003.3510163. URL: https://doi.org/10.1145/3510003.3510163.

[56] Chien-fei Chen, Rebecca Napolitano, Yuqing Hu, Bandana Kar, and Bing Yao. "Addressing machine learning bias to foster energy justice." In: *Energy Research & Social Science* 116 (2024), p. 103653.

[57] Fangwei Chen, Lei Li, Jing Jiang, and Li Zhang. "Predicting the number of forks for open source software project." In: *Proceedings of the 2014 3rd International workshop on evidential assessment of software technologies*. 2014, pp. 40–47.

[58] Huiqiang Chen, Tianqing Zhu, Tao Zhang, Wanlei Zhou, and Philip S Yu. "Privacy and fairness in Federated learning: on the perspective of Tradeoff." In: *ACM Computing Surveys* 56.2 (2023), pp. 1–37.

[59] Simin Chen, Mirazul Haque, Cong Liu, and Wei Yang. "DeepPerform: An Efficient Approach for Performance Testing of Resource-Constrained Neural Networks." In: *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. ASE '22. New York, NY, USA: Association for Computing Machinery, 2023. ISBN: 9781450394758. DOI: 10.1145/3551349.3561158. URL: https://doi.org/10.1145/3551349.3561158.

[60] Simin Chen, Cong Liu, Mirazul Haque, Zihe Song, and Wei Yang. "NMTSloth: Understanding and Testing Efficiency Degradation of Neural Machine Translation Systems." In: *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2022. New York, NY, USA: Association for Computing Machinery,

2022, 1148–1160. ISBN: 9781450394130. DOI: 10.1145/3540250.3549102. URL: https://doi.org/10.1145/3540250.3549102.

[61] Simin Chen, Zihe Song, Mirazul Haque, Cong Liu, and Wei Yang. "NICGSlowDown: Evaluating the Efficiency Robustness of Neural Image Caption Generation Models." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 15365–15374.

[62] Zhenpeng Chen, Yanbin Cao, Yuanqiang Liu, Haoyu Wang, Tao Xie, and Xuanzhe Liu. "A comprehensive study on challenges in deploying deep learning based software." In: *Proceedings of the 28th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering*. 2020, pp. 750–762.

[63] Zhenpeng Chen, Jie M. Zhang, Federica Sarro, and Mark Harman. "MAAT: A Novel Ensemble Approach to Addressing Fairness and Performance Bugs for Machine Learning Software." In: *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2022. New York, NY, USA: Association for Computing Machinery, 2022, 1122–1134. ISBN: 9781450394130. DOI: 10.1145/3540250.3549093. URL: https://doi.org/10.1145/3540250.3549093.

[64] Zhenpeng Chen, Jie M Zhang, Federica Sarro, and Mark Harman. "A Comprehensive Empirical Study of Bias Mitigation Methods for Machine Learning Classifiers." In: *ACM Transactions on Software Engineering and Methodology* 32.4 (2023), pp. 1–30.

[65] Zhenpeng Chen, Jie Zhang, Federica Sarro, and Mark Harman. "Fairness Improvement with Multiple Protected Attributes: How Far Are We?" In: *46th International Conference on Software Engineering (ICSE 2024)*. ACM. 2023.

[66] Min-Chi Chiang, Lu-Wen Zhang, Yu-Min Chou, and Jerry Chou. "Dynamic Resource Management for Machine Learning Pipeline Workloads." In: *SN Computer Science* 4.5 (2023), p. 665.

[67] Ruzanna Chitchyan, Christoph Becker, Stefanie Betz, Leticia Duboc, Birgit Penzenstadler, Norbert Seyff, and Colin C Venters. "Sustainability design in requirements engineering: state of practice." In: *Proceedings of the 38th International Conference on Software Engineering Companion*. 2016, pp. 533–542.

[68] Marcus Ciolkowski, Oliver Laitenberger, Sira Vegas, and Stefan Biffl. *Practical experiences in the design and conduct of surveys in empirical software engineering*. Springer, 2003.

[69] Jurgen Cito, Isil Dillig, Seohyun Kim, Vijayaraghavan Murali, and Satish Chandra. "Explaining mispredictions of machine learning models using rule induction." In: *Proceedings of the 29th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering*. 2021, pp. 716–727.

[70] Jacob Cohen. "A coefficient of agreement for nominal scales." In: *Educational and psychological measurement* 20.1 (1960), pp. 37–46.

[71] Giuseppe Colavito, Filippo Lanubile, and Nicole Novielli. "Benchmarking large language models for automated labeling: The case of issue report classification." In: *Information and Software Technology* (2025), p. 107758.

[72] William Jay Conover. *Practical nonparametric statistics*. Vol. 350. john wiley & sons, 1999.

[73] Marcelo Costalonga, Bianca Minetto Napoleão, Maria Teresa Baldassarre, Katia Romero Felizardo, Igor Steinmacher, and Marcos Kalinowski. "Can Machine Learning Support the Selection of Studies for Systematic Literature Review Updates?" In: *arXiv preprint arXiv:2502.08050* (2025).

[74] John W Creswell and Cheryl N Poth. *Qualitative inquiry and research design: Choosing among five approaches*. Sage publications, 2016.

[75] Luís Cruz. *Green Software Engineering Done Right: a Scientific Guide to Set Up Energy Efficiency Experiments*. http://luiscruz.github.io/2021/10/10/scientific-guide.html. Blog post. 2021. DOI: 10.6084/m9.figshare.22067846.v1.

[76] Daniela S Cruzes and Tore Dybå. "Recommended steps for thematic synthesis in software engineering." In: *International Symposium on Empirical Software Engineering and Measurement (ESEM)* (2011), pp. 275–284.

[77] Luiz Marcio Cysneiros, Majid Raffi, and Julio Cesar Sampaio do Prado Leite. "Software transparency as a key requirement for self-driving cars." In: *2018 IEEE 26th international requirements engineering conference (RE)*. IEEE. 2018, pp. 382–387.

[78] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. "Social coding in GitHub: transparency and collaboration in an open software repository." In: *Proceedings of the ACM 2012 conference on computer supported cooperative work*. 2012, pp. 1277–1286.

[79] Vincenzo De Martino, Joel Castano, Fabio Palomba, Xavier Franch, and Silverio Martınez-Fernández. "A Framework for Using LLMs for Repository Mining Studies in Empirical Software Engineering." In: *arXiv preprint arXiv:2411.09974* (2024).

[80] Vincenzo De Martino, Joel Castano, Fabio Palomba, Xavier Franch, and Silverio Martınez-Fernández. "A framework for using llms for repository mining studies in empirical software engineering." In: *WSESE@ ICSE* (2025).

[81] Vincenzo De Martino, Stefano Lambiase, Fabiano Pecorelli, Fabio Palomba, and Filomena Ferrucci. *Sustainability of Machine Learning-Enabled Systems: The Machine Learning Practitioner's Perspective*. 2023. DOI: 10.6084/m9.figshare.24249631. URL: https://figshare.com/articles/dataset/_/24249631.

[82] Vincenzo De Martino, Silverio Martínez-Fernández, and Fabio Palomba. "Do developers adopt green architectural tactics for ml-enabled systems? a mining software repository study." In: *arXiv preprint arXiv:2410.06708* (2024).

[83] Vincenzo De Martino and Fabio Palomba. "Classification and challenges of non-functional requirements in ML-enabled systems: A systematic literature review." In: *Information and Software Technology* (2025), p. 107678.

[84] Vincenzo De Martino, Gilberto Recupito, Giammaria Giordano, Filomena Ferrucci, Dario Di Nucci, and Fabio Palomba. "Into the ml-universe: An improved classification and characterization of machine-learning projects." In: *Available at SSRN 4960056* ().

[85] Vincenzo De Martino, Gianmario Voria, Ciro Troiano, Gemma Catolino, and Fabio Palomba. *Examining the Impact of Bias Mitigation Algorithms on the Sustainability of ML-enabled Systems*. 2024. DOI: 10.6084/m9.figshare.25673658. URL: https://figshare.com/articles/dataset/Examining_the_Impact_of_Bias_Mitigation_Algorithms_on_the_Sustainability_of_ML-enabled_Systems/25673658.

[86] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, et al. "Large scale distributed deep networks." In: *Advances in neural information processing systems* 25 (2012).

[87] Janez Demšar. "Statistical comparisons of classifiers over multiple data sets." In: *The Journal of Machine learning research* 7 (2006), pp. 1–30.

[88] Wesley Hanwen Deng, Manish Nagireddy, Michelle Seng Ah Lee, Jatinder Singh, Zhiwei Steven Wu, Kenneth Holstein, and Haiyi Zhu. "Exploring How Machine Learning Practitioners (Try To) Use Fairness Toolkits." In: *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*. FAccT '22. New York, NY, USA: Association for Computing Machinery, 2022, 473–484.

ISBN: 9781450393522. DOI: 10.1145/3531146.3533113. URL: https://doi.org/10.1145/3531146.3533113.

[89] Jesse Dillard, Veronica Dujon, and Mary C King. *Understanding the social dimension of sustainability*. Routledge, 2008.

[90] Don A Dillman, Jolene D Smyth, and Leah Melani Christian. *Internet, phone, mail, and mixed-mode surveys: The tailored design method*. John Wiley & Sons, 2014.

[91] Simone Disabato and Manuel Roveri. "Tiny machine learning for concept drift." In: *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[92] Guoliang Dong, Jingyi Wang, Jun Sun, Yang Zhang, Xinyu Wang, Ting Dai, Jin Song Dong, and Xingen Wang. "Towards interpreting recurrent neural networks through probabilistic abstraction." In: *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*. 2020, pp. 499–510.

[93] Yihong Dong, Xue Jiang, Zhi Jin, and Ge Li. "Self-collaboration code generation via chatgpt." In: *ACM Transactions on Software Engineering and Methodology* 33.7 (2024), pp. 1–38.

[94] Xiaoning Du, Xiaofei Xie, Yi Li, Lei Ma, Yang Liu, and Jianjun Zhao. "DeepStellar: Model-Based Quantitative Analysis of Stateful Deep Learning Systems." In: *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2019. Association for Computing Machinery, 2019, 477–487. ISBN: 9781450355728. DOI: 10.1145/3338906.3338954. URL: https://doi.org/10.1145/3338906.3338954.

[95] Isaac Dunn, Hadrien Pouget, Daniel Kroening, and Tom Melham. "Exposing previously undetectable faults in deep neural networks." In: *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 2021, pp. 56–66.

[96] Francisco Durán, Silverio Martínez-Fernández, Matias Martinez, and Patricia Lago. “Identifying architectural design decisions for achieving green ML serving.” In: *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI*. 2024, pp. 18–23.

[97] Anurag Dwarakanath, Manish Ahuja, Samarth Sikand, Raghotham M Rao, RP Jagadeesh Chandra Bose, Neville Dubash, and Sanjay Podder. “Identifying implementation bugs in machine learning based image classifiers using metamorphic testing.” In: *Proceedings of the 27th ACM SIGSOFT international symposium on software testing and analysis*. 2018, pp. 118–128.

[98] Felipe Ebert, Alexander Serebrenik, Christoph Treude, Nicole Novielli, and Fernando Castor. “On Recruiting Experienced GitHub Contributors for Interviews and Surveys on Prolific.” In: *International Workshop on Recruiting Participants for Empirical Software Engineering*. 2022.

[99] Peer Eyal, Rothschild David, Gordon Andrew, Evernden Zak, and Damer Ekaterina. “Data quality of platforms and panels for online behavioral research.” In: *Behavior research methods* (2021), pp. 1–20.

[100] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. “Robust Physical-World Attacks on Deep Learning Visual Classification.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[101] Hazem Fahmy, Fabrizio Pastore, Mojtaba Bagherzadeh, and Lionel Briand. “Supporting Deep Neural Network Safety Analysis and Retraining Through Heatmap-Based Unsupervised Learning.” In: *IEEE Transactions on Reliability* 70.4 (2021), pp. 1641–1657. DOI: 10.1109/TR.2021.3074750.

[102] Hazem Fahmy, Fabrizio Pastore, Lionel Briand, and Thomas Stifter. “Simulator-based explanation and debugging of hazard-triggering

events in DNN-based safety-critical systems." In: *ACM Transactions on Software Engineering and Methodology* 32.4 (2023), pp. 1–47.

[103] Angela Fan, Beliz Gokkaya, Mark Harman, Mitya Lyubarskiy, Shubho Sengupta, Shin Yoo, and Jie M Zhang. "Large language models for software engineering: Survey and open problems." In: *2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE)*. IEEE. 2023, pp. 31–53.

[104] Wagner Felidré, Leonardo Furtado, Daniel A Da Costa, Bruno Cartaxo, and Gustavo Pinto. "Continuous integration theater." In: *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE. 2019, pp. 1–10.

[105] Katia Romero Felizardo, Anderson Deizepe, Daniel Coutinho, Genildo Gomes, Maria Meireles, Marco Gerosa, and Igor Steinmacher. "On the Difficulties of Conducting and Replicating Systematic Literature Reviews Studies Using LLMs in Software Engineering." In: *2025 IEEE/ACM International Workshop on Methodological Issues with Empirical Studies in Software Engineering (WSESE)*. IEEE. 2025, pp. 20–23.

[106] Carmine Ferrara, Giulia Sellitto, Filomena Ferrucci, Fabio Palomba, and Andrea De Lucia. "Fairness-aware machine learning engineering: how far are we?" In: *Empirical Software Engineering* 29.1 (2024), p. 9.

[107] Jade S Franklin, Karan Bhanot, Mohamed Ghalwash, Kristin P Bennett, Jamie McCusker, and Deborah L McGuinness. "An Ontology for Fairness Metrics." In: *Proceedings of the 2022 AAAI/ACM Conference on AI, Ethics, and Society*. 2022, pp. 265–275.

[108] Wei Fu and Tim Menzies. "Easy over hard: A case study on deep learning." In: *Proceedings of the 2017 11th joint meeting on foundations of software engineering*. 2017, pp. 49–60.

[109] Sainyam Galhotra, Yuriy Brun, and Alexandra Meliou. "Fairness Testing: Testing Software for Discrimination." In: *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*.

ESEC/FSE 2017. New York, NY, USA: Association for Computing Machinery, 2017, 498–510. ISBN: 9781450351058. DOI: 10.1145/3106237.3106277. URL: https://doi.org/10.1145/3106237.3106277.

[110] Xinyu Gao, Yang Feng, Yining Yin, Zixi Liu, Zhenyu Chen, and Baowen Xu. "Adaptive test selection for deep neural networks." In: *Proceedings of the 44th International Conference on Software Engineering*. 2022, pp. 73–85.

[111] Xinyu Gao, Zhijie Wang, Yang Feng, Lei Ma, Zhenyu Chen, and Baowen Xu. "Benchmarking Robustness of AI-Enabled Multi-Sensor Fusion Systems: Challenges and Opportunities." In: *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2023, pp. 871–882.

[112] Xuanqi Gao, Juan Zhai, Shiqing Ma, Chao Shen, Yufei Chen, and Qian Wang. "FairNeuron: Improving Deep Neural Network Fairness with Adversary Games on Selective Neurons." In: *Proceedings of the 44th International Conference on Software Engineering*. ICSE '22. Association for Computing Machinery, 2022, 921–933. ISBN: 9781450392211. DOI: 10.1145/3510003.3510087. URL: https://doi.org/10.1145/3510003.3510087.

[113] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. "AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation." In: *2018 IEEE Symposium on Security and Privacy (SP)*. 2018, pp. 3–18. DOI: 10.1109/SP.2018.00058.

[114] Stefanos Georgiou, Maria Kechagia, Tushar Sharma, Federica Sarro, and Ying Zou. "Green AI: Do Deep Learning Frameworks Have Different Costs?" In: *Proceedings of the 44th International Conference on Software Engineering*. ICSE '22. Association for Computing Machinery, 2022, 1082–1094. ISBN: 9781450392211. DOI: 10.1145/351

0003.3510221. URL: https://doi.org/10.1145/3510003.3510221.

[115] *Getting Started - Anthropic*. https://docs.anthropic.com/en/api/getting-started. 2024.

[116] Salah Ghamizi, Maxime Cordy, Martin Gubri, Mike Papadakis, Andrey Boystov, Yves Le Traon, and Anne Goujon. "Search-Based Adversarial Testing and Improvement of Constrained Credit Scoring Systems." In: *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2020. New York, NY, USA: Association for Computing Machinery, 2020, 1089–1100. ISBN: 9781450370431. DOI: 10.1145/3368089.3409739. URL: https://doi.org/10.1145/3368089.3409739.

[117] Avijit Ghosh, Lea Genuit, and Mary Reagan. "Characterizing intersectional group fairness with worst-case comparisons." In: *Artificial Intelligence Diversity, Belonging, Equity, and Inclusion*. PMLR. 2021, pp. 22–34.

[118] Görkem Giray. "A software engineering perspective on engineering machine learning systems: State of the art and challenges." In: *Journal of Systems and Software* 180 (2021), p. 111031.

[119] *GitHub - IINemo/isanlp_srl_framebank: SRL parser for Russian based on FrameBank corpus — github.com*. https://github.com/IINemo/isanlp_srl_framebank.

[120] *GitHub - ambakick/Person-Detection-and-Tracking: A tensorflow implementation with SSD model for person detection and Kalman Filtering combined for tracking — github.com*. https://github.com/ambakick/Person-Detection-and-Tracking.

[121] *GitHub - bio-ontology-research-group/onto2vec: Representation learning for ontologies and their annotations — github.com*. https://github.com/bio-ontology-research-group/onto2vec.

[122] *GitHub - bio-ontology-research-group/onto2vec: Representation learning for ontologies and their annotations — github.com*. https://github.com/bio-ontology-research-group/onto2vec.

[123] *GitHub - daniel-cortez-stevenson/crypto-predict: A dockerized prediction API for crypto. — github.com*. https://github.com/daniel-cortez-stevenson/crypto-predict.

[124] *GitHub - devpranoy/video_coloriser: Pytorch Convolutional Neural Net and GAN based video coloriser that converts black and white video to colorised video. — github.com*. https://github.com/devpranoy/video_coloriser.

[125] *GitHub - openvinotoolkit/open_model_zoo: Pre-trained Deep Learning models and demos (high quality and extremely fast) — github.com*. https://github.com/openvinotoolkit/open_model_zoo.

[126] *GitHub - propublica/compas-analysis: Data and analysis for 'Machine Bias' — github.com*. https://github.com/propublica/compas-analysis. [Accessed 10-02-2024].

[127] *GitHub - sipeed/MaixPy-v1_scripts: micropython scripts for MaixPy — github.com*. https://github.com/sipeed/MaixPy-v1_scripts.

[128] *GitHub REST API documentation - GitHub Docs — docs.github.com*. https://docs.github.com/en/rest. 2022.

[129] Barney Glaser and Anselm Strauss. *Discovery of grounded theory: Strategies for qualitative research*. Routledge, 2017.

[130] Usman Gohar and Lu Cheng. "A Survey on Intersectional Fairness in Machine Learning: Notions, Mitigation, and Challenges." In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. IJCAI-2023. International Joint Conferences on Artificial Intelligence Organization, Aug. 2023, 6619–6627. DOI: 10.24963/ijcai.2023/742. URL: http://dx.doi.org/10.24963/ijcai.2023/742.

[131] Laurent Gomez., Marcus Wilhelm., José Márquez., and Patrick Duverger. “Security for Distributed Deep Neural Networks: Towards Data Confidentiality & Intellectual Property Protection.” In: *Proceedings of the 16th International Joint Conference on e-Business and Telecommunications - SECRYPT,* INSTICC. SciTePress, 2019, pp. 439–447. ISBN: 978-989-758-378-0. DOI: 10.5220/0007922404390447.

[132] Jesus M Gonzalez-Barahona, Gregorio Robles, Martin Michlmayr, Juan José Amor, and Daniel M German. “Macro-level software evolution: a case study of a large software compilation.” In: *Empirical Software Engineering* 14 (2009), pp. 262–285.

[133] Elizabeth González-Estrada and Waldenia Cosmes. “Shapiro–Wilk test for skew normal distributions based on data transformations.” In: *Journal of Statistical Computation and Simulation* 89.17 (2019), pp. 3258–3272.

[134] Alexandra González, Joel Castaño, Xavier Franch, and Silverio Martínez-Fernández. “Impact of ML optimization tactics on greener pre-trained ML models.” In: *Computing* 107.4 (2025), p. 103.

[135] Alexandra González, Xavier Franch, David Lo, and Silverio Martínez-Fernández. “How do Pre-Trained Models Support Software Engineering? An Empirical Study in Hugging Face.” In: *arXiv preprint arXiv:2506.03013* (2025).

[136] Danielle Gonzalez, Thomas Zimmermann, and Nachiappan Nagappan. “The state of the ml-universe: 10 years of artificial intelligence & machine learning software development on github.” In: *Proceedings of the 17th International conference on mining software repositories*. 2020, pp. 431–442.

[137] Pamela Grimm. “Social desirability bias.” In: *Wiley international encyclopedia of marketing* (2010).

[138] Iris Groher and Rainer Weinreich. “An Interview Study on Sustainability Concerns in Software Development Projects.” In: *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. 2017, pp. 350–358. DOI: 10.1109/SEAA.2017.70.

[139] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. “BadNets: Evaluating Backdooring Attacks on Deep Neural Networks.” In: *IEEE Access* 7 (2019), pp. 47230–47244. DOI: 10.1109/ACCESS.2019.2909068.

[140] Huizhong Guo, Jinfeng Li, Jingyi Wang, Xiangyu Liu, Dongxia Wang, Zehong Hu, Rong Zhang, and Hui Xue. “FairRec: Fairness testing for deep recommender systems.” In: *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*. 2023, pp. 310–321.

[141] Shikai Guo, Haorui Lin, Jiaoru Zhao, Hui Li, Rong Chen, Xiaochen Li, and He Jiang. “An Easy Data Augmentation Approach for Application Reviews Event Inference.” In: *IEEE Transactions on Software Engineering* (2023).

[142] Wenbo Guo, Dongliang Mu, Jun Xu, Purui Su, Gang Wang, and Xinyu Xing. “LEMNA: Explaining Deep Learning Based Security Applications.” In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’18. New York, NY, USA: Association for Computing Machinery, 2018, 364–379. ISBN: 9781450356930. DOI: 10.1145/3243734.3243792. URL: https://doi.org/10.1145/3243734.3243792.

[143] Umm-e Habiba, Mohammad Kasra Habib, Justus Bogner, Jonas Fritzsch, and Stefan Wagner. “How do ML practitioners perceive explainability? an interview study of practices and challenges.” In: *Empirical Software Engineering* 30.1 (2025), p. 18.

[144] Khan Mohammad Habibullah, Gregory Gay, and Jennifer Horkoff. “Non-functional requirements for machine learning: An exploration of system scope and interest.” In: *2022 IEEE/ACM 1st Interna-*

tional Workshop on Software Engineering for Responsible Artificial Intelligence (SE4RAI). IEEE. 2022, pp. 29–36.

[145] Khan Mohammad Habibullah, Gregory Gay, and Jennifer Horkoff. "Non-functional requirements for machine learning: Understanding current use and challenges among practitioners." In: *Requirements Engineering* 28.2 (2023), pp. 283–316.

[146] Khan Mohammad Habibullah and Jennifer Horkoff. "Non-functional requirements for machine learning: understanding current use and challenges in industry." In: *2021 IEEE 29th International Requirements Engineering Conference (RE)*. IEEE. 2021, pp. 13–23.

[147] Gaétan Hains, Arvid Jakobsson, and Youry Khmelevsky. "Towards formal methods and software engineering for deep learning: security, safety and productivity for dl systems development." In: *2018 Annual IEEE international systems conference (syscon)*. IEEE. 2018, pp. 1–5.

[148] Joseph F Hair, Arthur H Money, Philip Samouel, and Mike Page. "Research methods for business." In: *Education+ Training* 49.4 (2007), pp. 336–337.

[149] Raluca Maria Hampau, Maurits Kaptein, Robin van Emden, Thomas Rost, and Ivano Malavolta. "An Empirical Study on the Performance and Energy Consumption of AI Containerization Strategies for Computer-Vision Tasks on the Edge." In: *Proceedings of the International Conference on Evaluation and Assessment in Software Engineering 2022*. EASE '22. New York, NY, USA: Association for Computing Machinery, 2022, 50–59. ISBN: 9781450396134. DOI: 10.1145/3530019.3530025. URL: https://doi.org/10.1145/3530019.3530025.

[150] Mirazul Haque, Yaswanth Yadlapalli, Wei Yang, and Cong Liu. "EREBA: Black-Box Energy Testing of Adaptive Neural Networks." In: *Proceedings of the 44th International Conference on Software Engineering*. ICSE '22. New York, NY, USA: Association for Comput-

ing Machinery, 2022, 835–846. ISBN: 9781450392211. DOI: 10.1145/3510003.3510088. URL: https://doi.org/10.1145/3510003.3510088.

[151] Michaela Hardt et al. "Amazon SageMaker Clarify: Machine Learning Bias Detection and Explainability in the Cloud." In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. KDD '21. New York, NY, USA: Association for Computing Machinery, 2021, 2974–2983. ISBN: 9781450383325. DOI: 10.1145/3447548.3467177. URL: https://doi.org/10.1145/3447548.3467177.

[152] Agency for Healthcare Research and Quality. *Medical Expenditure Panel Survey Public Use File Details — meps.ahrq.gov*. https://meps.ahrq.gov/mepsweb/data_stats/download_data_files_detail.jsp?cboPufNumber=HC-181. [Accessed 08-01-2024].

[153] Patrick Henriksen, Francesco Leofante, and Alessio Lomuscio. "Repairing misclassifications in neural networks using limited data." In: *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*. 2022, pp. 1031–1038.

[154] *Here's How to Correct Deep Learning's Unsustainability — spectrum.ieee.org*. https://spectrum.ieee.org/deep-learning-sustainability.

[155] Abram Hindle, Daniel M German, and Ric Holt. "What do large commits tell us? a taxonomical study of large commits." In: *Proceedings of the 2008 international working conference on Mining software repositories*. 2008, pp. 99–108.

[156] Lorin Hochstein. "Why Don't We See Even More Failures?" In: *IEEE Software* 40.4 (2023), pp. 114–116.

[157] Rashina Hoda. "Socio-Technical Grounded Theory for Software Engineering." In: *IEEE Transactions on Software Engineering* 48.10 (2022), pp. 3808–3832. DOI: 10.1109/TSE.2021.3106280.

[158] Hans Hofmann. *Statlog (German Credit Data)*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5NC77. 1994.

[159] Myles Hollander, Douglas A Wolfe, and Eric Chicken. *Nonparametric statistical methods*. John Wiley & Sons, 2013.

[160] Sture Holm. "A simple sequentially rejective multiple test procedure." In: *Scandinavian journal of statistics* (1979), pp. 65–70.

[161] Jennifer Horkoff. "Non-functional requirements for machine learning: Challenges and new directions." In: *2019 IEEE 27th International Requirements Engineering Conference (RE)*. IEEE. 2019, pp. 386–391.

[162] Max Hort, Zhenpeng Chen, Jie M. Zhang, Mark Harman, and Federica Sarro. "Bias Mitigation for Machine Learning Classifiers: A Comprehensive Survey." In: *ACM J. Responsib. Comput.* 1.2 (2024). DOI: 10.1145/3631326. URL: https://doi.org/10.1145/3631326.

[163] Max Hort, Jie M. Zhang, Federica Sarro, and Mark Harman. "Fairea: A Model Behaviour Mutation Approach to Benchmarking Bias Mitigation Methods." In: *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2021. New York, NY, USA: Association for Computing Machinery, 2021, 994–1006. ISBN: 9781450385626. DOI: 10.1145/3468264.3468565. URL: https://doi.org/10.1145/3468264.3468565.

[164] Max Hort, Jie M Zhang, Federica Sarro, and Mark Harman. "Search-based automatic repair for fairness and accuracy in decision-making software." In: *Empirical Software Engineering* 29.1 (2024), p. 36.

[165] Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. "Large language models for software engineering: A systematic literature review." In: *arXiv preprint arXiv:2308.10620* (2023).

[166] Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. "Large language models for software engineering: A systematic literature review."

In: *ACM Transactions on Software Engineering and Methodology* 33.8 (2024), pp. 1–79.

[167] Siw Elisabeth Hove and Bente Anda. "Experiences from conducting semi-structured interviews in empirical software engineering research." In: *11th IEEE International Software Metrics Symposium (METRICS'05)*. IEEE. 2005, 10–pp.

[168] *How Do We Make Generative AI Green? — accenture.com*. https://www.accenture.com/us-en/blogs/consulting/making-generative-ai-green.

[169] Qiang Hu, Yuejun Guo, Xiaofei Xie, Maxime Cordy, Mike Papadakis, Lei Ma, and Yves Le Traon. "Aries: Efficient testing of deep neural networks via labeling-free accuracy estimation." In: *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE. 2023, pp. 1776–1787.

[170] Yutao Hu, Suyuan Wang, Wenke Li, Junru Peng, Yueming Wu, Deqing Zou, and Hai Jin. "Interpreters for GNN-based vulnerability detection: Are we there yet?" In: *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*. 2023, pp. 1407–1419.

[171] Ling Huang, Anthony D. Joseph, Blaine Nelson, Benjamin I.P. Rubinstein, and J. D. Tygar. "Adversarial Machine Learning." In: *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*. AISec '11. New York, NY, USA: Association for Computing Machinery, 2011, 43–58. ISBN: 9781450310031. DOI: 10.1145/2046684.2046692. URL: https://doi.org/10.1145/2046684.2046692.

[172] Pei Huang, Yuting Yang, Minghao Liu, Fuqi Jia, Feifei Ma, and Jian Zhang. "Weakened Robustness of Deep Neural Networks." In: *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*. ISSTA 2022. Association for Computing Machinery, 2022, 126–138. ISBN: 9781450393799. DOI: 10.1145/3533767.3534373. URL: https://doi.org/10.1145/3533767.3534373.

[173] Wei Huang, Youcheng Sun, Xingyu Zhao, James Sharp, Wenjie Ruan, Jie Meng, and Xiaowei Huang. "Coverage-Guided Testing for Recurrent Neural Networks." In: *IEEE Transactions on Reliability* 71.3 (2022), pp. 1191–1206. DOI: 10.1109/TR.2021.3080664.

[174] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. "Safety verification of deep neural networks." In: *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30*. Springer. 2017, pp. 3–29.

[175] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. "Gpipe: Efficient training of giant neural networks using pipeline parallelism." In: *Advances in neural information processing systems* 32 (2019).

[176] Yujin Huang and Chunyang Chen. "Smart App Attack: Hacking Deep Learning Models in Android Apps." In: *IEEE Transactions on Information Forensics and Security* 17 (2022), pp. 1827–1840. DOI: 10.1109/TIFS.2022.3172213.

[177] Yujin Huang, Han Hu, and Chunyang Chen. "Robustness of on-Device Models: Adversarial Attack to Deep Learning Models on Android Apps." In: *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. 2021, pp. 101–110. DOI: 10.1109/ICSE-SEIP52600.2021.00019.

[178] Katherine J Hunt, Natalie Shlomo, and Julia Addington-Hall. "Participant recruitment in sensitive surveys: a comparative trial of 'opt in'versus 'opt out'approaches." In: *BMC Medical Research Methodology* 13.1 (2013), pp. 1–8.

[179] Wiebke Hutiri, Aaron Yi Ding, Fahim Kawsar, and Akhil Mathur. "Tiny, always-on, and fragile: Bias propagation through design choices in on-device machine learning workflows." In: *ACM Trans-*

actions on Software Engineering and Methodology 32.6 (2023), pp. 1–37.

[180] Mia Mohammad Imran, Preetha Chatterjee, and Kostadin Damevski. "Uncovering the causes of emotions in software developer communication using zero-shot llms." In: *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 2024, pp. 1–13.

[181] Sayem Mohammad Imtiaz, Fraol Batole, Astha Singh, Rangeet Pan, Breno Dantas Cruz, and Hridesh Rajan. "Decomposing a recurrent neural network into modules for enabling reusability and replacement." In: *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE. 2023, pp. 1020–1032.

[182] Heli Järvenpää, Patricia Lago, Justus Bogner, Grace Lewis, Henry Muccini, and Ipek Ozkaya. "A synthesis of green architectural tactics for ml-enabled systems." In: *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Society*. 2024, pp. 130–141.

[183] Pin Ji, Yang Feng, Jia Liu, Zhihong Zhao, and Zhenyu Chen. "ASRTest: Automated Testing for Deep-Neural-Network-Driven Speech Recognition Systems." In: *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*. ISSTA 2022. Association for Computing Machinery, 2022, 189–201. ISBN: 9781450393799. DOI: 10.1145/3533767.3534391. URL: https://doi.org/10.1145/3533767.3534391.

[184] Zhenlan Ji, Pingchuan Ma, Shuai Wang, and Yanhui Li. "Causality-aided trade-off analysis for machine learning fairness." In: *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE. 2023, pp. 371–383.

[185] Wenxin Jiang, Jerin Yasmin, Jason Jones, Nicholas Synovic, Jiashen Kuo, Nathaniel Bielanski, Yuan Tian, George K Thiruvathukal, and James C Davis. "PeaTMOSS: A Dataset and Initial Analysis of Pre-Trained Models in Open-Source Software." In: *Proceedings of the*

21st International Conference on Mining Software Repositories. 2024, pp. 431–443.

[186] R Burke Johnson, Anthony J Onwuegbuzie, and Lisa A Turner. "Toward a definition of mixed methods research." In: *Journal of mixed methods research* 1.2 (2007), pp. 112–133.

[187] Heli Järvenpää, Patricia Lago, Justus Bogner, Grace Lewis, Henry Muccini, and Ipek Ozkaya. *A Synthesis of Green Architectural Tactics for ML-Enabled Systems*. 2023. arXiv: 2312.09610 [cs.SE].

[188] Faisal Kamiran and Toon Calders. "Data preprocessing techniques for classification without discrimination." In: *Knowledge and information systems* 33.1 (2012), pp. 1–33.

[189] Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma. "Fairness-aware classifier with prejudice remover regularizer." In: *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2012, Bristol, UK, September 24-28, 2012. Proceedings, Part II 23*. Springer. 2012, pp. 35–50.

[190] Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. "Preventing fairness gerrymandering: Auditing and learning for subgroup fairness." In: *International conference on machine learning*. PMLR. 2018, pp. 2564–2572.

[191] Aaiza Khan, Isma Farah Siddiqui, Mehwish Shaikh, Shabana Anwar, and Murk Shaikh. "Handling non-fuctional requirements in IoT-based machine learning systems." In: *2022 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT & NCON)*. IEEE. 2022, pp. 477–479.

[192] Jan Kietzmann, Linda W Lee, Ian P McCarthy, and Tim C Kietzmann. "Deepfakes: Trick or treat?" In: *Business horizons* 63.2 (2020), pp. 135–146.

[193] Minseo Kim, Taemin Kim, Thu Hoang Anh Vo, Yugyeong Jung, and Uichin Lee. "Exploring Modular Prompt Design for Emotion and Mental Health Recognition." In: *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. 2025, pp. 1–18.

[194] Minsu Kim, Walid Saad, Mohammad Mozaffari, and Merouane Debbah. "On the tradeoff between energy, precision, and accuracy in federated quantized neural networks." In: *ICC 2022-IEEE International Conference on Communications*. IEEE. 2022, pp. 2194–2199.

[195] Young Geun Kim and Carole-Jean Wu. "Autoscale: Energy efficiency optimization for stochastic edge inference using reinforcement learning." In: *2020 53rd Annual IEEE/ACM international symposium on microarchitecture (MICRO)*. IEEE. 2020, pp. 1082–1096.

[196] Jóakim v. Kistowski, Jeremy A Arnold, Karl Huppler, Klaus-Dieter Lange, John L Henning, and Paul Cao. "How to build a benchmark." In: *Proceedings of the 6th ACM/SPEC international conference on performance engineering*. 2015, pp. 333–336.

[197] Barbara A Kitchenham, Tore Dyba, and Magne Jorgensen. "Evidence-based software engineering." In: *Proceedings. 26th International Conference on Software Engineering*. IEEE. 2004, pp. 273–281.

[198] Barbara A Kitchenham and Shari L Pfleeger. "Personal Opinion Surveys." In: *Guide to advanced empirical software engineering*. Springer, 2008, pp. 63–92.

[199] Barbara Kitchenham, O Pearl Brereton, David Budgen, Mark Turner, John Bailey, and Stephen Linkman. "Systematic literature reviews in software engineering–a systematic literature review." In: *Information and software technology* 51.1 (2009), pp. 7–15.

[200] Barbara Kitchenham and Stuart Charters. "Guidelines for performing Systematic Literature Reviews in Software Engineering." In: 2 (Jan. 2007).

[201] Stefan Koch. “Software evolution in open source projects—a large-scale investigation.” In: *Journal of Software Maintenance and Evolution: Research and Practice* 19.6 (2007), pp. 361–382.

[202] Klaus Krippendorff. *Content analysis: An introduction to its methodology*. 4th ed. Sage publications, 2018.

[203] Fumihiro Kumeno. “Sofware engneering challenges for machine learning applications: A literature review.” In: *Intelligent Decision Technologies* 13.4 (2019), pp. 463–476.

[204] Irwin Kwan, Adrian Schroter, and Daniela Damian. “Does socio-technical congruence have an effect on software build success? a study of coordination in a software project.” In: *IEEE Transactions on Software Engineering* 37.3 (2011), pp. 307–324.

[205] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. “Quantifying the carbon emissions of machine learning.” In: *arXiv preprint arXiv:1910.09700* (2019).

[206] Patricia Lago, Sedef Akinli Koçak, Ivica Crnkovic, and Birgit Penzenstadler. “Framing sustainability as a property of software quality.” In: *Communications of the ACM* 58.10 (2015), pp. 70–78.

[207] Patricia Lago, Per Runeson, Qunying Song, and Roberto Verdecchia. “Threats to validity in software engineering–hypocritical paper section or essential analysis?” In: *Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. 2024, pp. 314–324.

[208] Matheus De Morais Leça, Lucas Valença, Reydne Santos, and Ronnie De Souza Santos. “Applications and Implications of Large Language Models in Qualitative Analysis: A New Frontier for Empirical Software Engineering.” In: *2025 IEEE/ACM International Workshop on Methodological Issues with Empirical Studies in Software Engineering (WSESE)*. IEEE. 2025, pp. 36–43.

[209] Michelle Seng Ah Lee and Jat Singh. "The Landscape and Gaps in Open Source Fairness Toolkits." In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. CHI '21. New York, NY, USA: Association for Computing Machinery, 2021. ISBN: 9781450380966. DOI: 10.1145/3411764.3445261. URL: https://doi.org/10.1145/3411764.3445261.

[210] Philipp Leitner, Erik Wittern, Josef Spillner, and Waldemar Hummer. "A mixed-method empirical study of Function-as-a-Service software development in industrial practice." In: *Journal of Systems and Software* 149 (2019), pp. 340–359.

[211] Guofu Li, Pengjia Zhu, Jin Li, Zhemin Yang, Ning Cao, and Zhiyi Chen. "Security matters: A survey on adversarial machine learning." In: *arXiv preprint arXiv:1810.07339* (2018).

[212] Jia Li, Ge Li, Yongmin Li, and Zhi Jin. "Structured chain-of-thought prompting for code generation." In: *ACM Transactions on Software Engineering and Methodology* 34.2 (2025), pp. 1–23.

[213] Linyi Li, Yuhao Zhang, Luyao Ren, Yingfei Xiong, and Tao Xie. "Reliability assurance for deep neural network architectures against numerical defects." In: *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE. 2023, pp. 1827–1839.

[214] Mingchen Li, Mahdi Soltanolkotabi, and Samet Oymak. "Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks." In: *International conference on artificial intelligence and statistics*. PMLR. 2020, pp. 4313–4324.

[215] Renjue Li, Pengfei Yang, Cheng-Chao Huang, Youcheng Sun, Bai Xue, and Lijun Zhang. "Towards practical robustness analysis for DNNs based on PAC-model learning." In: *Proceedings of the 44th International Conference on Software Engineering*. 2022, pp. 2189–2201.

[216] Ying Li, Yifan Sun, and Adwait Jog. "Path Forward Beyond Simulators: Fast and Accurate GPU Execution Time Prediction for DNN

Workloads." In: *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*. 2023, pp. 380–394.

[217] Yu Li, Muxi Chen, and Qiang Xu. "HybridRepair: Towards Annotation-Efficient Repair for Deep Learning Models." In: *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*. ISSTA 2022. Association for Computing Machinery, 2022, 227–238. ISBN: 9781450393799. DOI: 10.1145/3533767.3534408. URL: https://doi.org/10.1145/3533767.3534408.

[218] Yuanchun Li, Jiayi Hua, Haoyu Wang, Chunyang Chen, and Yunxin Liu. "DeepPayload: Black-box Backdoor Attack on Deep Learning Models through Neural Payload Injection." In: *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. 2021, pp. 263–274. DOI: 10.1109/ICSE43902.2021.00035.

[219] Jenny T Liang, Carmen Badea, Christian Bird, Robert DeLine, Denae Ford, Nicole Forsgren, and Thomas Zimmermann. "Can gpt-4 replicate empirical software engineering research?" In: *Proceedings of the ACM on Software Engineering* 1.FSE (2024), pp. 1330–1353.

[220] Lizhi Liao, Heng Li, Weiyi Shang, and Lei Ma. "An Empirical Study of the Impact of Hyperparameter Tuning and Model Optimization on the Performance Properties of Deep Neural Networks." In: *ACM Trans. Softw. Eng. Methodol.* 31.3 (2022). ISSN: 1049-331X. DOI: 10.1145/3506695. URL: https://doi.org/10.1145/3506695.

[221] Ji Lin, Wei-Ming Chen, Yujun Lin, Chuang Gan, Song Han, et al. "Mcunet: Tiny deep learning on iot devices." In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 11711–11722.

[222] Bo Liu, Ming Ding, Sina Shaham, Wenny Rahayu, Farhad Farokhi, and Zihuai Lin. "When machine learning meets privacy: A survey and outlook." In: *ACM Computing Surveys (CSUR)* 54.2 (2021), pp. 1–36.

[223] Shangqing Liu, Bozhi Wu, Xiaofei Xie, Guozhu Meng, and Yang Liu. "Contrabert: Enhancing code pre-trained models via contrastive learning." In: *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE. 2023, pp. 2476–2487.

[224] Zixi Liu, Yang Feng, Yining Yin, and Zhenyu Chen. "DeepState: Selecting Test Suites to Enhance the Robustness of Recurrent Neural Networks." In: *Proceedings of the 44th International Conference on Software Engineering*. ICSE '22. Association for Computing Machinery, 2022, 598–609. ISBN: 9781450392211. DOI: 10.1145/3510003.3510231. URL: https://doi.org/10.1145/3510003.3510231.

[225] Sin Kit Lo, Qinghua Lu, Chen Wang, Hye-Young Paik, and Liming Zhu. "A systematic literature review on federated machine learning: From a software engineering perspective." In: *ACM Computing Surveys (CSUR)* 54.5 (2021), pp. 1–39.

[226] Cristina V Lopes, Vanessa I Klotzman, Iris Ma, and Iftekar Ahmed. "Commit Messages in the Age of Large Language Models." In: *arXiv preprint arXiv:2401.17622* (2024).

[227] Nikolaos Louloudakis, Perry Gibson, José Cano, and Ajitha Rajan. "DeltaNN: Assessing the impact of computational environment parameters on the performance of image recognition models." In: *2023 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE. 2023, pp. 414–424.

[228] Lucy Ellen Lwakatare, Aiswarya Raj, Ivica Crnkovic, Jan Bosch, and Helena Holmström Olsson. "Large-scale machine learning systems in real-world industrial settings: A review of challenges and solutions." In: *Information and software technology* 127 (2020), p. 106368.

[229] Shiqing Ma, Yingqi Liu, Wen-Chuan Lee, Xiangyu Zhang, and Ananth Grama. "MODE: automated neural network model debugging via state differential analysis and input selection." In: *Proceedings of the 2018 26th ACM Joint Meeting on European Software*

Engineering Conference and Symposium on the Foundations of Software Engineering. 2018, pp. 175–186.

[230] Wei Ma, Mike Papadakis, Anestis Tsakmalis, Maxime Cordy, and Yves Le Traon. "Test selection for deep learning systems." In: *ACM Transactions on Software Engineering and Methodology (TOSEM)* 30.2 (2021), pp. 1–22.

[231] Irene Manotas, Christian Bird, Rui Zhang, David Shepherd, Ciera Jaspan, Caitlin Sadowski, Lori Pollock, and James Clause. "An empirical study of practitioners' perspectives on green software engineering." In: *Proceedings of the 38th international conference on software engineering*. 2016, pp. 237–248.

[232] Niccolò Marini, Leonardo Pampaloni, Filippo Di Martino, Roberto Verdecchia, and Enrico Vicario. "Green AI: Which Programming Language Consumes the Most?" In: *arXiv preprint arXiv:2501.14776* (2024).

[233] Silverio Martínez-Fernández. "Environmental sustainability of machine learning systems: Reducing the carbon impact of their lifecycle process." In: *International Conference on Product-Focused Software Process Improvement*. Springer. 2024, pp. 3–7.

[234] Silverio Martínez-Fernández, Justus Bogner, Xavier Franch, Marc Oriol, Julien Siebert, Adam Trendowicz, Anna Maria Vollmer, and Stefan Wagner. "Software engineering for AI-based systems: a survey." In: *ACM Transactions on Software Engineering and Methodology (TOSEM)* 31.2 (2022), pp. 1–59.

[235] Silverio Martínez-Fernández, Xavier Franch, and Francisco Durán. "Towards green AI-based software systems: an architecture-centric approach (GAISSA)." In: *arXiv preprint arXiv:2307.09964* (2023).

[236] Vincenzo De Martino, Joel Castaño, Fabio Palomba, Xavier Franch, and Silverio Martínez-Fernández. "A Methodological Framework for LLM-Based Mining of Software Repositories." In: (July 2025). DOI: 10.6084/m9.figshare.29647169. URL: https://figshare

.com/articles/dataset/A_Methodological_Framework_for_LLM-Based_Mining_of_Software_Repositories/29647169.

[237] Vincenzo De Martino, Silverio Martínez-Fernández, and Fabio Palomba. "Large Language Models for Understanding and Eliciting Green Architectural Tactics in ML-enabled Systems." In: (Jan. 2025). DOI: 10.6084/m9.figshare.28230107. URL: https://figshare.com/articles/dataset/Large_Language_Models_for_Understanding_and_Eliciting_Green_Architectural_Tactics_in_ML-enabled_Systems/28230107.

[238] Vincenzo De Martino and Fabio Palomba. "Classification and Challenges of Non-Functional Requirements in ML-Enabled Systems: A Systematic Literature Review." In: (Jan. 2025). DOI: https://doi.org/10.6084/m9.figshare.22815770. URL: https://figshare.com/articles/dataset/Classification_Challenges_and_Automated_Approaches_to_Handle_Non-Functional_Requirements_in_ML-Enabled_Systems_A_Systematic_Literature_Review/22815770.

[239] Vincenzo De Martino, Gilberto Recupito, Giammaria Giordano, Filomena Ferrucci, Dario Di Nucci, and Fabio Palomba. "Into the ML-universe: An Improved Classification and Characterization of Machine-Learning Projects." In: (June 2024). DOI: 10.6084/m9.figshare.25974886.v9. URL: https://figshare.com/articles/dataset/Replication_Package_of_b_Into_the_ML-universe_An_Improved_Classification_and_Characterization_of_Machine-Learning_Projects_b_/25974886.

[240] Antonio Mastropaolo, Luca Pascarella, Emanuela Guglielmi, Matteo Ciniselli, Simone Scalabrino, Rocco Oliveto, and Gabriele Bavota. "On the robustness of code generation techniques: An empirical study on github copilot." In: *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE. 2023, pp. 2149–2160.

[241] B Bafandeh Mayvan, Abbas Rasoolzadegan, and Z Ghavidel Yazdi. "The state of the art on design patterns: A systematic mapping of the literature." In: *Journal of Systems and Software* 125 (2017), pp. 93–118.

[242] Sean McGuire, Erin Schultz, Bimpe Ayoola, and Paul Ralph. "Sustainability is stratified: Toward a better theory of sustainable software engineering." In: *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE. 2023, pp. 1996–2008.

[243] Patrick E McKnight and Julius Najab. "Mann-Whitney U Test." In: *The Corsini encyclopedia of psychology* (2010), pp. 1–1.

[244] Collin McMillan, Mario Linares-Vasquez, Denys Poshyvanyk, and Mark Grechanik. "Categorizing software applications for maintenance." In: *2011 27th ieee international conference on software maintenance (icsm)*. IEEE. 2011, pp. 343–352.

[245] Marcel Meesters, Petra Heck, and Alexander Serebrenik. "What is an AI engineer? An empirical analysis of job ads in The Netherlands." In: *Proceedings of the 1st International Conference on AI Engineering: Software Engineering for AI*. 2022, pp. 136–144.

[246] Claire Cain Miller. "Can an algorithm hire better than a human." In: *The New York Times* 25 (2015).

[247] Joseph Misiti. *Awesome Machine Learning: A curated list of awesome Machine Learning frameworks, libraries and software*. https://github.com/josephmisiti/awesome-machine-learning. Accessed: 2024-08-31. 2015.

[248] Sparsh Mittal. "A survey on optimized implementation of deep learning models on the nvidia jetson platform." In: *Journal of Systems Architecture* 97 (2019), pp. 428–442.

[249] Erica Mourão, João Felipe Pimentel, Leonardo Murta, Marcos Kalinowski, Emilia Mendes, and Claes Wohlin. "On the performance of hybrid search strategies for systematic literature reviews in software engineering." In: *Information and software technology* 123 (2020), p. 106294.

[250] Nuthan Munaiah, Steven Kroh, Craig Cabrey, and Meiyappan Nagappan. "Curating github for engineered software projects." In: *Empirical Software Engineering* 22 (2017), pp. 3219–3253.

[251] Nadia Nahar, Haoran Zhang, Grace Lewis, Shurui Zhou, and Christian Kästner. "The Product Beyond the Model–An Empirical Study of Repositories of Open-Source ML Products." In: *arXiv preprint arXiv:2308.04328* (2023).

[252] Daye Nam, Andrew Macvean, Vincent Hellendoorn, Bogdan Vasilescu, and Brad Myers. "Using an llm to help with code understanding." In: *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 2024, pp. 1–13.

[253] Harish Karthi Natarajan, Danielly de Paula, Christian Dremel, and Falk Uebernickel. "A theoretical review on AI affordances for sustainability." In: (2022).

[254] Peter Bjorn Nemenyi. *Distribution-free multiple comparisons*. Princeton University, 1963.

[255] MIT News. *Shrinking deep learning's carbon footprint*. https://news.mit.edu/2020/shrinking-deep-learning-carbon-footprint-0807. Accessed: 2023-11-26. 2023.

[256] Giang Nguyen, Stefan Dlugolinsky, Martin Bobák, Viet Tran, Álvaro López García, Ignacio Heredia, Peter Malík, and Ladislav Hluchỳ. "Machine learning and deep learning frameworks and libraries for large-scale data mining: a survey." In: *Artificial Intelligence Review* 52 (2019), pp. 77–124.

[257] Phuong T Nguyen, Claudio Di Sipio, Juri Di Rocco, Massimiliano Di Penta, and Davide Di Ruscio. "Adversarial attacks to api recommender systems: Time to wake up and smell the coffee?" In: *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE. 2021, pp. 253–265.

[258] Tien Nguyen, Waris Gill, and Muhammad Ali Gulzar. "Are the Majority of Public Computational Notebooks Pathologically Non-Executable?" In: *arXiv preprint arXiv:2502.04184* (2025).

[259] David Nigenda, Zohar Karnin, Muhammad Bilal Zafar, Raghu Ramesha, Alan Tan, Michele Donini, and Krishnaram Kenthapadi. "Amazon SageMaker Model Monitor: A System for Real-Time Insights into Deployed Machine Learning Models." In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. KDD '22. New York, NY, USA: Association for Computing Machinery, 2022, 3671–3681. ISBN: 9781450393850. DOI: 10.1145/3534678.3539145. URL: https://doi.org/10.1145/3534678.3539145.

[260] Parmy Olson. "The algorithm that beats your bank manager." In: *CNN Money March* 15 (2011).

[261] Joanna Isabelle Olszewska. "Snakes in Trees: An Explainable Artificial Intelligence Approach for Automatic Object Detection and Recognition." In: *ICAART (3)*. 2022, pp. 996–1002.

[262] Abbas Ourmazd. "Science in the age of machine learning." In: *Nature Reviews Physics* 2.7 (2020), pp. 342–343.

[263] Tiago P Pagano, Rafael B Loureiro, Fernanda VN Lisboa, Rodrigo M Peixoto, Guilherme AS Guimarães, Gustavo OR Cruz, Maira M Araujo, Lucas L Santos, Marco AS Cruz, Ewerton LS Oliveira, et al. "Bias and unfairness in machine learning models: a systematic review on datasets, tools, fairness metrics, and identification and mitigation methods." In: *Big data and cognitive computing* 7.1 (2023), p. 15.

[264] Lawrence A Palinkas, Sarah M Horwitz, Carla A Green, Jennifer P Wisdom, Naihua Duan, and Kimberly Hoagwood. "Purposeful sampling for qualitative data collection and analysis in mixed method implementation research." In: *Administration and policy in mental health and mental health services research* 42 (2015), pp. 533–544.

[265] Fabio Palomba, Damian Andrew Tamburri, Francesca Arcelli Fontana, Rocco Oliveto, Andy Zaidman, and Alexander Serebrenik. "Beyond technical aspects: How do community smells influence

the intensity of code smells?" In: *IEEE transactions on software engineering* 47.1 (2018), pp. 108–129.

[266] Rangeet Pan and Hridesh Rajan. "On Decomposing a Deep Neural Network into Modules." In: *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2020. New York, NY, USA: Association for Computing Machinery, 2020, 889–900. ISBN: 9781450370431. DOI: 10.1145/3368089.3409668. URL: https://doi.org/10.1145/3368089.3409668.

[267] Rangeet Pan and Hridesh Rajan. "Decomposing Convolutional Neural Networks into Reusable and Replaceable Modules." In: *Proceedings of the 44th International Conference on Software Engineering*. ICSE '22. Association for Computing Machinery, 2022, 524–535. ISBN: 9781450392211. DOI: 10.1145/3510003.3510051. URL: https://doi.org/10.1145/3510003.3510051.

[268] Aastha Pant, Rashina Hoda, Simone V Spiegler, Chakkrit Tantithamthavorn, and Burak Turhan. "Ethics in the Age of AI: An Analysis of AI Practitioners' Awareness and Challenges." In: *ACM Transactions on Software Engineering and Methodology* (2023).

[269] Maria Pawelec. "Deepfakes and democracy (theory): How synthetic audio-visual media for disinformation and hate speech threaten core democratic functions." In: *Digital society* 1.2 (2022), p. 19.

[270] Kewen Peng, Joymallya Chakraborty, and Tim Menzies. "FairMask: Better fairness via model-based rebalancing of protected attributes." In: *IEEE Transactions on Software Engineering* 49.4 (2022), pp. 2426–2439.

[271] Kewen Peng, Joymallya Chakraborty, and Tim Menzies. "FairMask: Better Fairness via Model-Based Rebalancing of Protected Attributes." In: *IEEE Transactions on Software Engineering* 49.4 (2023), pp. 2426–2439. DOI: 10.1109/TSE.2022.3220713.

[272] Zi Peng, Jinqiu Yang, Tse-Hsun Chen, and Lei Ma. "A first look at the integration of machine learning models in complex autonomous driving systems: a case study on apollo." In: *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2020, pp. 1240–1250.

[273] Anjana Perera, Aldeida Aleti, Chakkrit Tantithamthavorn, Jirayus Jiarpakdee, Burak Turhan, Lisa Kuhn, and Katie Walker. "Search-based fairness testing for regression-based machine learning systems." In: *Empirical Software Engineering* 27.3 (2022), p. 79.

[274] Dorin Pomian, Abhiram Bellur, Malinda Dilhara, Zarina Kurbatova, Egor Bogomolov, Timofey Bryksin, and Danny Dig. "Next-generation refactoring: Combining llm insights and ide capabilities for extract method." In: *2024 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE. 2024, pp. 275–287.

[275] Ali Ebrahimi Pourasad and Walid Maalej. "Does GenAI Make Usability Testing Obsolete?" In: *arXiv preprint arXiv:2411.00634* (2024).

[276] David MW Powers. "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation." In: *arXiv preprint arXiv:2010.16061* (2020).

[277] Andria Procopiou and Andriani Piki. "The 12th Player: Explainable Artificial Intelligence (XAI) in Football: Conceptualisation, Applications, Challenges and Future Directions." In: *Proceedings of the 11th International Conference on Sport Sciences Research and Technology Support*. Vol. 1. Science and Technology Publications, Lda. 2023, pp. 213–220.

[278] Ben Purvis, Yong Mao, and Darren Robinson. "Three pillars of sustainability: in search of conceptual origins." In: *Sustainability science* 14 (2019), pp. 681–695.

[279] Binhang Qi, Hailong Sun, Xiang Gao, Hongyu Zhang, Zhaotian Li, and Xudong Liu. "Reusing deep neural network models through model re-engineering." In: *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE. 2023, pp. 983–994.

[280] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. "Language models are unsupervised multitask learners." In: *OpenAI blog* 1.8 (2019), p. 9.

[281] Paul Ralph, Nauman bin Ali, Sebastian Baltes, Domenico Bianculli, Jessica Diaz, Yvonne Dittrich, Neil Ernst, Michael Felderer, Robert Feldt, Antonio Filieri, et al. "Empirical standards for software engineering research." In: *arXiv preprint arXiv:2010.03525* (2020).

[282] Sebastian Raschka, Joshua Patterson, and Corey Nolet. "Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence." In: *Information* 11.4 (2020), p. 193.

[283] Daniele Ravì, Charence Wong, Fani Deligianni, Melissa Berthelot, Javier Andreu-Perez, Benny Lo, and Guang-Zhong Yang. "Deep learning for health informatics." In: *IEEE journal of biomedical and health informatics* 21.1 (2016), pp. 4–21.

[284] Rozilawati Razali and Humairath Abu Bakar. "Combining Phenomenology and Grounded Theory in Software Engineering: An Experience." In: *ECRM 2023 22nd European Conference on Research Methods in Business and Management*. Academic Conferences and publishing limited. 2023.

[285] Jörg Rech and Klaus-Dieter Althoff. "Artificial intelligence and software engineering: Status and future trends." In: *KI* 18.3 (2004), pp. 5–11.

[286] Álvaro Domingo Reguero, Silverio Martínez-Fernández, and Roberto Verdecchia. "Energy-efficient neural network training through runtime layer freezing, model quantization, and early stopping." In: *Computer Standards & Interfaces* (2024), p. 103906.

[287] Álvaro Domingo Reguero, Silverio Martínez-Fernández, and Roberto Verdecchia. "Energy-efficient neural network training through runtime layer freezing, model quantization, and early stopping." In: *Computer Standards & Interfaces* 92 (2025), p. 103906.

[288] Brittany Reid, Markus Wagner, Marcelo d'Amorim, and Christoph Treude. "Software Engineering User Study Recruitment on Prolific: An Experience Report." In: *arXiv preprint arXiv:2201.05348* (2022).

[289] Xiaoning Ren, Yun Lin, Yinxing Xue, Ruofan Liu, Jun Sun, Zhiyong Feng, and Jin Song Dong. "DeepArc: Modularizing neural networks for the model maintenance." In: *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE. 2023, pp. 1008–1019.

[290] Harvard Business Review. *AI Is Not Just Getting Better, it's Becoming More Pervasive*. https://hbr.org/sponsored/2019/02/ai-is-not-just-getting-better-its-becoming-more-pervasive. Accessed: 2023-03-05. 2023.

[291] Santiago del Rey, Silverio Martínez-Fernández, Luís Cruz, and Xavier Franch. "Do DL models and training environments have an impact on energy consumption?" In: *arXiv preprint arXiv:2307.05520* (2023).

[292] Vincenzo Riccio, Gunel Jahangirova, Andrea Stocco, Nargiz Humbatova, Michael Weiss, and Paolo Tonella. "Testing machine learning based systems: a systematic mapping." In: *Empirical Software Engineering* 25 (2020), pp. 5193–5254.

[293] Matteo Rizzo, Alberto Veneri, Andrea Albarelli, Claudio Lucchese, Marco Nobile, and Cristina Conati. "A Theoretical Framework for AI Models Explainability with Application in Biomedicine." In: *2023 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. IEEE. 2023, pp. 1–9.

[294] Per Runeson and Martin Höst. "Guidelines for conducting and reporting case study research in software engineering." In: *Empirical Software Engineering* 14.2 (2009), pp. 131–164.

[295] Dhia Elhaq Rzig, Foyzul Hassan, Chetan Bansal, and Nachiappan Nagappan. "Characterizing the Usage of CI Tools in ML Projects." In: *Proceedings of the 16th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. 2022, pp. 69–79.

[296] Shirin Salehi and Anke Schmeink. "Data-centric green artificial intelligence: A survey." In: *IEEE Transactions on Artificial Intelligence* (2023).

[297] June Sallou, Thomas Durieux, and Annibale Panichella. "Breaking the silence: the threats of using llms in software engineering." In: *Proceedings of the 2024 ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results*. 2024, pp. 102–106.

[298] Theresia Ratih Dewi Saputri and Seok-Won Lee. "Software sustainability requirements: A unified method for improving requirements process for software development." In: *2021 IEEE 29th International Requirements Engineering Conference (RE)*. IEEE. 2021, pp. 506–507.

[299] Federica Sarro. "Search-Based Software Engineering in the Era of Modern Software Systems." In: *2023 IEEE 31st International Requirements Engineering Conference (RE)*. 2023, pp. 3–5. DOI: 10.1109/RE57278.2023.00010.

[300] Benjamin Saunders, Julius Sim, Tom Kingstone, Shula Baker, Jackie Waterfield, Bernadette Bartlam, Heather Burroughs, and Clare Jinks. "Saturation in qualitative research: exploring its conceptualization and operationalization." In: *Quality & quantity* 52 (2018), pp. 1893–1907.

[301] Mohammed Sayagh, Noureddine Kerzazi, Bram Adams, and Fabio Petrillo. "Software configuration engineering in practice interviews, survey, and systematic literature review." In: *IEEE Transactions on Software Engineering* 46.6 (2018), pp. 646–673.

[302] Mark A Schmuckler. "What is ecological validity? A dimensional analysis." In: *Infancy* 2.4 (2001), pp. 419–436.

[303] Mathias Schneider, Ruben Prokscha, Seifeddine Saadani, and Alfred H¨oß. “ECBA-MLI: Edge computing benchmark architecture for machine learning inference.” In: *2022 IEEE International Conference on Edge Computing and Communications (EDGE)*. IEEE. 2022, pp. 23–32.

[304] Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. “Green ai.” In: *Communications of the ACM* 63.12 (2020), pp. 54–63.

[305] SeldonIO. *SeldonIO/seldon-core*. https://github.com/SeldonIO/seldon-core/blob/5509dc09e494fa5cf0551a96294321383e588087/examples/kubeflow/pipeline/pipeline_steps/clean_text/Transformer.py#L4. [Accessed 10-01-2025].

[306] Md Shafikuzzaman, Md Rakibul Islam, Alex C Rolli, Sharmin Akhter, and Naeem Seliya. “An Empirical Evaluation of the Zero-Shot, Few-Shot, and Traditional Fine-Tuning Based Pretrained Language Models for Sentiment Analysis in Software Engineering.” In: *IEEE Access* (2024).

[307] Shriram Shanbhag, Sridhar Chimalakonda, Vibhu Saujanya Sharma, and Vikrant Kaulgud. “Towards a catalog of energy patterns in deep learning development.” In: *Proceedings of the 26th International Conference on Evaluation and Assessment in Software Engineering*. 2022, pp. 150–159.

[308] Samuel Sanford Shapiro and Martin B Wilk. “An analysis of variance test for normality (complete samples).” In: *Biometrika* 52.3/4 (1965), pp. 591–611.

[309] Mohammad Sadegh Sheikhaei, Yuan Tian, Shaowei Wang, and Bowen Xu. “An empirical study on the effectiveness of large language models for SATD identification and classification.” In: *Empirical Software Engineering* 29.6 (2024), p. 159.

[310] Reza Shokri and Vitaly Shmatikov. “Privacy-Preserving Deep Learning.” In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. CCS ’15. New York, NY, USA: Association for Computing Machinery, 2015, 1310–1321. ISBN:

9781450338325. DOI: 10.1145/2810103.2813687. URL: https://doi.org/10.1145/2810103.2813687.

[311] Connor Shorten and Taghi M Khoshgoftaar. "A survey on image data augmentation for deep learning." In: *Journal of big data* 6.1 (2019), pp. 1–48.

[312] Yap Yan Siang, Mohd Ridzuan Ahamd, and Mastura Shafinaz Zainal Abidin. "Anomaly detection based on tiny machine learning: A review." In: *Open International Journal of Informatics* 9.Special Issue 2 (2021), pp. 67–78.

[313] Miltiadis Siavvas, Dimitrios Tsoukalas, Marija Jankovic, Dionysios Kehagias, and Dimitrios Tzovaras. "Technical debt as an indicator of software security risk: a machine learning approach for software development enterprises." In: *Enterprise Information Systems* 16.5 (2022), p. 1824017.

[314] *Significant EEOC Race/Color Cases(Covering Private and Federal Sectors) — eeoc.gov*. https://www.eeoc.gov/initiatives/e-race/significant-eeoc-racecolor-casescovering-private-and-federal-sectors#intersectional.

[315] Samarth Sikand, Vibhu Saujanya Sharma, Vikrant Kaulgud, and Sanjay Podder. "Green AI Quotient: Assessing Greenness of AI-based software and the way forward." In: *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE. 2023, pp. 1828–1833.

[316] Luciana Lourdes Silva, Janio Rosa da Silva, João Eduardo Montandon, Marcus Andrade, and Marco Tulio Valente. "Detecting code smells using chatgpt: Initial insights." In: *Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. 2024, pp. 400–406.

[317] Chiappa Silvia, Jiang Ray, Stepleton Tom, Pacchiano Aldo, Jiang Heinrich, and Aslanides John. "A general approach to fairness with optimal transport." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04. 2020, pp. 3633–3640.

[318] S.E. Sim, S. Easterbrook, and R.C. Holt. "Using benchmarking to advance research: a challenge to software engineering." In: *25th International Conference on Software Engineering, 2003. Proceedings.* 2003, pp. 74–83. DOI: 10.1109/ICSE.2003.1201189.

[319] Susan Elliott Sim, Steve Easterbrook, and Richard C Holt. "Using benchmarking to advance research: A challenge to software engineering." In: *25th International Conference on Software Engineering, 2003. Proceedings.* IEEE. 2003, pp. 74–83.

[320] Leslie N Smith. "Cyclical learning rates for training neural networks." In: *2017 IEEE winter conference on applications of computer vision (WACV).* IEEE. 2017, pp. 464–472.

[321] Jiayang Song, Deyun Lyu, Zhenya Zhang, Zhijie Wang, Tianyi Zhang, and Lei Ma. "When cyber-physical systems meet AI: A benchmark, an evaluation, and a way forward." In: *Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice.* 2022, pp. 343–352.

[322] Peter Sprent and Nigel C Smeeton. *Applied nonparametric statistical methods.* CRC press, 2007.

[323] Klaas-Jan Stol and Brian Fitzgerald. "Grounded theory in software engineering research: a critical review and guidelines." In: *ACM Computing Surveys (CSUR)* 50.4 (2018), pp. 1–37.

[324] Anselm Strauss and Juliet M Corbin. *Grounded theory in practice.* Sage, 1997.

[325] Anselm Strauss and Juliet Corbin. *Basics of qualitative research.* Sage publications, 1990.

[326] Bing Sun, Jun Sun, Long H Pham, and Jie Shi. "Causality-based neural network repair." In: *Proceedings of the 44th International Conference on Software Engineering.* 2022, pp. 338–349.

[327] Nuttanai Suwonchoochit and Twittie Senivongse. "Classification of Database Technology Problems on Stack Overflow." In: *2021 IEEE/ACIS 19th International Conference on Software Engineering Research, Management and Applications (SERA)*. IEEE. 2021, pp. 21–26.

[328] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. *Intriguing properties of neural networks*. 2013. DOI: 10.48550/ARXIV.1312.6199. URL: https://arxiv.org/abs/1312.6199.

[329] Abdulaziz Tabbakh, Lisan Al Amin, Mahbubul Islam, GM Iqbal Mahmud, Imranul Kabir Chowdhury, and Md Saddam Hossain Mukta. "Towards sustainable AI: a comprehensive framework for Green AI." In: *Discover Sustainability* 5.1 (2024), p. 408.

[330] Damian A Tamburri. "Sustainable mlops: Trends and challenges." In: *2020 22nd international symposium on symbolic and numeric algorithms for scientific computing (SYNASC)*. IEEE. 2020, pp. 17–23.

[331] Guanhong Tao, Weisong Sun, Tingxu Han, Chunrong Fang, and Xiangyu Zhang. "RULER: Discriminative and Iterative Adversarial Training for Deep Neural Network Fairness." In: *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2022. New York, NY, USA: Association for Computing Machinery, 2022, 1173–1184. ISBN: 9781450394130. DOI: 10.1145/3540250.3549169. URL: https://doi.org/10.1145/3540250.3549169.

[332] Steven K Thompson. *Sampling*. Vol. 755. John Wiley & Sons, 2012.

[333] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. "DeepTest: Automated Testing of Deep-Neural-Network-Driven Autonomous Cars." In: *Proceedings of the 40th International Conference on Software Engineering*. ICSE '18. New York, NY, USA: Association for Computing Machinery, 2018, 303–314. ISBN: 9781450356381. DOI:

10.1145/3180155.3180220. URL: https://doi.org/10.1145/3180155.3180220.

[334] Walter F Tichy. "Where's the science in software engineering? Ubiquity Symposium: The science in computer science." In: *Ubiquity* 2014.March (2014), pp. 1–6.

[335] Saeid Tizpaz-Niari, Ashish Kumar, Gang Tan, and Ashutosh Trivedi. "Fairness-Aware Configuration of Machine Learning Libraries." In: *Proceedings of the 44th International Conference on Software Engineering*. ICSE '22. Association for Computing Machinery, 2022, 909–920. ISBN: 9781450392211. DOI: 10.1145/3510003.3510202. URL: https://doi.org/10.1145/3510003.3510202.

[336] Tanja Tornede, Alexander Tornede, Jonas Hanselle, Felix Mohr, Marcel Wever, and Eyke Hüllermeier. "Towards green automated machine learning: Status quo and future directions." In: *Journal of Artificial Intelligence Research* 77 (2023), pp. 427–457.

[337] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. "Llama: Open and efficient foundation language models." In: *arXiv preprint arXiv:2302.13971* (2023).

[338] Florian Tramèr, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, Jean-Pierre Hubaux, Mathias Humbert, Ari Juels, and Huang Lin. "FairTest: Discovering Unwarranted Associations in Data-Driven Applications." In: *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*. 2017, pp. 401–416. DOI: 10.1109/EuroSP.2017.29.

[339] Bianca Trinkenreich, Fabio Calefato, Geir Hanssen, Kelly Blincoe, Marcos Kalinowski, Mauro Pezzé, Paolo Tell, and Margaret-Anne Storey. "Get on the Train or be Left on the Station: Using LLMs for Software Engineering Research." In: (2025).

[340] Hong-Linh Truong and Tri-Minh Nguyen. "QoA4ML - A Framework for Supporting Contracts in Machine Learning Services." In: *2021 IEEE International Conference on Web Services (ICWS)*. 2021, pp. 465–475. DOI: 10.1109/ICWS53863.2021.00066.

[341] Michele Tufano, Fabio Palomba, Gabriele Bavota, Rocco Oliveto, Massimiliano Di Penta, Andrea De Lucia, and Denys Poshyvanyk. "When and why your code starts to smell bad." In: *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. Vol. 1. IEEE. 2015, pp. 403–414.

[342] Alessandro Tundo, Marco Mobilio, Shashikant Ilager, Ivona Brandić, Ezio Bartocci, and Leonardo Mariani. "An energy-aware approach to design self-adaptive ai-based applications on the edge." In: *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE. 2023, pp. 281–293.

[343] Sebastian Uchitel, Marsha Chechik, Massimiliano Di Penta, Bram Adams, Nazareno Aguirre, Gabriele Bavota, Domenico Bianculli, Kelly Blincoe, Ana Cavalcanti, Yvonne Dittrich, et al. "Scoping Software Engineering for AI: The TSE Perspective." In: *IEEE Transactions on Software Engineering* 50.11 (2024), pp. 2709–2711.

[344] Gias Uddin and Martin P Robillard. "How API documentation fails." In: *Ieee software* 32.4 (2015), pp. 68–75.

[345] Sakshi Udeshi, Pryanshu Arora, and Sudipta Chattopadhyay. "Automated Directed Fairness Testing." In: *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. ASE '18. New York, NY, USA: Association for Computing Machinery, 2018, 98–108. ISBN: 9781450359375. DOI: 10.1145/3238147.3238165. URL: https://doi.org/10.1145/3238147.3238165.

[346] Secil Ugurel, Robert Krovetz, and C Lee Giles. "What's the code? automatic classification of source code archives." In: *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2002, pp. 632–638.

[347] Aimee Van Wynsberghe. “Sustainable AI: AI for sustainability and the sustainability of AI.” In: *AI and Ethics* 1.3 (2021), pp. 213–218.

[348] Colin C Venters, Rafael Capilla, Stefanie Betz, Birgit Penzenstadler, Tom Crick, Steve Crouch, Elisa Yumi Nakagawa, Christoph Becker, and Carlos Carrillo. “Software sustainability: Research and practice from a software architecture viewpoint.” In: *Journal of Systems and Software* 138 (2018), pp. 174–188.

[349] Roberto Verdecchia, Luís Cruz, June Sallou, Michelle Lin, James Wickenden, and Estelle Hotellier. “Data-centric green ai an exploratory empirical study.” In: *2022 international conference on ICT for sustainability (ICT4S)*. IEEE. 2022, pp. 35–45.

[350] Roberto Verdecchia, June Sallou, and Luís Cruz. “A systematic review of Green AI.” In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* (2023), e1507.

[351] Hugo Villamizar, Tatiana Escovedo, and Marcos Kalinowski. “Requirements engineering for machine learning: A systematic mapping study.” In: *2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE. 2021, pp. 29–36.

[352] Ricardo Vinuesa, Hossein Azizpour, Iolanda Leite, Madeline Balaam, Virginia Dignum, Sami Domisch, Anna Felländer, Simone Daniela Langhans, Max Tegmark, and Francesco Fuso Nerini. “The role of artificial intelligence in achieving the Sustainable Development Goals.” In: *Nature communications* 11.1 (2020), pp. 1–10.

[353] Gianmario Voria, Giulia Sellitto, Carmine Ferrara, Francesco Abate, Andrea De Lucia, Filomena Ferrucci, Gemma Catolino, and Fabio Palomba. “Fairness-aware practices from developers’ perspective: A survey.” In: *Information and Software Technology* 182 (2025), p. 107710.

[354] David Wagner and Paolo Soto. “Mimicry Attacks on Host-Based Intrusion Detection Systems.” In: *Proceedings of the 9th ACM Conference on Computer and Communications Security*. CCS ’02.

New York, NY, USA: Association for Computing Machinery, 2002, 255–264. ISBN: 1581136129. DOI: 10.1145/586110.586145. URL: https://doi.org/10.1145/586110.586145.

[355] Stefan Wagner, Marvin Muñoz Barón, Davide Falessi, and Sebastian Baltes. "Towards Evaluation Guidelines for Empirical Studies involving LLMs." In: *arXiv preprint arXiv:2411.07668* (2024).

[356] Chengcheng Wan, Shicheng Liu, Henry Hoffmann, Michael Maire, and Shan Lu. "Are machine learning cloud apis used correctly?" In: *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE. 2021, pp. 125–137.

[357] Yuxuan Wan, Wenxuan Wang, Pinjia He, Jiazhen Gu, Haonan Bai, and Michael R Lyu. "Biasasker: Measuring the bias in conversational ai system." In: *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2023, pp. 515–527.

[358] Ruiqi Wang, Jiyu Guo, Cuiyun Gao, Guodong Fan, Chun Yong Chong, and Xin Xia. "Can llms replace human evaluators? an empirical study of llm-as-a-judge in software engineering." In: *Proceedings of the ACM on Software Engineering* 2.ISSTA (2025), pp. 1955–1977.

[359] Weixuan Wang, Choon Meng Lee, Jianfeng Liu, Talha Colakoglu, and Wei Peng. "An empirical study of cyclical learning rate on neural machine translation." In: *Natural Language Engineering* 29.2 (2023), pp. 316–336.

[360] Xianmin Wang, Jing Li, Xiaohui Kuang, Yu-an Tan, and Jin Li. "The security of machine learning in an adversarial setting: A survey." In: *Journal of Parallel and Distributed Computing* 130 (2019), pp. 12–23.

[361] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. "Chain-of-thought prompting elicits reasoning in large language models." In: *Advances*

in neural information processing systems 35 (2022), pp. 24824–24837.

[362] Xiaokai Wei, Sujan Kumar Gonugondla, Shiqi Wang, Wasi Ahmad, Baishakhi Ray, Haifeng Qian, Xiaopeng Li, Varun Kumar, Zijian Wang, Yuchen Tian, et al. "Towards greener yet powerful code generation via quantization: An empirical study." In: *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2023, pp. 224–236.

[363] Michael Weiss and Paolo Tonella. "Fail-safe execution of deep learning based systems through uncertainty monitoring." In: *2021 14th IEEE conference on software testing, verification and validation (ICST)*. IEEE. 2021, pp. 24–35.

[364] Ratnadira Widyasari, Zhou Yang, Ferdian Thung, Sheng Qin Sim, Fiona Wee, Camellia Lok, Jack Phan, Haodi Qi, Constance Tan, Qijin Tay, et al. "NICHE: A Curated Dataset of Engineered Machine Learning Projects in Python." In: *arXiv preprint arXiv:2303.06286* (2023).

[365] Claes Wohlin. "Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering." In: *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*. EASE '14. New York, NY, USA: Association for Computing Machinery, 2014. ISBN: 9781450324762. DOI: 10.1145/2601248.2601268. URL: https://doi.org/10.1145/2601248.2601268.

[366] Claes Wohlin. "Second-generation systematic literature studies using snowballing." In: *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*. 2016, pp. 1–6.

[367] Claes Wohlin, Emilia Mendes, Katia Romero Felizardo, and Marcos Kalinowski. "Guidelines for the search strategy to update system-

atic literature reviews in software engineering." In: *Information and software technology* 127 (2020), p. 106366.

[368] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[369] Cong Wu, Jing Chen, Ziwei Wang, Ruichao Liang, and Ruiying Du. "Semantic Sleuth: Identifying Ponzi Contracts via Large Language Models." In: *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*. 2024, pp. 582–593.

[370] Yisong Xiao, Aishan Liu, Tianlin Li, and Xianglong Liu. "Latent imitator: Generating natural individual discriminatory instances for black-box fairness testing." In: *Proceedings of the 32nd ACM SIGSOFT international symposium on software testing and analysis*. 2023, pp. 829–841.

[371] Xiaofei Xie, Lei Ma, Felix Juefei-Xu, Minhui Xue, Hongxu Chen, Yang Liu, Jianjun Zhao, Bo Li, Jianxiong Yin, and Simon See. "Deephunter: a coverage-guided fuzz testing framework for deep neural networks." In: *Proceedings of the 28th ACM SIGSOFT international symposium on software testing and analysis*. 2019, pp. 146–157.

[372] Mengwei Xu, Jiawei Liu, Yuanqiang Liu, Felix Xiaozhu Lin, Yunxin Liu, and Xuanzhe Liu. "A First Look at Deep Learning Apps on Smartphones." In: *The World Wide Web Conference*. WWW '19. New York, NY, USA: Association for Computing Machinery, 2019, 2125–2136. ISBN: 9781450366748. DOI: 10.1145/3308558.3313591. URL: https://doi.org/10.1145/3308558.3313591.

[373] Runhua Xu, Nathalie Baracaldo, and James Joshi. "Privacy-preserving machine learning: Methods, challenges and directions." In: *arXiv preprint arXiv:2108.04417* (2021).

[374] Weiwei Xu, Kai Gao, Hao He, and Minghui Zhou. "LiCoEval: Evaluating LLMs on License Compliance in Code Generation." In: *arXiv preprint arXiv:2408.02487* (2024).

[375] Yinlena Xu, Silverio Martínez-Fernández, Matias Martinez, and Xavier Franch. "Energy Efficiency of Training Neural Network Architectures: An Empirical Study." In: *arXiv preprint arXiv:2302.00967* (2023).

[376] Shen Yan, Hsien-te Kao, and Emilio Ferrara. "Fair Class Balancing: Enhancing Model Fairness without Observing Sensitive Attributes." In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. CIKM '20. New York, NY, USA: Association for Computing Machinery, 2020, 1715–1724. ISBN: 9781450368599. DOI: 10.1145/3340531.3411980. URL: https://doi.org/10.1145/3340531.3411980.

[377] Tim Yarally, Luıs Cruz, Daniel Feitosa, June Sallou, and Arie Van Deursen. "Uncovering Energy-Efficient Practices in Deep Learning Training: Preliminary Steps Towards Green AI." In: *2023 IEEE/ACM 2nd International Conference on AI Engineering–Software Engineering for AI (CAIN)*. IEEE. 2023, pp. 25–36.

[378] Yining Yin, Yang Feng, Shihao Weng, Zixi Liu, Yuan Yao, Yichi Zhang, Zhihong Zhao, and Zhenyu Chen. "Dynamic Data Fault Localization for Deep Neural Networks." In: *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2023, pp. 1345–1357.

[379] Bing Yu, Hua Qi, Qing Guo, Felix Juefei-Xu, Xiaofei Xie, Lei Ma, and Jianjun Zhao. "DeepRepair: Style-Guided Repairing for Deep Neural Networks in the Real-World Operational Environment." In: *IEEE Transactions on Reliability* 71.4 (2022), pp. 1401–1416. DOI: 10.1109/TR.2021.3096332.

[380] Daochen Zha, Zaid Pervaiz Bhat, Kwei-Herng Lai, Fan Yang, and Xia Hu. "Data-centric ai: Perspectives and challenges." In: *Proceedings of the 2023 SIAM International Conference on Data Mining (SDM)*. SIAM. 2023, pp. 945–948.

[381] Biao Zhang, Barry Haddow, and Alexandra Birch. "Prompting large language model for machine translation: A case study." In: *Inter-*

national Conference on Machine Learning. PMLR. 2023, pp. 41092–41110.

[382] Fuyuan Zhang, Sankalan Pal Chowdhury, and Maria Christakis. "Deepsearch: A simple and effective blackbox attack for deep neural networks." In: *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2020, pp. 800–812.

[383] Fuyuan Zhang, Xinwen Hu, Lei Ma, and Jianjun Zhao. "DeepRover: A Query-Efficient Blackbox Attack for Deep Neural Networks." In: *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2023, pp. 1384–1394.

[384] Hao Zhang and W.K. Chan. "Apricot: A Weight-Adaptation Approach to Fixing Deep Learning Models." In: *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 2019, pp. 376–387. DOI: 10.1109/ASE.2019.00043.

[385] Jie M Zhang and Mark Harman. "" Ignorance and Prejudice" in Software Fairness." In: *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE. 2021, pp. 1436–1447.

[386] Jie M. Zhang and Mark Harman. """Ignorance and Prejudice"" in Software Fairness." In: *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. 2021, pp. 1436–1447. DOI: 10.1109/ICSE43902.2021.00129.

[387] Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. "Why gradient clipping accelerates training: A theoretical justification for adaptivity." In: *arXiv preprint arXiv:1905.11881* (2019).

[388] Mengdi Zhang and Jun Sun. "Adaptive fairness improvement based on causality analysis." In: *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2022, pp. 6–17.

[389] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. "DeepRoad: GAN-Based Metamorphic Testing and Input Validation Framework for Autonomous Driving Systems." In: *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. ASE '18. New York, NY, USA: Association for Computing Machinery, 2018, 132–142. ISBN: 9781450359375. DOI: 10.1145/3238147.3238187. URL: https://doi.org/10.1145/3238147.3238187.

[390] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. "Deep learning based recommender system: A survey and new perspectives." In: *ACM computing surveys (CSUR)* 52.1 (2019), pp. 1–38.

[391] Tao Zhang, Tianqing Zhu, Kun Gao, Wanlei Zhou, and S Yu Philip. "Balancing learning model privacy, fairness, and accuracy with early stopping criteria." In: *IEEE Transactions on Neural Networks and Learning Systems* 34.9 (2021), pp. 5557–5569.

[392] Ting Zhang, Ivana Clairine Irsan, Ferdian Thung, and David Lo. "Revisiting Sentiment Analysis for Software Engineering in the Era of Large Language Models." In: *ACM Transactions on Software Engineering and Methodology* 34.3 (2025), pp. 1–30.

[393] Yichi Zhang, Zixi Liu, Yang Feng, and Baowen Xu. "Leveraging Large Language Model to Assist Detecting Rust Code Comment Inconsistency." In: *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*. 2024, pp. 356–366.

[394] Yingyi Zhang, Zan Wang, Jiajun Jiang, Hanmo You, and Junjie Chen. "Toward Improving the Robustness of Deep Learning Models via Model Transformation." In: *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. ASE '22. New York, NY, USA: Association for Computing Machinery, 2023. ISBN: 9781450394758. DOI: 10.1145/3551349.3556920. URL: https://doi.org/10.1145/3551349.3556920.

[395] Zhenya Zhang, Deyun Lyu, Paolo Arcaini, Lei Ma, Ichiro Hasuo, and Jianjun Zhao. "FalsifAI: Falsification of AI-enabled hybrid control

systems guided by time-aware coverage criteria." In: *IEEE Transactions on Software Engineering* 49.4 (2022), pp. 1842–1859.

[396] Ziqi Zhang, Yuanchun Li, Jindong Wang, Bingyan Liu, Ding Li, Yao Guo, Xiangqun Chen, and Yunxin Liu. "ReMoS: reducing defect inheritance in transfer learning via relevant model slicing." In: *Proceedings of the 44th International Conference on Software Engineering*. 2022, pp. 1856–1868.

[397] Yue Zhao, Hong Zhu, Kai Chen, and Shengzhi Zhang. "AI-Lancet: Locating Error-Inducing Neurons to Optimize Neural Networks." In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. CCS '21. New York, NY, USA: Association for Computing Machinery, 2021, 141–158. ISBN: 9781450384544. DOI: 10.1145/3460120.3484818. URL: https://doi.org/10.1145/3460120.3484818.

[398] Haibin Zheng, Zhiqing Chen, Tianyu Du, Xuhong Zhang, Yao Cheng, Shouling Ji, Jingyi Wang, Yue Yu, and Jinyin Chen. "Neuronfair: Interpretable white-box fairness testing through biased neuron identification." In: *Proceedings of the 44th International Conference on Software Engineering*. 2022, pp. 1519–1531.

[399] Zibin Zheng, Kaiwen Ning, Qingyuan Zhong, Jiachi Chen, Wenqing Chen, Lianghong Guo, Weicheng Wang, and Yanlin Wang. "Towards an understanding of large language models in software engineering tasks." In: *Empirical Software Engineering* 30.2 (2025), p. 50.

[400] Hao Zhong and Zhendong Su. "Detecting API documentation errors." In: *Proceedings of the 2013 ACM SIGPLAN international conference on Object oriented programming systems languages & applications*. 2013, pp. 803–816.

[401] Jianlong Zhou and Fang Chen. *Human and Machine Learning*. Springer, 2018.

[402] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. "Large language

models are human-level prompt engineers." In: *arXiv preprint arXiv:2211.01910* (2022).

[403] Jie Zhu, Leye Wang, and Xiao Han. "Safety and Performance, Why Not Both? Bi-Objective Optimized Model Compression toward AI Software Deployment." In: *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. ASE '22. New York, NY, USA: Association for Computing Machinery, 2023. ISBN: 9781450394758. DOI: 10.1145/3551349.3556906. URL: https://doi.org/10.1145/3551349.3556906.

[404] Deqing Zou, Yawei Zhu, Shouhuai Xu, Zhen Li, Hai Jin, and Hengkai Ye. "Interpreting Deep Learning-Based Vulnerability Detector Predictions Based on Heuristic Searching." In: *ACM Trans. Softw. Eng. Methodol.* 30.2 (2021). ISSN: 1049-331X. DOI: 10.1145/3429444. URL: https://doi.org/10.1145/3429444.

[405] *bitextor/bitextor*. May 21, 2024. URL: https://github.com/bitextor/bitextor.

[406] *cmasch/densenet*. Dec. 15, 2019. URL: https://github.com/cmasch/densenet.

[407] guotong1988. *guotong1988/BERT-GPU*. https://github.com/guotong1988/BERT-GPU/blob/b0d2a9147c198e3fe6da30027eaaf4f8c95e2ebb/run_pretraining_gpu.py#L4. [Accessed 10-01-2025].

[408] hasanirtiza. *hasanirtiza/Pedestron*. https://github.com/hasanirtiza/Pedestron/blob/a8ce101ca882417adf651479a74766effd87bff6/configs/rpn_x101_32x4d_fpn_1x.py#L4. [Accessed 10-01-2025].

[409] hasanirtiza. *hasanirtiza/Pedestron*. https://github.com/hasanirtiza/Pedestron/blob/a8ce101ca882417adf651479a74766effd87bff6/configs/lsfm/caltech/convmlpHR_mixup_width_refine_ecp_2x16.py#L4. [Accessed 10-01-2025].

[410] *imfunniee/pymine*. June 22, 2019. URL: https://github.com/imfunniee/pymine.

[411] *intuit/foremast-brain*. Feb. 3, 2023. URL: https://github.com/intuit/foremast-brain.

[412] microsoft. *microsoft/onnxruntime*. https://github.com/microsoft/onnxruntime/blob/e7d8596c7c78cd4df5f9ec1d15beb04b3ddf333c/onnxruntime/python/tools/transformers/models/t5/t5_decoder.py. [Accessed 10-01-2025].

[413] pyannote. *pyannote/pyannote-audio*. https://github.com/pyannote/pyannote-audio/blob/3c167f201fdbdd5391cc8da67f0f9eb25238e5a6/pyannote/audio/cli/train.py. [Accessed 10-01-2025].

[414] salu133445. *salu133445/musegan*. https://github.com/salu133445/musegan/blob/3c9a0dc71ab743b58f4e0620832f21dbe023408e/v2/training_data/store_to_sa.py#L4. [Accessed 10-01-2025].

[415] asteroid team. *asteroid-team/asteroid*. https://github.com/asteroid-team/asteroid/blob/154c52f6e15de9e42213b9997aa1a0ad8b0d453b/egs/musdb18/OpenUnmix/test_dataloader.py#L4. [Accessed 10-01-2025].

[416] asteroid team. *asteroid-team/asteroid*. https://github.com/asteroid-team/asteroid/blob/154c52f6e15de9e42213b9997aa1a0ad8b0d453b/egs/wham/MixIT/train.py#L4. [Accessed 10-01-2025].