

Università degli Studi di Salerno

Dipartimento di Informatica

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

Tesi di Dottorato / Ph.D. Thesis

Metadata-Driven Data Management and Analysis: From Profiling to Applications

Roberto STANZIONE

Supervisor: **Loredana GARUCCIO**

PhD Program Director: **Prof. Andrea DE LUCIA**

A.A 2024/2025

ABSTRACT

The digitalization of the public sector has led to the availability of large and heterogeneous collections of data, emphasizing the need for effective methods to extract, interpret, and exploit data-driven knowledge. To this end, Data Profiling techniques enable to automatically infer metadata that describe the structure, patterns, and relationships within datasets. Among these, data dependencies represent an increasingly exploited form of metadata that can be leveraged to support a variety of data management and analytical tasks. This thesis investigates the use data dependencies to design metadata-driven approaches for three relevant tasks: *Data Imputation*, *Concept Drift Detection*, and *Query Relaxation*. It also contributes to the advancement of data profiling itself, by proposing new methods for discovering, interpreting, and explaining dependencies. The **first part of the thesis** provides a comprehensive overview of data profiling techniques and introduces two novel proposals: S-DIMε, a semantic-aware dependency discovery algorithm capable of handling long textual attributes, and RYAN, a visualization and explanation tool that employs Large Language Models to describe the evolution of discovered dependencies over time. The **second part of the thesis** focuses on data quality, with particular attention to the imputation of missing data. It presents RIFT, a metadata-driven framework for imputing multivariate time series, and an empirical study demonstrating the mutual benefits of integrating data profiling and metadata-driven imputation in dynamic scenarios. The **third part of the thesis** explores additional use cases of metadata. In particular, it will introduce a novel metadata-driven method for concept drift detection, showing that monitoring the evolution of dependencies provides accurate drift estimations in machine learning pipelines. Furthermore, a metadata-driven query relaxation tool, namely REQUIRED, is introduced to support flexible and context-aware data exploration. Together, the promising results observed in the proposed contributions pave the way for future investigations aimed at developing more ro-

bust, interpretable, and metadata-centric paradigms that underlie increasingly complex data-driven tasks.

ABSTRACT (ITALIAN)

La digitalizzazione del settore pubblico ha portato alla disponibilità di collezioni di dati ampie ed eterogenee, evidenziando la necessità di metodi efficaci per estrarre, interpretare e sfruttare conoscenza dai dati. A tal fine, le tecniche di *Data Profiling* consentono di inferire automaticamente metadati che descrivono la struttura, i pattern e le relazioni all'interno dei dataset. Tra questi, le dipendenze tra dati rappresentano una forma di metadato sempre più utilizzata, che può essere sfruttata per supportare una varietà di attività di gestione e analisi dei dati. Questa tesi esplora l'uso delle dipendenze tra dati per lo sviluppo di approcci *metadata-driven* rivolti a tre compiti rilevanti: *Data Imputation*, *Concept Drift Detection* e *Query Relaxation*. Essa contribuisce inoltre all'avanzamento del campo del *Data Profiling*, proponendo nuovi metodi per il discovery, l'interpretazione e la spiegazione delle dipendenze. La **prima parte della tesi** fornisce una panoramica completa delle tecniche di *Data Profiling* e introduce due proposte innovative: S-DiMε, un algoritmo di discovery di dipendenze semantic-aware, capace di gestire attributi testuali di grandi dimensioni, e RYAN, uno strumento di visualizzazione e spiegazione che impiega *Large Language Model* per descrivere l'evoluzione delle dipendenze scoperte nel tempo. La **seconda parte della tesi** si concentra sulla qualità dei dati, con particolare attenzione all'imputazione dei valori mancanti. Essa presenta RIFT, un framework *metadata-driven* per l'imputazione di serie temporali multivariate, e uno studio empirico che dimostra i benefici reciproci derivanti dall'integrazione tra *Data Profiling* e imputazione guidata dai metadati in scenari dinamici. La **terza parte della tesi** esplora ulteriori casi d'uso dei metadati. In particolare, introduce un nuovo metodo *metadata-driven* per la rilevazione del *concept drift*, mostrando come il monitoraggio dell'evoluzione delle dipendenze consenta di stimare in modo accurato i cambiamenti nei modelli di apprendimento automatico. Inoltre, viene presentato uno strumento interattivo di *Query Relaxation*, denominato REQUIRED, che supporta un'esplorazione

dei dati flessibile e sensibile al contesto. Complessivamente, i risultati promettenti osservati nei contributi proposti aprono la strada a future ricerche volte a sviluppare paradigmi più robusti, interpretabili e incentrati sui metadati, che si pongono alla base di task guidati dai dati sempre più complessi.

CONTENTS

1 Introduction 1

I Data Profiling and its applications

2 Data Profiling 9
- 2.1 Data Profiling 9
- 2.2 Functional Dependencies 11
  - 2.2.1 Updating RFDs over time 15
- 2.3 Data Profiling Use Cases 17
  - 2.3.1 Schema Normalization 17
  - 2.3.2 Data Cleaning 18
  - 2.3.3 Query Optimization 20
  - 2.3.4 Query Relaxation 21
  - 2.3.5 Data Integration 21
  - 2.3.6 Machine Learning 22
- 2.4 Dependency Discovery: Literature review 22

3 Data Profiling: From theory to real world scenarios 29
- 3.1 Open Challenges and Research Gaps 29
  - 3.1.1 Challenge 1: Semantic-aware discovery algorithms 29
  - 3.1.2 Challenge 2: Visualizing and Explaining discovery results 30
- 3.2 Solving challenge 1: S-DIMε [42] 31
  - 3.2.1 The S-DIMε algorithm 31
  - 3.2.2 Evaluation 41
- 3.3 Solving Challenge 2: RYAN [41] 51
  - 3.3.1 The RYAN tool 52
  - 3.3.2 User Study 64

II Metadata-driven Data Imputation

4 Data Imputation 73
- 4.1 Problem Overview 73
  - 4.1.1 RFD_c-based Data Imputation 74
- 4.2 Literature Review 75
  - 4.2.1 Imputation of Relational Data 76
  - 4.2.2 Imputation of Time Series Data 86

4.3 Open Challenges and research Gaps 89
5 Data Imputation for Time Series Data 91
5.1 Problem Overview 91
5.2 RIFT: Multivariate Time Series imputation [172] 92
5.2.1 The proposed approach 93
5.2.2 Experimental Evaluation 100
6 Data Imputation in Incremental Scenarios 115
6.1 Problem Overview 115
6.1.1 Missing value imputation in incremental scenarios 115
6.1.2 Impact of missing value imputation on RFD_c discovery 116
6.2 Data dependency meets data imputation [26] 117
6.2.1 Experimental Settings 118
6.2.2 Analysis Method 128
6.2.3 Improving the imputation quality in incremental scenarios (RQ1) 129
6.2.4 Mitigating the effect of missing values in the discovery processes (RQ2) 134
6.2.5 Discussion 141

III Applying Data Profiling in other use cases
7 Applying Data Profiling in other use cases 149
7.1 Concept Drift Detection 149
7.1.1 Literature Review 150
7.1.2 Problem Overview 152
7.1.3 An RFD-based approach for concept drift detection [44] 154
7.2 Query Relaxation 178
7.2.1 Literature Review 179
7.2.2 Problem Overview 183
7.2.3 Required [39] 185
8 Conclusion and Future Directions 193

Bibliography 199

LIST OF FIGURES

Figure 2.1 A general taxonomy of data profiling tasks 10

Figure 3.1 Overview of the S-DIMε algorithm.. The algorithm is characterized by two different steps, namely *Preprocessing* and *Discovery*. The former is depicted in detail in Figure 3.2. 32

Figure 3.2 Overview of the preprocessing module. 33

Figure 3.3 Comparison of S-DIMε and DIMε preprocessing times. 43

Figure 3.4 Comparison of S-DIMε and DIMε memory consumption. 43

Figure 3.5 Categorization of the RFDs discovered by S-DIMε with different coverage values (0 (a), 0.2 (b), 0.4 (c), 0.6 (d), 0.8 (e)) and different threshold configurations (C_1, C_2, and C_3). 47

Figure 3.6 Overview of the architecture of RYAN. 53

Figure 3.7 Overview of the Data Processing module of RYAN. 55

Figure 3.8 RYAN's homepage. 59

Figure 3.9 Tutorial page. 59

Figure 3.10 Overview of RYAN's visual components. 60

Figure 3.11 Overview of RYAN interfaces. 62

Figure 3.12 Answers provided by participants for Q5-Q18. 68

Figure 5.1 Time series with D dimensions and length n. 91

Figure 5.2 Overview of the RIFT Framework. 93

Figure 5.3 Snippet of the *Opportunity (Act.5)* dataset used for the Running Example. 94

Figure 5.4 Average number of discovered RFDs and discovery time by varying the number of dimensions. 95

Figure 5.5 Discovery times with and without RIFT's optimizations. Scale-ups computed between RIFT (red) and RFD_c discovery without optimizations (blue). Asterisks mark scale-ups computed considering the TL. 104
Figure 5.6 Imputation times with and without applying RYAN's optimizations. The scale-ups are computed between RYAN (red) and the imputation without optimizations (blue). 105
Figure 5.7 Impact of the proposed approach in terms of *Precision* and *Recall*. 106
Figure 5.8 Influence of k (upper box) and α (middle box) on the imputation accuracy (the right axis in blue is for the RMSE), and on the execution times (lower box). 107
Figure 5.9 Impact of RIFT in terms of *RMSE* 109
Figure 5.10 Visualization of the impact of RIFT in terms of *Precision* and *Recall*. 110
Figure 5.11 Average gain when applying RIFT as pre-processing step in terms of *Precision*, *Recall*, *RMSE*, and execution times. 110
Figure 5.12 Impact of RIFT in terms of *Precision*, *Recall*, and *RMSE* when considering MVs caused by sensor failures on the *Opportunity (Act. 5) dataset*. 111
Figure 6.1 Overview of the *Imputation Pipeline* at a generic step, the batch size k is set to 4. 121
Figure 6.2 Overview of the *Discovery Pipeline* at a generic step, the batch size k is set to 4. 124
Figure 6.3 Comparison between the imputation results of the *Imputation Pipeline* and those of the baselines. 130
Figure 6.4 RMSE gain of the imputation pipeline w.r.t. the baselines for each dataset (a) and for each missing rate (b). 131

Figure 6.5 Imputation results with other missing mechanisms. 133
Figure 6.6 Imputation results when using DERAND. 134
Figure 6.7 Comparison of the RFD$_c$s extracted by the *Discovery Pipeline* with those of the oracle. 135
Figure 6.8 Comparison of the RFD$_c$s extracted by the baseline with those of the oracle. 136
Figure 6.9 RFDs Not Found reduction (%). 137
Figure 6.10 Comparing the baseline and the *Discovery Pipeline*. 138
Figure 6.11 Comparing the baseline and the *Discovery Pipeline* with other missing mechanisms (a) and FDs (b). 139
Figure 7.1 A general overview of the proposed approach. 157
Figure 7.2 An overview of the initialization and updating steps underlying the proposed approach. 158
Figure 7.3 Correlations of metrics with the models' F1-Measure (the lower the better) on datasets with *Known Drift* (1-18). 171
Figure 7.4 Correlations with the models' F1-Measure on datasets with *Known Drift*. 171
Figure 7.5 Correlations with different ML models. 174
Figure 7.6 Correlations with the models' F1-Measure on datasets with *Unknown Drift*. 174
Figure 7.7 Correlations obtained by using smaller test batches. 175
Figure 7.8 Trend analysis. 176
Figure 7.9 Overview of REQUIRED's architecture. 186
Figure 7.10 Interface for performing a query on a dataset. 186
Figure 7.11 Interface for the selection of an RFD. 187
Figure 7.12 Query extension interface. 187
Figure 7.13 Query relaxation interface. 187

LIST OF TABLES

Table 2.1 A summary of the main symbols used throughout the thesis. 11
Table 2.2 A snippet of a used car dataset. 15
Table 3.1 A snippet of the Universal Studios reviews dataset. 34
Table 3.2 $M_{\mathsf{Rating}_{abs}}$ 34
Table 3.3 $M_{\mathsf{Rating}_{abs},\mathsf{Title}_{lev}}$ 35
Table 3.4 Overview of the datasets adopted in the experimental evaluation of S-DIMε. 41
Table 3.5 A snippet from the Gamestop dataset. 48
Table 3.6 A snippet from the Hotel dataset. 49
Table 3.7 A snippet from the *Kindle* dataset. 50
Table 3.8 The questionnaire proposed to participants. 66
Table 4.1 A snippet of the *Restaurant* dataset. 75
Table 4.2 Summary of imputation approaches for relational data by category, year, and technique. 77
Table 4.3 Summary of imputation approaches for time series data by category, year, and technique. 86
Table 5.1 Details of the datasets employed in the evaluation and number of MVs injected in each configuration. 97
Table 5.2 Comparison of RIFT's *Precision* and *Recall* when employing different measures to perform the Step 1 clustering. The best configuration is highlighted in green. 113
Table 6.1 A snippet of the *Restaurant* dataset with a new batch of tuples added at time $\tau + 1$. 118
Table 6.2 Details of the datasets employed in the evaluation. 119
Table 6.3 Number of injected missing values for each dataset. 119

Table 7.1 An example of RFD_c sets holding at two time instants τ and $\tau + 1$. 155
Table 7.2 Reference table for the notations used to categorize different type of RFD evolution. 162
Table 7.3 RFD-based divergences. 164
Table 7.4 Metrics inspired by confusion matrix items. 166
Table 7.5 Details of the datasets employed in the evaluation. 170
Table 8.1 Summary of the papers presented in this thesis to introduce advancements in Data Profiling, along with the key findings/-contributions of each work. 194
Table 8.2 Summary of the papers presented in this thesis to introduce advancements in Data Imputation, along with the key findings/-contributions of each work. 195
Table 8.3 Summary of the papers presented in this thesis to introduce novel applications of RFDs, along with the key findings/contributions of each work. 196

1

INTRODUCTION

Recent years have witnessed a significant shift within the Italian public administration (PA) toward digitalization, which involves the integration of digital technologies and data-driven approaches into the operations, service delivery, and decision-making processes. The goal is to leverage technology to improve the efficiency, transparency, accessibility, and overall effectiveness of governance. This transformation process is particularly significant given the vast scale and complexity of the Italian public sector, which comprises approximately 8,000 municipalities and 23,000 local agencies, collectively serving a population of around 61 million citizens [61]. In this scenario, the National Recovery and Resilience Plan (PNRR) has been an important driving force. The PNRR is a strategic plan designed to stimulate economic recovery and long-term resilience in the wake of the challenges posed by the COVID-19 pandemic. It is composed of 6 missions, among which Mission 1 is dedicated to the digital transition. The plan places a significant emphasis on this mission, allocating around 20% of the funds available (40.32 out of 191.5 billion euros), of which 9.75 billion are specifically dedicated to the digitalization of the PA[1]. More specifically, PNRR includes a wide range of digital initiatives that encompass the development of digital services, the modernization of administrative processes, and the improvement of digital infrastructure. Furthermore, the plan promotes the use of data-driven decision-making processes and emphasizes the importance of cybersecurity to both protect sensitive government information and ensure trust in digital services. One of the primary objectives of the PNRR in relation to digitalization is to bring Italy's public administration up to par with European standards and best practices. In fact, these national goals align with the broader communitarian digitalization strategy: the European Union is also pushing for

1 https://www.governo.it/it/approfondimento/digitalizzazione-innovazione-competitivit-e-cultura/16701

the digitalization of member countries with its Digital Decade[2] program, which sets digitalization targets for citizens, businesses, infrastructures, and public services. Europe has very high expectations regarding digitization: the deployment of digital solutions is anticipated to have a profound impact on industries and is expected to increase energy efficiency, improve public transport, and enhance healthcare outcomes through remote patient monitoring. Furthermore, the European Union expects important benefits for the economy. For example, mobility data is expected to save public transportation users over 27 million hours and up to 20 billion euros per year in labor costs. Studies and simulations suggest that the digital transformation of Europe could lead to a cumulative additional GDP (Gross Domestic Product) growth of 14% by 2030, and Europe stands to gain 2 trillion euros in value and create 2 million additional jobs [56].

Challenges. Although the emerging digitalized context promises many benefits, it also raises several challenges. Digitalization has led to the generation of large amounts of data that have to be processed and/or analyzed. Within these massive datasets, valuable insights and patterns can be discovered, potentially offering a deeper understanding of various phenomena. However, uncovering this knowledge is not a straightforward task. The vast scale of data requires approaches to extract knowledge in a compact and interpretable form that can effectively support analyses and decision-making processes. In addition, this knowledge could also be used to support or define novel approaches to address other challenges that characterize this digitalized scenario. Among the latter, one critical issue concerns data quality: datasets often contain missing values that can degrade the knowledge extracted [123] and introduce biases that lead to distorted results [10]. Such missing data arise in many real-word settings for several reasons, including sensor malfunctions, incomplete or delayed data entry, or information loss due to integration of unrelated data. These situations are common in industrial and administrative environments, making the problem of missing values particularly widespread and impactful. Thus, there is a

2 https://eufordigital.eu/new-step-towards-the-digital-decade-2030-policy-programme-comes-into-force/

need to introduce *Data Imputation* mechanisms to impute missing values [141], thereby improving the quality of datasets and leading to an improvement of downstream tasks [68]. Moreover, the problem of data imputation has to be investigated by considering dynamic scenarios. As a matter of fact, data is not a static entity, but it evolves over time with insertions, deletions, or updates. This dynamic context also raises other challenges related to concept drift, i.e., the phenomenon where the underlying data distribution or the meaning of data samples changes over time, challenging the stability and performance of machine learning models, data analytics, and decision-making processes[104, 187]. Thus, concept drift can have significant implications and adds a layer of complexity [110] to digitalization efforts. This dynamic nature of data requires *Concept Drift Detection* approaches to monitor data evolution and ensure that decision-makers are working with the most relevant and up-to-date information. Finally, another challenge raised by the vast amount of data generated by digitalization concerns the ability to query large datasets effectively. As a matter of fact, it is not easy for a user to achieve the desired outputs, and it may be necessary to change the imposed conditions several times before getting a satisfying result. For this reason, to support such analytical processes, it is necessary to implement effective *Query Relaxation* approaches to automatically relax or modify search criteria to broaden the scope of data retrieval, considering the trade-off between the size of the result set and its relevance with respect to the original user intent.

Thesis Objective. The growing digitalization of public administration has led to the generation of vast and heterogeneous datasets, requiring the definition of approaches capable of extracting knowledge in the form of patterns, statistics, and relationships. As already noted a decade ago, *"If we just have a bunch of data sets in a repository, it is unlikely anyone will ever be able to find, let alone re-use, any of this data. With adequate metadata, there is some hope, but even so, challenges will remain..."* [113]. To this end, *Data Profiling* techniques enable to automatically extract knowledge in the form of metadata from datasets [1]. Metadata range from simple statistics (e.g., data type, value distribution, number of missing values) to more complex relationships (e.g., functional

dependencies, inclusion dependencies). Over the past years, the use of metadata to support downstream tasks has already shown promising results. Metadata-driven methods have been successfully applied in contexts such as anomaly detection [31], data integration [179], missing value imputation [30], and the optimization of machine learning models in both batch and online modes [115]. Consequently, exploring new practical applications of data profiling and/or improving existing metadata-driven techniques represents a relevant research direction, especially considering the dynamic and heterogeneous context of public sector digitalization.

In light of these previously discussed issues and given the potential of Data Profiling in supporting downstream tasks, this thesis focuses on exploring its practical applications by introducing new metadata-driven approaches. Specifically, this thesis will focus on the application of a specific type of metadata, namely *Data Dependencies*, to different analytical and data management tasks. In particular, it will present metadata-driven approaches for *Data Imputation*, *Concept Drift Detection*, and *Query Relaxation*. The former represents the main focus of this thesis, as it addresses a crucial aspect of data quality that should also be considered in more complex scenarios, such as the imputation of time series data [172], a frequently observed type of data for which the use of dependencies remains largely unexplored. Moreover, this thesis also investigates the problem of metadata-driven imputation in dynamic scenarios [26]. Finally, it will also explore the use of metadata in other practical applications. In particular, it will present a novel metadata-driven approach to quantify concept drift [44], as well as a metadata-driven query relaxation tool [39].

In addition to the practical application of metadata, part of this thesis also focuses on addressing some Data Profiling-related challenges, in order to expand the overall applicability of these approaches. In particular, this thesis will present a dependency discovery algorithm [42], which enables the use of semantic similarity measures, allowing the identification of dependencies even in datasets with long textual attributes. Finally, a novel dependency visualization and explanation tool will be presented [41]. It enables non-expert users to explore and analyze metadata evo-

lution over time.

THESIS OUTLINE. This thesis is composed of three main parts. **Part I** provides an overview of existing Data Profiling techniques and use cases, providing background notions concerning the most relevant types of profiling metadata, with a particular emphasis on Functional Dependencies (FDs) and Relaxed Functional Dependencies (RFDs). This part will also discuss open challenges and research gaps, particularly focusing on the necessity of devising more general discovery algorithms suitable for different types of data values and on the lack of available to interpret and explain discovered metadata. Following this discussion, Part I will present S-DIME [42], an RFD discovery algorithm capable of applying semantic measures to evaluate similarity during the discovery process, thereby enabling the extraction of meaningful RFDs also in datasets characterized by long texts. Then, the first part will end with the presentation of RYAN [41], a visualization tool that leverages Large Language Models (LLMs) to explain the evolution of RFDs over time. **Part II** focuses on Data Imputation, discussing the problem and providing a literature review of existing techniques. Then, the problem of data imputation for time series data will be discussed in detail, followed by the presentation of RIFT [172], an RFD-based imputation approach tailored for multivariate time series data. Part II will also detail the problem of data imputation in dynamic scenarios and will discuss an empirical study aimed at demonstrating the usefulness of the cooperation between data profiling and RFD based imputation algorithms in such scenarios [26]. The last part will focus on other use cases of Data Profiling techniques. In particular, **Part III** will discuss the problem of *Concept Drift Detection*, as well as current open challenges. Then, it will describe a novel RFD-based concept drift detection approach [44], which analyzes metadata evolution to provide a quantitative drift measure. Finally, Part III will discuss the problem of *Query Relaxation* and will provide an overview of existing methods and challenges. Then, it will describe REQUIRED [39], a novel RFD-based query relaxation tool aimed at enriching the query result set by including items that closely resemble the ones originally searched by the user.

Part I

DATA PROFILING AND ITS APPLICATIONS

2

DATA PROFILING

This chapter provides an overview of Data Profiling techniques. More specifically, Section 2.1 discusses the different types of profiling tasks and metadata, while Section 2.2 focuses on functional dependencies, providing formal definitions and examples. Then, Section 2.3 discusses several use cases in which dependencies were employed in recent years, while Section 2.4 concludes the chapter with a review of the algorithms used to automatically infer dependencies from data.

2.1 DATA PROFILING

Data Profiling refers to the set of methods and techniques aiming at extracting metadata from a given dataset. Data Profiling tasks can be categorized into different groups [1] according to the type of metadata extracted, which are depicted in Figure 2.1. In particular, metadata range from simple single-column statistics to more complex multi-column relationships. The former provide a basic understanding of the dataset at hand and typically comprises value distributions, data types, patterns, number of missing values, number of distinct values, and maximum/minimum values. These kinds of statistics are mostly leveraged by Database Management Systems (DBMS) for query optimization [2]. On the other hand, multi-column metadata refers to patterns and relationships that enable the analysis of inter-value dependencies across columns. This category comprises correlations, association rules, and the identification of clusters and outliers. The last category is represented by data dependencies, which are still multi-column properties but, given their importance, are often considered as a category on their own. One of the most common Data Profiling tasks concerning dependencies is the discovery of Unique Column Combinations (UCCs) [96], which entails the identification of sets of columns that satisfy the uniqueness constraint, leading to the identification of rows within a dataset. In fact, in

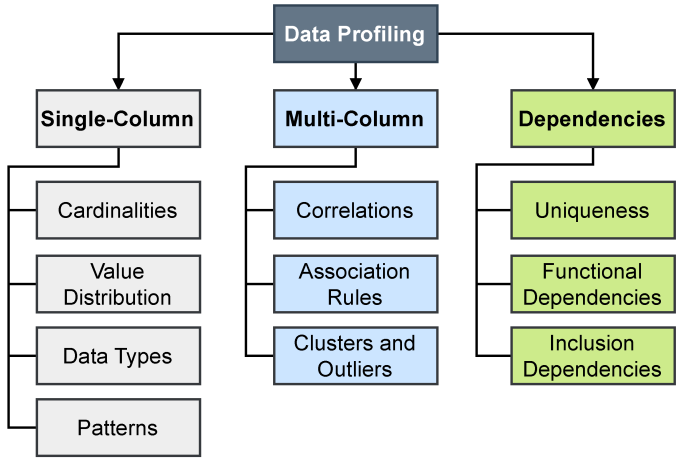


Figure 2.1: A general taxonomy of data profiling tasks

real-world applications, the discovery of UCCs is the first step to identify the primary key of a relation. Another important task concerns the discovery of Inclusion Dependencies (INDs) [105, 176], which state that all the values or combinations of them from a set of columns also appear in another set of columns. Thus, INDs can be used to support the identification of foreign keys. Finally, another type of dependency is represented by Functional Dependencies (FDs), which describe functional relationships among sets of attributes. FDs and their extensions have shown promising applications in several domains, spanning from Data Cleaning [30] to Data Integration [179], and are at the basis of this thesis. For this reason, the following section will provide a detailed discussion of functional dependencies, as well as the necessary theoretical notions and practical examples. Notice that, in order to facilitate the reading, Table 2.1 lists the key notations used throughout the section.

Symbol	Description
$\mathcal{R}$	Relational database schema
R	Relation schema
r	Relation instance
$attr(R)$	Relation schema attributes
A, B	Attribute
$dom(A)$	Attribute domain
X, Y	Set of attributes
t	Tuple
$t[A]$	Projection of tuple t on attribute A
φ, ρ	Relaxed functional dependency
φ'	Generalization of φ
φ''	Specialization of φ
c	Predicate
ϕ_1, ϕ_2	Conjunction of predicates
ψ	Coverage measure
ϵ, α, β	Threshold
Σ	Set of RFDs
τ	Time instant

Table 2.1: A summary of the main symbols used throughout the thesis.

2.2 FUNCTIONAL DEPENDENCIES

The following definition recalls the basic concepts of relational databases, which will be the foundation of the subsequent concepts related to data dependencies.

Definition 1 (Relational database concepts). *A relational database schema $\mathcal{R}$ is a collection of relation schemas $(R_1, \ldots, R_n)$, where each R_i is defined over a fixed set of attributes $attr(R_i)$. Each attribute A_k is associated with a domain $dom(A_k)$, which can be finite or infinite. A relation instance r_i of R_i is a set of tuples t such that for each attribute $A_k \in attr(R_i)$, the projection of tuple t on the attribute A_k, i.e., $t[A_k]$,*

belongs to $dom(A_k)$ $\forall t \in r_i$. *A database instance* r *of* $\mathcal{R}$ *is a collection of relations* $(r_1, \ldots, r_n)$, *where each* r_i *is a relation instance of* R_i, *for* $i \in [1, n]$.

Functional Dependencies (FDs) describe integrity constraints among relation attributes. More formally, an FD can be defined as follows:

Definition 2 (Functional Dependency). *Given a relation schema* R, *a Functional Dependency (FD) represents an integrity constraint that expresses a relationship between two sets of attributes* X *and* Y, *denoted as* $X \rightarrow Y$ (X *implies* Y), *with* $X, Y \subseteq attr(R)$ *and* $X \cap Y = \emptyset$. *An FD is satisfied on an instance* r *of* R *if and only if for every pair of tuples* (t_1, t_2), *whenever* $t_1[X] = t_2[X]$, *then* $t_1[Y] = t_2[Y]$; *where* $t_i[X]$ *represents the projection of the tuple* t_i *on the attribute set* X. *The attribute set* $X = X_1, X_2, ..., X_h$ *represents the Left Hand Side (LHS) of the FD, whereas the set* $Y = Y_1, Y_2, ..., Y_k$ *is the Right Hand Side (RHS).*

The definition of FD has been recently extended to deal with real-world data. The latter often contains inconsistencies and inaccuracies, leading to the violation of the strict constraints adopted by FDs. Moreover, many modern datasets include continuous, real-valued attributes (e.g., time series datasets), which can also easily lead to FD violations. This problem required the introduction of more flexible constraints, leading to the introduction of Relaxed Functional Dependencies (RFDs), which can be categorized according to the applied relaxation criteria [46]. In particular, RFDs relaxing on the *comparison* criteria (namely RFD_cs) enable the usage of similarity/distance constraints as a matching paradigm, while RFDs relaxing on the *extent* criteria (namely RFD_es) admit a limited number of violations.

Definition 3 (Relaxed Functional Dependency). *A Relaxed Functional Dependency* φ *is formally defined as follows:*

$$\mathbb{D}_C : X_{\Phi_1} \xrightarrow{\Psi \leq \varepsilon} Y_{\Phi_2}$$

where:

- $\mathbb{D}_C = \{t \in dom(R) | \bigwedge_{i=1}^{m} C_i(t[A_i])\}$ *with* $c = (c_1, \ldots, c_m)$ *and each* c_i *is a predicate on* $dom(A_i)$ *filtering the tuples on which* φ *applies;*

- $X = X_1, \ldots, X_h$ *and* $Y = X_1, \ldots, X_k$*, with* $X, Y \subseteq attr(R)$ *and* $X \cap Y = \emptyset$*;*
- $\Phi_1 = \bigwedge_{X_i \in X} \phi_i[X_i] (\Phi_2 = \bigwedge_{Y_j \in Y} \phi_j[Y_j], resp.)$*, where* $\phi_i (\phi_j, resp.)$ *is a conjunction of predicates on* $X_i (Y_j, resp.)$ *with* $i = 1, \ldots, h$ $(j = 1, \ldots, k, resp.)$*. For any pair of tuples* $(t_1, t_2) \in dom(R)$*, the constraint* $\Phi_1 (\Phi_2, resp.)$ *is true if* $t_1[X_i]$ *and* $t_2[X_i]$ $(t_1[Y_j]$ *and* $t_2[Y_j], resp.)$ *satisfy the constraint* $\phi_i (\phi_j, resp.) \forall i \in [1, h] (j \in [1, k], resp.)$*. In particular, a constraint* ϕ*, over an attribute* $A \in attr(R)$*, is composed of a similarity (or distance) function, a comparison operator, and a threshold.*
- Ψ *is a coverage measure defined on* $dom(R)$ *that quantifies the amount of tuples satisfying or violating* φ*. It is defined as a function* $\Psi : dom(X) \times dom(Y) \Rightarrow \mathbb{R}^+$*, where* $dom(X)$ *denotes the cartesian product of the domains of the attributes in* X*;*
- ε *is a threshold representing the upper bound (or lower if the comparison operator is* $\geq$*) for the coverage measure.*

Considering a relation instance $r \subseteq \mathbb{D}_C$ *defined on* R*,* r *satisfies the* RFD φ $(r \models \varphi)$ *if and only if* $\forall (t_1, t_2) \in r$*, if* Φ_1 *returns true then* Φ_2 *almost always return true, where almost always is quantified by the constraint* $\Psi \leq \varepsilon$*.*

For the sake of simplicity and without loss of generality, in the following, we only consider RFDs with a single attribute on the RHS, i.e., $X_{\Phi_1} \xrightarrow{\Psi \leq \varepsilon} A_{\phi_2}$. Moreover, the following examples will refer to constraints defined through distance functions, with a comparison operator ($\leq$) and the associated threshold. Notice that a specific distance function is applied according to the nature of the attributes (e.g., absolute difference for numerical attributes and the Levenshtein distance [117] for textual ones).

Example 2.2.1. Consider the tuples ranging from t_0 to t_6 of the snippet of a used cars dataset shown in Table 2.2, then an holding RFD_c is:

$$\varphi : \mathsf{Model}_{\leq 4}, \mathsf{Year}_{\leq 1} \xrightarrow{\Psi_{err(0)}} \mathsf{Price}_{\leq 300}$$

where $\Psi_{err(0)}$ denotes that φ holds on the complete relation instance (i.e., from t_0 to t_6).

In this example, φ denotes that whenever two tuples have similar values on Model and Year, which is the case of (t_0, t_6) and (t_2, t_4), then they have a similar Price.

One of the most important characteristics of an RFD is its *minimality*, which guarantees that the RFD_c no longer holds after either (*i*) increasing one or more thresholds on the LHS constraints, (*ii*) removing an LHS attribute, or (*iii*) decreasing the RHS threshold.

Definition 4 (**Minimal RFD**). *Given an instance r of a relation schema R, an* RFD φ: $X_{1\leq\alpha_1}, \ldots, X_{h\leq\alpha_h} \xrightarrow{\Psi\leq\varepsilon} A_{\leq\beta}$ *is minimal iff:*

- $\rho : X_{1\leq\alpha_1+\varepsilon_1}, \ldots, X_{h\leq\alpha_h+\varepsilon_h} \xrightarrow{\Psi\leq\varepsilon} A_{\leq\beta}$ *does not hold on r, where $\forall j = 1, \ldots, h$ then $\varepsilon_j \geq 0$, $\exists j = 1, \ldots, h$ s.t. $\varepsilon_j > 0$; and*
- $\rho : X_{1\leq\alpha_1}, \ldots, X_{i-1\leq\alpha_{i-1}}, X_{i+1\leq\alpha_{i+1}}, \ldots, X_{h\leq\alpha_h} \xrightarrow{\Psi\leq\varepsilon} A_{\leq\beta}$ *does not hold on r, where $\exists i = 1, \ldots, h$; and*
- $\rho : X_{1\leq\alpha_1}, \ldots, X_{h\leq\alpha_h} \xrightarrow{\Psi\leq\varepsilon} A_{\leq\beta-\varepsilon_{h+1}}$ *does not hold on r, where $\varepsilon_{h+1} > 0$.*

Since it is difficult to apriori define proper RFDs characterizing real-world scenarios, it is necessary to exploit discovery algorithms to automatically infer them from data [49, 87, 168]. This entails searching for RFDs holding on a given dataset.

Definition 5 (Discovery of minimal RFDs)**.** *Given a relational schema R and an instance r of R, a discovery process consists of finding the set Σ of all possible* RFDs *$\varphi : X_{\Phi_1} \rightarrow A_{\phi_2}$ that hold on r (i.e., $r \vDash \Sigma$) such that $\forall \varphi \in \Sigma$, φ is also minimal. In other words, $\nexists\, \rho \in \Sigma$, with ρ minimal with respect to φ.*

Notice that RFD discovery algorithms deal with a problem that, in the worst case, is exponential in the number of attributes, since they have to browse a search space that considers all possible attribute combinations. More formally, given a relation instance r with M attributes, the complexity of discovering RFDs is defined as follows:

		Model	Year	# Owners	Price
	t_0	Hyundai i10	2016	2	6.000
-	t_1	Kia Picanto	2015	1	4.000
	t_2	Renault Clio	2018	1	8.300
	t_3	Ford Fiesta tdci	2022	1	10.500
	t_4	Renault Clio dCi	2019	1	8.600
	t_5	Fiat Panda	2019	1	8.500
	t_6	Hyundai i10 1.0	2016	1	6.250
+	t_7	Renault Clio	2019	3	6.000

Table 2.2: A snippet of a used car dataset.

$$\sum_{k=2}^{M} \binom{M}{k} \cdot k$$

Moreover, when the distance thresholds have to be automatically inferred, the algorithm must consider all possible d dispositions of distance thresholds for each attribute combination [45]:

$$\sum_{k=2}^{M} \binom{M}{k} \cdot k d^k$$

Nevertheless, thanks to specific pruning strategies, mostly based on the minimality property, it is possible to significantly reduce the search space, widely lowering discovery times [101].

2.2.1 *Updating* RFDs *over time*

The nature of real-world data is inherently dynamic, constantly evolving over time following inserts, deletions, and updates of data. Consequently, RFDs must also evolve accordingly, with respect to the type of performed operations. In what follows, we first discuss the evolution of RFDs after deletion operations and then after the insertion ones. Notice that no specific considerations need to be done for update operations, since they can be represented as a deletion followed by an insertion.

Let Σ be the set of RFDs holding at time τ, and Σ' the set of RFDs holding at time $\tau+1$. A tuple deletion at time $\tau+1$ cannot invalidate any RFDs $\varphi \in \Sigma$. Nevertheless, it could make a given $\varphi \in \Sigma$ no longer minimal, requiring the evaluation of one or more *generalizations* of φ, which can hold at time $\tau+1$.

Definition 6 (Generalization of an RFD). *Given a relation schema R, an instance r of R, and an* RFD φ: $X_{1\leq\alpha_1}, \ldots, X_{h\leq\alpha_h} \xrightarrow{\Psi\leq\varepsilon} A_{\leq\beta}$ *holding on r. An* RFD $\varphi' : X'_{\Phi_1} \xrightarrow{\Psi\leq\varepsilon} A_{\phi_2}$ *is a generalization of φ iff:*

- $\varphi' : X_{1\leq\alpha_1+\varepsilon_1}, \ldots, X_{h\leq\alpha_h+\varepsilon_h} \xrightarrow{\Psi\leq\varepsilon} A_{\leq\beta-\varepsilon_{h+1}}$ *holds on r, where $\forall j = 1, \ldots, h+1$ then $\varepsilon_j \geq 0$, $\exists j = 1, \ldots, h+1$ s.t. $\varepsilon_j > 0$; or*
- $\varphi' : X_{1\leq\alpha_1}, \ldots, X_{i-1\leq\alpha_{i-1}}, X_{i+1\leq\alpha_{i+1}}, \ldots, X_{h\leq\alpha_h} \xrightarrow{\Psi\leq\varepsilon} A_{\leq\beta}$ *holds on r, where $\exists i = 1, \ldots, h$; or*
- $\varphi' : X_{1\leq\alpha_1+\varepsilon_1}, \ldots, X_{i-1\leq\alpha_{i-1}+\varepsilon_{i-1}}, X_{i+1\leq\alpha_{i+1}+\varepsilon_{i+1}}, \ldots, X_{h\leq\alpha_h+\varepsilon_h} \xrightarrow{\Psi\leq\varepsilon} A_{\leq\beta-\varepsilon_{h+1}}$ *holds on r, where $\forall j = 1 \ldots, h+1$ then $\varepsilon_j \geq 0$, $\exists j = 1, \ldots, h+1$ s.t. $\varepsilon_j > 0$, and $\exists i = 1, \ldots, h$.*

As an example, consider the tuples ranging from t_0 to t_6 shown in Table 2.2, and suppose that t_1 gets deleted. Thus, the RFD_c $\varphi : \mathsf{Model}_{\leq 4}, \mathsf{Year}_{\leq 1} \xrightarrow{\Psi_{err(0)}} \mathsf{Price}_{\leq 300}$ is still valid, but it is no longer minimal, since $\varphi' : \mathsf{Year}_{\leq 1} \xrightarrow{\Psi_{err(0)}} \mathsf{Price}_{\leq 300}$ holds on the updated dataset.

On the other hand, a tuple insertion at time $\tau+1$ can invalidate an RFD $\varphi \in \Sigma$, requiring the evaluation of one or more *specializations* of φ, which can hold at time $\tau+1$.

Definition 7 (Specialization of an RFD). *Given a relational schema R, an instance r of R, and an* RFD φ: $X_{1\leq\alpha_1} \ldots, X_{h\leq\alpha_h} \xrightarrow{\Psi\leq\varepsilon} A_{\leq\beta}$ *holding on r. An* RFD $\varphi'' : X''_{\Phi_1} \xrightarrow{\Psi\leq\varepsilon} A_{\phi_2}$ *is a specialization of φ iff:*

- $\varphi'' : X_{1\leq\alpha_1-\varepsilon_1}, \ldots, X_{h\leq\alpha_h-\varepsilon_h} \xrightarrow{\Psi\leq\varepsilon} A_{\leq\beta+\varepsilon_{h+1}}$ *holds on r, where $\forall j = 1, \ldots, h+1$ then $\varepsilon_j \geq 0$, and $\exists j = 1, \ldots, h+1$ s.t. $\varepsilon_j > 0$; or*
- $\varphi'' : X_{1\leq\alpha_1} \ldots, X_{h\leq\alpha_h}, X_{h+1\leq\alpha_{h+1}} \xrightarrow{\Psi\leq\varepsilon} A_{\leq\beta}$ *holds on r, where $\exists i = 1, \ldots, h+1$; or*

- $\varphi'' : X_{1\leq\alpha_1-\varepsilon_1},\dots,X_{h\leq\alpha_h-\varepsilon_h},X_{h+1\leq\alpha_{h+1}} \xrightarrow{\Psi\leq\varepsilon} A_{\leq\beta+\varepsilon_{h+2}}$ *holds on r,where* $\forall j = 1,\dots,h+2$ *then* $\varepsilon_j \geq 0$, *and* $\exists j = 1,\dots,h+2$ *s.t.* $\varepsilon_j > 0$, *and* $\exists i = 1,\dots,h+1$.

As an example, consider the tuples ranging from t_0 to t_6 shown in Table 2.2, and suppose that tuple t_7 is inserted. In this case, the RFD $\varphi : \mathsf{Model}_{\leq 4}, \mathsf{Year}_{\leq 1} \xrightarrow{\Psi_{err(0)}} \mathsf{Price}_{\leq 300}$ is no longer valid, since although (t_2,t_7) and (t_4,t_7) satisfy the constraints defined on the LHS of φ, they violate the RHS constraint on the attribute Price. However, the following specialization of φ holds on the updated dataset: $\varphi'' : \mathsf{Model}_{\leq 4}, \mathsf{Year}_{\leq 1}, \#\mathsf{Owners}_{\leq 1} \xrightarrow{\Psi_{err(0)}} \mathsf{Price}_{\leq 300}$.

2.3 DATA PROFILING USE CASES

This section provides an overview of existing use cases for Data Profiling techniques, with an emphasis on approaches that leverage (relaxed) functional dependencies.

2.3.1 *Schema Normalization*

Schema normalization refers to the process of removing redundancy in relational instances by decomposing the latter into relations with fewer attributes. FDs play an important role in this context, since they represent redundancy that has to be removed. As a matter of fact, the Boyce-Codd normal form (BCNF) [57] is based on FDs. While the problem of normalizing schemas according to this form has been extensively studied from a theoretical perspective, the practical implementation is still a complex and often manual task. For this reason, in [145] the authors introduce a semi-automatic data-driven algorithm. Their approach automatically discovers FDs, and includes a feature that automatically identifies reliable dependencies from the original set to reduce complexity. The authors also leverage t index-structures and parallelization to improve the execution times. While the previous work enabled only console-based interaction, in [77] the authors propose a web interface to explore schema transformations and to automatically generate transformation scripts. A more strict version of BCNF is the fourth normal form (4NF), which was

defined starting from the concept of Multivalued Dependency (MVD) [72]. More recent works also explored the application of applying MVDs to normalize XML data [9, 103].

2.3.2 *Data Cleaning*

Data Cleaning refers to the set of activities that aim to identify and correct corrupt, inaccurate, or irrelevant records from a set of data, and it is considered one of the most common use cases of Data Profiling [2]. As a matter of fact, several works in this field leverage dependencies to guide the cleaning process. For instance, Rezig et al. [158] exploit FDs to produce a graph that encodes data patterns and their interactions. Starting from this representation, they introduce Horizon, an end-to-end repair system that selects the most frequent patterns to ensure accurate repairs, while featuring a linear-time algorithm that scales to datasets with millions of records. On the other hand, a recent work [33] introduced a graph-based framework, namely GRIMP, for imputing missing values in relational datasets with both categorical and numerical attributes. GRIMP models the data as a heterogeneous graph and leverages graph neural networks with self-supervised multi-task learning and incorporates FDs to further improve imputation accuracy. Other works leveraged Denial Constraints (DCs), which generalize the concept of FD and other constraints. Among these, Chu et al. [55] introduced a data repairing framework that projects DCs on the dataset instance and compiles violations into a conflict hypergraph. Then, all the violations are resolved together with respect to a unified objective function. A popular example of data cleaning approach based on DCs is HOLOCLEAN [157], a probabilistic framework that leverages DCs for error detection and to enforce consistency, combining statistical inference with integrity constraints to automatically suggest high-confidence repairs. This approach was recently extended by Cui et al. [59] to enable the repairing of multi-source data, both structured and semi-structured. Other works extended the definition of FD to pursue the development of more refined data cleaning approaches. For instance, Zheng et al. [190] introduced the concept of Ontology Functional Dependencies (OFDs), which enables capturing similarities among

synonyms by leveraging ontologies. The authors also introduce OFDCLEAN, an OFD-based data repairing approach, showing that it outperforms HOLOCLEAN [157], albeit at a cost of additional overhead. On the other hand, in [84], the authors propose LLUNATIC, a data cleaning framework that aims to involve different types of constraints and strategies to repair data. To this end, they introduce a language to define constraints based on Equality Generating Dependencies (EGDs), which generalizes many of the constraints used in the literature, thus enabling the standardization of dependency representation and their extension to express inter-table constraints. This work was further extended in [85] by enabling schema mappings.

Existing works also explored the use of RFDs for data cleaning purposes. For instance, Bertossi et al. [24] leveraged Matching Dependencies (MDs), a specific type of RFD [46], introducing a chase-like repairing procedure which iteratively applies MDs upon reaching convergence. In [131], the authors propose an interactive system for data repairing tailored for non-expert users. The proposed system considers a fixed set of Conditional Functional Dependencies (CFDs), and requires a training step in which the user is asked to resolve a few violations. Then, the trained system is able to automatically determine when and how to apply a correction in case of a violation. On the other hand, Song et al. [170] leveraged Differential Dependencies (DDs), a specific type of RFD [46], for imputing missing values within datasets. To enrich the data imputation process, the authors studied the problem of maximizing the number of imputed missing values. In particular, the authors presented 4 different algorithms for treating missing values, i.e., an integer linear programming, its approximation, a randomized algorithm, and its derandomized version, called DERAND. The latter is identified by the authors as the ideal solution for imputing multiple missing values with a deterministic bound of approximation. While DERAND leverages RFDs only for detecting suitable candidates for imputing a missing value, RENUVER [30] leverages RFDs also for selecting the candidate to use, leading to more precise imputations.

2.3.3 *Query Optimization*

Query optimization refers to the process of selecting an efficient physical execution plan for a given query, with the goal of minimizing execution time and resource consumption. Most query optimizations leverage simple single-column metadata (i.e., statistics and cardinalities) [108]. However, there also exist several approaches that employ dependencies such as UCCs, FDs, and INDs to support different phases of the optimization process. As regards FDs, they were used to simplify grouping operations [174] that involve all the attributes of a holding FD. In particular, since all the tuples with the same values for the attributes on the LHS of the FD will also show the same values for the attributes in the RHS, the latter can be removed from the grouping operation. Similarly, the same approach can be applied for distinctness, since the latter is considered a special case of grouping [50], and for sorting operations [165]. In the context of duplicate elimination, it has also been shown that FDs can prevent unnecessary duplicate-removing projections, since redundancy can often be inferred directly from known dependencies [183]. Moreover, further refinements were introduced by Date & Darwen [60], who observed that, under certain conditions, the application of a DISTINCT after a selection may be redundant, as FDs can guarantee uniqueness of the result values. FDs were also applied for the optimization of joins between different relations. In particular, they have been employed to identify join trees that can be safely discarded, thereby reducing the complexity of the search space [69]. Moreover, Abiteboul et al. [4] demonstrated that an equality self-join can be avoided when the join attribute functionally determines all the attributes in the projection of the query. Other studies [5, 91] leveraged FDs to provide an output size bound for join queries. Finally, FDs can also be used to optimize selection operations. For instance, given a query with two equality predicates that involve two attributes that are also involved in an FD, the filtering can be applied by simply evaluating the attribute on the LHS and then only a single element of the attribute on the RHS. More recently, dependency-based query optimization has been investigated from a practical perspective, with systems that automatically discover, validate, and integrate dependencies

into the query optimizer, demonstrating substantial performance improvements across different benchmarks [121].

2.3.4 *Query Relaxation*

Query relaxation refers to the process of automatically relaxing query constraints in order to derive some approximate results when the answer set is small. Among metadata-driven approaches, earlier works leveraged small sets of precomputed statistical summaries to rewrite an approximate version of the original query [149]. A first proposal that leverages data dependencies was introduced in [137], in which the authors exploit Approximate Functional Dependencies (AFDs). The intuition underlying their approach is that, given a query, the attributes with the lowest influence should be relaxed first in order to reduce the impact of the relaxation process on the similarity of the results. To this aim, the authors use AFDs to compute influence weights that describe the influence of an attribute over the other ones, enabling the definition of an influence ranking of all attributes. More recently, Caruccio et al. [47] proposed an RFD-based approach for rewriting failing queries that are in the disjunctive normal form and involve both discrete and continuous attributes. More specifically, the proposed approach ranks the RFDs holding on the dataset according to a utility strategy that enables estimating which dependency is most suited for the current query. To rewrite a failing query, the approach considers the attributes that are involved both in the query and in the selected RFDs, and then it leverages the RFD thresholds to relax the query constraint.

2.3.5 *Data Integration*

Data Integration refers to the process of matching data from different sources or formats to create a reconciled and coherent view of information. Since metadata can provide characteristic signatures, well-known schema matching tools such as COMA [66, 128] and SEMINT [120] rely on statistics and dependencies to identify correspondences. Over the years, researchers have increasingly relied upon dependencies to introduce novel approaches. Wang et al. [179] introduce Probabilistic Functional

Dependencies (PFDs), a specific type of RFD [46], and leverage them to perform data integration in large-scale scenarios such as web data sources. In [16], the authors propose to extend ontology-based mediation architectures with FDs, showing how they can facilitate query reconciliation across heterogeneous sources, even when no common identifiers are available.

2.3.6 *Machine Learning*

Dependencies have also been used for different tasks in machine learning contexts. For instance, in [34] the authors leverage RFDs to derive new features from the original ones, consequently improving prediction scores. On the other hand, in [114] the authors leverage FDs to address the problem of unbalanced datasets. Specifically, they first introduce a measure to assess the distance between two FD sets. Then, they propose an undersampling strategy that selects the instances of the majority class to maximize the distance from the minority class in terms of FDs. Recently, Le Guilly et al. [115] leveraged RFDs to formally establish an upper bound on the accuracy achievable by any classifier over a given dataset, thereby providing interpretable guarantees and helping domain experts assess the feasibility of applying machine learning models in the first place. Recent works also explored the use of dependencies for evaluating the presence of biases within datasets. In [12], the authors use AFDs to identify cases in which attributes such as gender, ethnicity, or religion determine other ones in the dataset. Finally, Chen et al. [51] leveraged FDs in their data augmentation framework, which is characterized by a Generative Adversarial Network (GAN) architecture. They encoded the constraints defined by FDs into the loss function to ensure that the synthetically generated data complies with them.

2.4 DEPENDENCY DISCOVERY: LITERATURE REVIEW

Initially, the challenge of identifying data dependencies was addressed through the automatic discovery of canonical FDs. The algorithms developed for discovering FDs introduced fundamental concepts and key strategies to tackle this complex problem, which becomes increasingly difficult with a growing number

of attributes. Remarkably, certain approaches in the literature have demonstrated efficient execution times even when dealing with datasets containing over 100 attributes [144]. There are two primary categories of methods for the automated discovery of FDs from data: column-based and row-based approaches. Column-based approaches represent the search space as an attribute lattice, allowing for the consideration of candidate FDs at each lattice level in terms of edges. These candidate dependencies undergo validation, which involves the assessment of value combinations or efficient representations of FDs. Notable examples of column-based algorithms include the algorithms TANE [101], FD_Mine [188], FUN [140], and DFD [3]. The first three are algorithms based on similar concepts, while the last is a more recent approach that is significantly more efficient on large real datasets. DFD leverages aggressive pruning through a random-walk depth-first traversal and an efficient result verification mechanism, allowing it to disregard numerous false positives. Instead, row-based approaches derive candidate FDs by analyzing the cross product between all possible combinations of tuple pairs, aiming to create two attribute subsets, namely *agree-set* and *difference-set*. The latter contains attribute sets that agree on the values of certain tuple pairs, since they can functionally determine other attributes agreeing on the same tuple pairs. Once all the agree sets are calculated, it is possible to derive all valid FDs from them. Examples of row-based algorithms include the algorithms FDep [78], DepMiner [126], and FastFD [186]. In [78], the authors approached the challenge of discovering database dependencies from data through the lens of machine learning. By framing the problem as an induction challenge, they harnessed the power of generality ordering, a fundamental concept in computational induction. In their work, the authors tested three fundamental approaches: top-down, bottom-up, and bi-directional. Among these, the bottom-up algorithm emerged as the most efficient and demonstrated its superiority over contemporaneous algorithms. DepMiner [126] combines the discovery of FDs with the construction of real-world Armstrong relations, while FastFD [186] draws inspiration from Artificial Intelligence by opting for a depth-first, heuristic-driven search strategy.

In recent years, the Data Profiling community started to deal with the RFD discovery problem, which is even more complex due to the necessity of considering a generalized property on data, which permits the restriction of the validity of a dependency over subsets of data, i.e., *relaxation on the extent* (RFD$_e$), and/or to compare data in terms of their similarity rather than the simple equality, i.e., *relaxation on the attribute comparison* (RFD$_c$). Although recent works have surveyed different types of RFDs [46, 169], very few of them are equipped with algorithms for discovering RFDs from data [122]. Among the approaches for discovering RFD$_e$s, the algorithm proposed in [54] generates RFD$_e$ candidates by exploiting the attribute lattice derived from the partitions of attribute values. Alternatively, the greedy algorithm proposed in [87] tries to derive valid RFDs by finding a close-to-optimal tableau, where the closeness is measured by using support and confidence measures. Indeed, the method proposed in [107] exploits the $g3$-error measure of super keys to determine the satisfiability degree in order to tolerate possible errors in the validity of FDs. In [75], the authors adapted three algorithms previously introduced for FD discovery, i.e., TANE, FD_Mine, and FastFD, to tackle the problem of discovering RFD$_e$s. The RFDs relaxing only on the attribute comparison (i.e. RFD$_c$s) differ in the way they manage similarity predicates to accommodate possible errors over data, and/or different representations in unreliable data sources [74]. Thus, discovery algorithms have to evaluate the validity of dependency by means of an approximate matching paradigm, which can be defined as similarity/difference constraints, which are composed of similarity/difference functions, operators, and thresholds. The latter could be set as input parameters, or they are automatically inferred from data [45, 49]. One of the most recent discovery algorithms for this class of RFDs evaluates the utility of a candidate dependency for a given database instance, determining the corresponding similarity threshold pattern [168]. The utility is measured by means of support and confidence, whereas thresholds are determined by analyzing the statistical distribution of data. DOMINO is a recent proposal for the discovery of RFD$_c$s [45]. It is able to infer RFDs together with difference thresholds by exploiting the concept of multi-attribute dominance. Instead, another discovery algorithm for the same class of RFDs is de-

fined in [112], which tries to reduce the problem search space by assuming a user-specified distance threshold as an upper limit for the distance intervals of LHS. The algorithm is based on a distance-based subspace clustering model and exploits pruning strategies to efficiently discover RFDs when high threshold values are specified. A recent approach is FastDD [111], an algorithm for discovering Differential Dependencies (DDs) by leveraging a novel structure named "diff-set". As opposed to these discovery algorithms, which focus on a specific class of RFDs, the discovery algorithm proposed in [49] aims to identify RFDs relaxing on both the extent and the attribute comparison, namely *hybrid* RFDs [46]. In particular, the approach relies on a column-based algorithm conceived for FD discovery, which is fed with similar subsets of tuples derived from previously computed differential matrices. In this work, the authors also demonstrate the NP-hardness of the RFD discovery problem for hybrid RFDs, yielding the necessity of designing approximate solutions. Another type of dependency is represented by temporal dependencies, which describe properties by including temporal constraints. Combi et al. [58], formalize the temporal constraint for both FDs and RFDs, resulting in the definition of *Temporal Functional Dependencies* (TFDs) and Approximate Temporal Functional Dependencies (ATFDs). The authors also propose two strategies for mining such properties from data.

All the discovery algorithms presented above need to process the whole dataset, and they need to be re-executed from scratch whenever the datasets are updated. To avoid this, new incremental algorithms for discovering FDs and RFDs have been defined, aiming at updating metadata upon data changes [180]. In [36], authors propose a new methodology capable of updating the set of holding FDs upon data insertions, which has been successively improved in [37] with a new efficient validation method for FDs that relies on regular expressions. DynFD [163], instead, combines row-based and column-based strategies and continuously updates the set of FDs holding on data downstream of batches of inserts, updates, and deletions of data. One of the most recent FD discovery algorithms is DynASt [21], which focuses on the efficient maintenance of agree-sets, which represent the main

bottleneck of row-based discovery strategies. As regards the discovery of relaxed dependencies, some proposals for RFD$_e$s include AD-Miner [73], which relies on logical operations to lower the computational complexity, and BIRD [35], which exploits effective data structures and an efficient execution strategy that allows splitting the discovery process into a level-wise parallel execution. For RFD$_c$s, a recent incremental approach is INDIBITS [28], which combines an innovative binary representation for mapping data similarities with a new validation method for empowering the dynamic update of the set of holding RFD$_c$s over time.

All the described profiling strategies allow to extract a wide range of metadata from datasets in both static and dynamic scenarios. However, all these algorithms apply the same implementations of the comparison criteria, which entails the use of the absolute difference for numerical values and syntactic distances for textual ones. This can represent an issue when dealing with long textual attributes that require semantic-aware comparisons, as we will discuss in detail in the next Chapter (see Section 3.1.1). Moreover, an open problem in the data profiling community is to devise effective visualization techniques for better analyzing the discovery results [1]. Thus, recent works began to explore this research direction. A first proposal for visualizing large sets of RFDs is described in [48]. It presents several metaphors for representing RFDs at different levels of detail. Starting from a high-level visualization of attribute correlations, details are interactively revealed, also including information on the relaxation criteria. Moreover, among recent proposals, two of the most effective platforms for data profiling are the Metanome project [143] and StradyVar [27]. The former embeds several algorithms to automatically discover complex metadata and various result management techniques, such as interactive diagrams of discovery results, while the latter allows both monitoring RFD discovery algorithms during their executions and performing several ad-hoc analyses on the resulting metadata. Despite the recent introduction of the described tools and platforms, none of these tools provides targeted explanations to support users and domain experts in the analysis of the evolution of metadata. For instance, Metanome is mainly focused on the execution of discovery algorithms, providing a

visualization of extracted metadata, while StradyVar is focused on providing statistics on the discovered RFDs without analyzing reasons behind their evolution. Moreover, none of these works provide a way to explain dependencies to expert and/or non-expert users, which may hinder the effective analysis of RFDs in real-world scenarios. We will discuss this research gap in detail in the next Chapter (see Section 3.1.2).

3

DATA PROFILING: FROM THEORY TO REAL WORLD SCENARIOS

In the transition from theory to practice, the application of Data Profiling techniques encounters a number of challenges that limit their effective adoption in real-world scenarios. These open challenges slow down the translation of scientific advances into tools that are genuinely useful for practitioners and organizations. This chapter will first discuss current research gaps, focusing on those regarding (relaxed) functional dependencies (Section 3.1), then it will present two proposals aimed at addressing such problems (Sections 3.2 and 3.3).

3.1 OPEN CHALLENGES AND RESEARCH GAPS

While the efforts of the research community in the definition of novel and efficient *Data Profiling* techniques have led to a plethora of available discovery approaches (detailed in Section 2.4), several research gaps still exist. In this section, we discuss two challenges that need to be addressed in the context of Data Profiling. Specifically, Section 3.1.1 will discuss the problem of employing semantic-aware distances within discovery processes, while Section 3.1.2 will discuss the problem of effectively visualizing, summarizing and explaining discovery results.

3.1.1 *Challenge 1: Semantic-aware discovery algorithms*

The Data Profiling community has devoted its efforts to properly and efficiently managing searching and validation strategies to accommodate relaxation criteria on the discovery problem by obtaining meaningful results, but little effort has been devoted to enriching the set of possible implementations of relaxation criteria. In this direction, introducing different implementations could improve the applicability of discovery algorithms across heterogeneous sets of data, thereby enabling the discovery of patterns and

properties also in such scenarios. For example, one of the most relevant challenges in the field of Data Profiling consists in dealing with datasets that include unstructured or semi-structured attributes such as long textual fields (e.g., comments, reviews, descriptions), which is common in real-world scenarios. As a matter of fact, current discovery algorithms only rely on syntactic distance measures (e.g., edit distance, Jaro distance) to assess the similarity/distance among textual values during the discovery process, failing to capture semantic similarities. This inability to properly assess similarities hinders the extraction of meaningful dependencies, as it easily invalidates the latter. To this end, to effectively extract a set of meaningful dependencies in such scenarios, it is necessary to extend the possible implementations of relaxation criteria by applying semantic-aware comparisons, enabling the evaluation of similarities between long, heterogeneous textual values. This problem will be addressed in the work presented in Section 3.2.

3.1.2 *Challenge 2: Visualizing and Explaining discovery results*

Another important challenge highlighted in recent studies refers to the visualization and interpretation of profiling results. As a matter of fact, a given dataset might contain thousands of syntactically valid dependencies [1]. As a consequence, raw profiling results cannot be directly shown to end users. Instead, there is a need to introduce visualization techniques to intuitively show and summarize discovery results. Moreover, as dependencies are difficult to interpret for non-expert users, it is important to introduce methods to provide textual explanations. While recently introduced data profiling platforms [27, 48, 143] focused on providing interactive diagrams of discovery results and/or performing several ad-hoc analyses on the resulting metadata, none of them provides targeted explanations to support users and domain experts in the analysis of metadata. In this direction, the introduction of Large Language Models (LLMs) represents a promising research direction to provide context-aware, natural-language explanations of discovery results. Such capabilities would also be valuable in dynamic contexts, allowing for ex-

planations of how and why metadata evolves over time. This problem will be addressed in the work discussed in Section 3.3.

3.2 SOLVING CHALLENGE 1: S-DIMε [42]

As discussed in Section 3.1, current RFD discovery algorithms only leverage syntactic functions to evaluate distances/similarities among textual values. However, these kinds of functions are often unable to effectively capture the latter, leading to the extraction of RFDs that may not reflect the actual properties of the data at hand. In this scenario, the extraction of meaningful RFDs requires a sentence-level semantic similarity strategy, which would improve the overall applicability and effectiveness of discovery processes. To address this problem, we proposed an extension of the DIMε [49] algorithm, aiming to enable semantic comparisons between textual attributes by considering their vector embeddings. The description and the evaluation of this new algorithm, namely S-DIMε (**Semantic-Difference Matrix and ε-threshold**), will be detailed in the following sections (i.e., Section 3.2.1 and 3.2.2, respectively).

3.2.1 *The* S-DIMε *algorithm*

Figure 3.1 provides an overview of S-DIMε. As can be seen, it is composed of two main steps, namely (i) *Preprocessing* and (ii) *Discovery*. The former starts from a set of user-specified distance constraints (Φ) and a set of data, and is responsible for creating the structures required by the discovery step, i.e., the *Difference Matrices* and the *Stripped Similarity Subsets*, through the computation of the differences between the values for each pair of tuples. After the preprocessing step, S-DIMε starts by defining the set of initial candidate RFDs to be validated, and performs a level-by-level generation of candidate RFDs according to the column-based Apriori strategy defined in [49], which models the search space considering a graph representation of a lattice. The latter contains different levels with the property that each level L_i contains all the combinations of attributes with size i. Each node in the lattice represents a unique set of attributes, which is linked to nodes that contain a direct superset (at the highest level) or a subset of

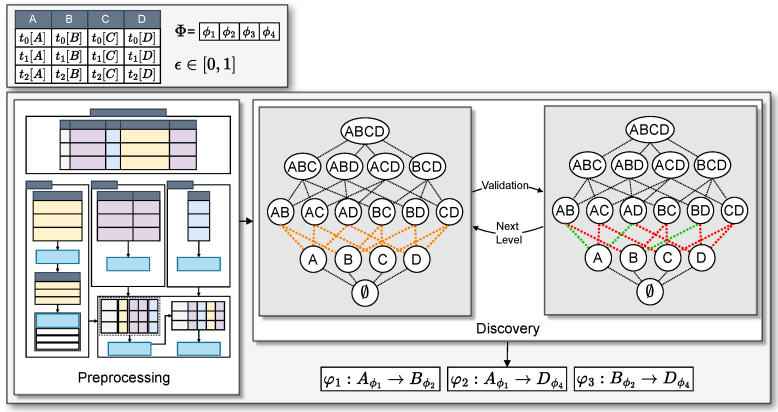


Figure 3.1: Overview of the S-DIMε algorithm.. The algorithm is characterized by two different steps, namely *Preprocessing* and *Discovery*. The former is depicted in detail in Figure 3.2.

attributes (at the lowest level), as can be seen in Figure 3.1. In this type of structure, each edge represents the inclusion relation between two different attribute sets, allowing for representing the LHS and the RHS of an RFD [144]. According to this strategy, S-DIMε generates candidate RFDs by first defining attribute sets X, and then all the possible RFDs $X \backslash A \rightarrow A$, with $A \in X$, to be validated. This strategy allows for efficiently analyzing the search space, reduces the number of RFDs to be validated, and extract the set of holding RFDs for the considered set of data. In what follows, the two steps underlying S-DIMε will be described in detail.

3.2.1.1 *Preprocessing*

The preprocessing step of S-DIMε consists of two main phases: *i*) the generation of a *Difference Matrix* and *ii*) the creation of *Stripped similarity subsets*. Before detailing the novel sentence-level semantic approach of S-DIMε, in what follows, we will first provide notions and examples regarding these two structures and how they are generated in traditional RFD discovery approaches.

The *Difference Matrix* is a structure where each row represents a comparison between two tuples of the dataset. The columns repre-

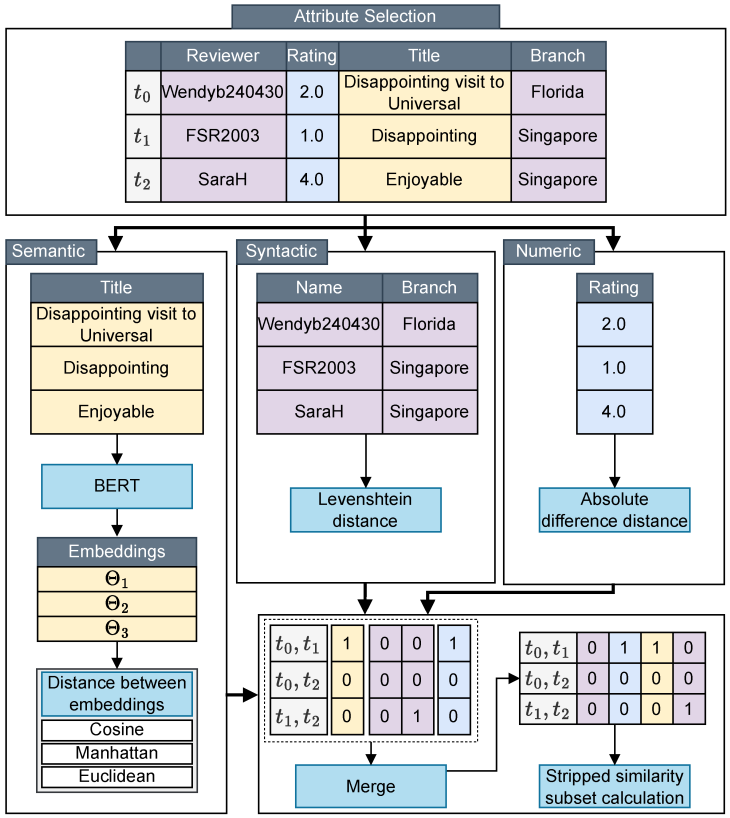


Figure 3.2: Overview of the preprocessing module.

sent the dataset's attributes, and each cell contains the difference among the considered values according to a distance function, which is selected according to the nature of the attribute. The *Difference Matrix* can be formalized as follows:

Definition 8 (Difference matrix). *Let r be an instance of a relation schema R, A an attribute of R, and δ a distance function defined on the domain of A. The* difference matrix *for A is a matrix M_A whose entry (i,j) contains the distance value $\delta(t_i[A], t_j[A])$ of the projections of tuples t_i and t_j on A, according to a distance function δ.*

Two values $t_i[A]$ and $t_j[A]$ are similar if their distance is less or equal to a given threshold ϵ defined for the attribute A.

	Reviewer	Rating	Title	Branch
t_0	Wendyb240430	2.0	Disappointing visit to Universal	Florida
t_1	YahooHK	3.0	Fun but avoid holiday crowds	Singapore
t_2	TeenoLol	5.0	Enjoyable day	Singapore
t_3	SaraH	4.0	Enjoyable	Singapore
t_4	NerdyP	2.0	Disappointed	Florida
t_5	Sunandra K	3.0	Don't go during holiday	Singapore
t_6	FSR2003	1.0	Disappointing	Florida
t_7	Nolaochan	5.0	Like a Dream	Japan

Table 3.1: A snippet of the Universal Studios reviews dataset.

	0	1	2	3	4	5	6	7
0	0	1	3	2	0	1	1	3
1	1	0	2	1	1	0	2	2
2	3	2	0	1	3	2	4	0
3	2	1	1	0	2	1	3	1
4	0	1	3	2	0	1	1	3
5	1	0	2	1	1	0	2	2
6	1	2	4	3	1	2	0	4
7	3	2	0	1	3	2	4	0

Table 3.2: $M_{\mathsf{Rating}_{abs}}$

Example 3.2.1. Consider the sample dataset shown in Table 3.1 and a set of thresholds $E = [2.0, 1.0, 4.0, 1.0]$ included into the distance constraints Φ and defined over its attributes, i.e., Reviewer, Rating, Title, and Branch. Suppose that we need to generate the *Difference Matrix* for the attribute Rating. Since the latter is a numerical attribute, we use the *absolute difference* as a distance measure. The resulting *Difference Matrix* is shown in Table 3.2. By considering the threshold defined in E for Rating (i.e., 1.0), we can then assess which values are similar. For instance, we can see that the value of t_0[Rating] is similar to the ones of t_1, t_4, t_5, and t_6.

	0	1	2	3	4	5	6	7
0	0, 0	1, 28	3, 28	2, 29	0, 21	1, 24	1, 19	3, 27
1	1, 28	0, 0	2, 22	1, 25	1, 24	0, 19	2, 25	2, 24
2	3, 28	2, 22	0, 0	1, 4	3, 13	2, 17	4, 13	0, 11
3	2, 29	1, 25	1, 4	0, 0	2, 11	1, 20	3, 12	1, 10
4	0, 21	1, 24	3, 13	2, 11	0, 0	1, 18	1, 3	3, 11
5	1, 24	0, 19	2, 17	1, 20	1, 18	0, 0	2, 18	2, 18
6	1, 19	2, 25	4, 13	3, 12	1, 3	2, 18	0, 0	4, 12
7	3, 27	2, 24	0, 11	1, 10	3, 11	2, 18	4, 12	0, 0

Table 3.3: $M_{\mathsf{Rating}_{abs},\mathsf{Title}_{lev}}$

The definition of *Difference Matrix* can be easily generalized to attribute sets, in which an entry will contain an n-tuple of distance values, one for each attribute in the set. The *Difference Matrix* for an attribute set X can be derived from the difference matrices for the single attributes composing it, by concatenating elements with the same coordinates. An example is provided in Table 3.3, in which we consider the attributes Rating and Title. Notice that, since the latter is a textual attribute, we use the *Levenshtein* distance to compare the values.

Starting from the difference matrices, it is possible to generate the *similar pattern* of a tuple t (a matrix row), which represents sets of tuples (or matrix columns) satisfying a given constraint with respect to t.

Definition 9 (Similar pattern). *Let r be an instance of a relation schema R, $X = \{A_1, \ldots, A_n\}$ an attribute set of R, M_{X_Δ} a Difference Matrix for X, and $\Phi = (\phi_1, \ldots, \phi_n)$ a sequence of constraints on the values of M_{X_Δ}. A similar pattern of tuple t_i of M_{X_Δ}, denoted as $\tau_X^{t_i}$, is the sequence $(j_1, \ldots, j_h)$ with $h \leq |r|$ and $M[i, j_k] = (d_1, \ldots, d_n)$, where d_q satisfies the constraints ϕ_q $\forall q \in [1, n]$ and $\forall k \in [1, h]$.*

Successively, the tuples sharing the same similar patterns can be grouped into *similar pattern subsets*:

Definition 10 (Similar pattern subsets). *A* similar pattern subset *S_X for X is defined as $S_X = \{j_1, \ldots, j_h\}_{\{i_1,\ldots,i_k\}}$ with $1 \leq k \leq h \leq |r|$, $\tau_X^{i_p} = (j_1, \ldots, j_h)$ $\forall p \in [1, k]$ and $\tau_X^{i_v} \neq (j_1, \ldots, j_h)$ $\forall v \notin [1, k]$. The set of different* similar pattern subsets *for X is denoted as I_X.*

Example 3.2.2. Let us consider the dataset sample of Table 3.1, and suppose the user specifies constraints based on the *abs* function, the $\leq$ comparison operator, 1 as a threshold for the attribute Rating, and the *levenshtein* function, the $\leq$ comparison operator, 4 as a threshold for the attribute Title, then the following sets of similar pattern subsets are generated:

- $I_{\text{Rating}} = \{\{0,1,4,5,6\}_{\{0,4\}}, \{0,1,3,4,5\}_{\{1,5\}}, \{2,3,7\}_{\{2,7\}}, \{1,2,3,5,7\}_{\{3\}}\{0,4,6\}_{\{6\}}\}$;
- $I_{\text{Title}} = \{\{0\}_{\{0\}}, \{1\}_{\{1\}}, \{2,3\}_{\{2,3\}}, \{4,6\}_{\{4,6\}}, \{5\}_{\{5\}}, \{7\}_{\{7\}}\}$;
- $I_{\{\text{Rating},\text{Title}\}} = \{\{0\}_{\{0\}}, \{1\}_{\{1\}}, \{2,3\}_{\{2,3\}}, \{4,6\}_{\{4,6\}}, \{5\}_{\{5\}}, \{7\}_{\{7\}}\}$;

The generated similar patterns will be then provided to the Discovery step, which will leverage the latter for validating RFDs (see Section 3.2.1.2).

Current RFD discovery algorithms only generate similar patterns by using syntactic distance functions such as the *levenshtein* distance when dealing with textual values. This aspect represents a limitation, since in datasets with large textual values, it is unlikely to appropriately capture similarities. This can already be noticed in the example carried throughout this section. Even if, for the sake of simplicity, the dataset snippet of Table 3.1 contains only the (short) review titles and not the reviews themselves, with the *levenshtein* distance, it is already difficult to capture all similarities. For instance, the value t_0[Title] is clearly similar to the ones of t_4 and t_6, but the distance values provided by the distance function are high, thereby exceeding the similarity threshold defined by the user.

In general, the inability to properly assess similarities can negatively affect the validation of a given RFD when a long textual attribute appears on the LHS or the RHS. In the first case, during the validation, tuples that are actually referring to the same concept may be judged as distant. As a result, it is likely that the set of attributes on the LHS represents a key (i.e., a combination of columns for which all tuples are different), leading to a *key dependency*, which in several cases is considered trivial and not meaningful. On the other hand, the presence of a long textual attribute on the RHS of a given RFD can lead to the invalidation of the latter. In particular, if two tuples are considered similar on the LHS, it is likely that they will result distant on the RHS. As

a straightforward strategy to avoid this problem, it might seem sufficient to increase the similarity thresholds, thereby relaxing the maximum allowed distance for considering two values similar. However, this adjustment would inevitably cause unrelated textual values to be treated as similar, which reduces the meaningfulness of the discovered RFDs. Therefore, a distance function that properly accounts for semantic similarities is required.

For this reason, with S-DIMε, we extend the DIMε discovery algorithm by introducing a novel preprocessing step (depicted in Figure 3.2) that enables users to apply semantic comparisons among the selected textual attributes. These attributes are extracted and sent to a dedicated module (i.e., the *Semantic Module*), while the others are processed according to the traditional preprocessing strategy. The semantic module allows the generation of the similarity matrix for the subset of attributes it receives as input. Its work is based on the use of sentence embedding models, i.e., models capable of representing entire sentences as vectors of real numbers. In our work, we relied on the *Sentence Transformers* approach, for which there exist a wide range of pre-trained embedding models[1]. Among these, we selected *distiluse-base-multilingual-cased-v2*, a multilingual model based on Sentence-BERT (SBERT) [155]. SBERT is a modified version of the original Google BERT (Bidirectional Encoder Representation from Transformer) [63]. It adds a pooling operation to the output of BERT and uses siamese and triplet network structures to derive meaningful sentence embeddings. The model we selected is able to use the Knowledge Distillation [156] process to support more than 50 languages while maintaining good performance. It is important to notice that our approach can be easily generalized to any other state-of-the-art model, by simply invoking a different model. For S-DIMε, we leverage a general-purpose model, since we aim to show the generalizability of our approach to a wide range of application domains. Nevertheless, it could be possible to leverage a fine-tuned embedding model for each specific context to further enhance the quality of the embeddings.

The semantic module considers one attribute at a time and for each one generates a binary column that will be part of the

1 https://www.sbert.net/docs/pretrained_models.html

similarity matrix to be given as output. To generate a column associated with an attribute X, the algorithm exploits the embedding model to generate the vectors associated with each value taken by X. Subsequently, it calculates the pairwise-distances between the vectors to assess the closeness between each pair of values. This can be accomplished by leveraging different distance measures. For S-DiMε, we enabled the possibility to choose between three measures: *Cosine*, *Euclidean*, and *Manhattan* distances.

Definition 11 (Cosine distance). *The Cosine Similarity of two vectors is represented by the cosine of the angle between them, determining whether they are pointed in a similar direction or not. Since the Cosine similarity is a value in the range* $[0, 1]$*, the Cosine distance can be obtained by simply subtracting the cosine similarity from 1. Therefore, given two vectors A and B of length n, the cosine distance is defined as follows:*

$$1 - \cos(A, B) = 1 - \frac{A \cdot B}{\|A\| \cdot \|B\|} = 1 - \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} (a_i)^2} \sqrt{\sum_{i=1}^{n} (b_i)^2}}$$

where $a_i (b_i, resp.)$ *is the i-th component of vector* $A(B, resp.)$*.*

Definition 12 (Minkowski distance). *The Minkowski distance is the generalization of several distance metrics such as Euclidean distance, Manhattan distance, and Chebyshev distance. Formally, given two vectors A and B of length n, it is defined as follows:*

$$D(A, B) = (\sum_{i=1}^{n} |a_i - b_i|^p)^{\frac{1}{p}}$$

When $p = 1$*, the Minkowski distance is equivalent to the Manhattan distance, whereas when* $p = 2$ *it is equivalent to the Euclidean distance.*

We selected the cosine distance because it measures the angular distance between vectors, focusing on their orientation rather than their magnitude. This property is particularly suitable for semantically oriented embeddings, where the direction of the vector encodes meaning more effectively than its length. On the other hand, the Euclidean and Manhattan distances quantify absolute differences between embedding components and provide complementary perspectives on spatial separation.

After applying one of the previously defined distances, S-DIMε considers the distance threshold set for X and generates a binary column, where, for each pair $t_i[X], t_j[X]$, 1 indicates that their values are similar (i.e., the distance is below the threshold) and 0 indicates that their values are not similar (i.e., their distance exceeds the threshold). After all columns are calculated, they are merged, forming a *Similarity Matrix*. As for the other attributes that were not selected for the semantic comparison, these are processed in a traditional manner, by applying the *Levenshtein* distance for textual attributes and the *absolute difference* for numeric values. Thus, after evaluating all the dataset's attributes, three binary similarity matrices are obtained: one for the semantic attributes, one for the syntactic attributes, and another one for the numeric attributes of the dataset. These three matrices are merged according to the original attribute ordering, obtaining the final similarity matrix.

Example 3.2.3. Consider Figure 3.2, which shows an example related to a sample of the *Universal* dataset, and let us assume that the user has selected the Title attribute as the one that has to be processed by taking into account semantic similarities. Thus, the Title column is given as input to the semantic module, which will compute the embeddings with SBERT and then perform a comparison between the resulting vectors according to one of the three available distance measures. In this example, we can see that t_0[Title] is similar to t_1[Title]. As for the other attributes (i.e., Reviewer, Rating, and Branch), these are processed with the traditional distance functions, i.e., *levenshtein* distance for textual values and *absolute difference* for numbers. After obtaining the matrices for all attributes, these are merged together according to the original attribute ordering. Finally, the resulting matrix is leveraged to compute the *Similar Pattern Subsets* that will be given as input to the discovery step.

3.2.1.2 *Discovery*

The validation process is a fundamental task in discovery algorithms, since it makes it possible to identify RFDs holding on a dataset and apply search space pruning strategies according to the newly validated RFDs. This process must be able to efficiently

validate RFDs and avoid errors in the validation process. To this end, in this section, we will discuss the validation processes underlying S-DIMε.

The validation of candidate RFDs entails verifying that, whenever two tuples are similar on the LHS attributes, they are also similar on the RHS one for a significant portion of the database, where *significant portion* is quantified by a coverage measure. In order to validate candidate RFDs, S-DIMε reduces the number of *Similar Pattern Subsets* (denoted by $\hat{I}$) for the attributes of the LHS, by eliminating singletons (since they correspond to the matrix diagonal entries that trivially compare each tuple with itself), obtaining the *Stripped Similar Pattern Subsets*. The latter can be used to validate candidate RFDs based on the concept of *similar pattern subset refinement*.

Definition 13 (Refinement property). *A set $\hat{I}_X$ is said to* refine *another set $\hat{I}_{X\cup A}$ if every similar pattern subset in $\hat{I}_X$ is contained in one of the subsets in $\hat{I}_{X\cup A}$.*

Based on the refinement notion, we derive the following lemma for RFDs.

Lemma 3.2.1. A relaxed functional dependency $X_{\Phi_1} \overset{\Psi_{err(0)}}{\longrightarrow} A_{\Phi_2}$ holds on an entire database instance if and only if $\hat{I}_X$ *refines* $\hat{I}_A$.

Example 3.2.4. Let us consider the database instance of Table 3.1. If we set the constraints as defined in Example 3.2.2, then according to Lemma 3.2.1 the following RFD is valid:

$$\varphi : \mathsf{Title}_{\leq 4} \overset{\Psi_{err(0)}}{\longrightarrow} \mathsf{Rating}_{\leq 1}$$

since each similar pattern in $\hat{I}_{\mathsf{Title,\ Rating}}$ is included in a similar pattern of $\hat{I}_{\mathsf{Rating}}$:

- $\hat{I}_{\mathsf{Title,Rating}} = \{\{2,3\}_{\{2,3\}}, \{4,6\}_{\{4,6\}}\}$,
- $\hat{I}_{\mathsf{Rating}} = \{\{0,1,4,5,6\}_{\{0,4\}}, \{0,1,3,4,5\}_{\{1,5\}}, \{2,3,7\}_{\{2,7\}}, \{1,2,3,5,7\}_{\{3\}}, \{0,4,6\}_{\{6\}}\}$,

In case of hybrid RFDs (i.e., RFDs that relax on both the extent and comparison criteria), S-DIMε should also consider as valid the RFDs holding on a subset of tuples, according to a coverage

measure and a user-defined threshold. In this case, to accomplish the validation process, it is necessary to calculate the satisfiability degree of the instance according to a coverage measure. S-DIMε relies on the *g*3-*error* coverage measure [101], but it can also work with other coverage measures by replacing the function computing the measure from similar subsets. The computation of the *g*3-*error* for RFDs with hybrid relaxation is an NP-complete problem [167], since the MINIMUM VERTEX COVER problem can be reduced to it. A solution to this problem can be found in polynomial time in the case of disjoint similar pattern subsets, in which case, we can use the same strategy of AFD discovery [101]. However, since in the case of intersecting similar pattern subsets each of these tuples can fall in more than one similar pattern subset of I_X, it is not possible to perform a computation local to each similar pattern subset, but a more global analysis is required. Thus, we provide a greedy solution for calculating the *g*3-*error* in polynomial time [102].

3.2.2 *Evaluation*

In this section, we present the experimental evaluation performed to evaluate S-DIMε. In all the experiments, we used the original DIMε algorithm as baseline approach, allowing to properly evaluate S-DIMε, since the latter is mainly based on it. In the experiments, we first evaluated the impact of the novel semantic

ID	Name	Columns	Rows
Universal	Reviews of Universal Studios	6	2,000
Food	Amazon Fine Food reviews	7	6,000
Kindle	Amazon reviews: Kindle Store Category	7	7,000
German	Dataset German news	9	2,000
Drug	Drug Review Dataset	9	3,096
Gamestop	Gamestop products reviews	10	2,367
Hotel	Hotel reviews	13	5,000
Amazon	Amazon Walmart	14	4,500

Table 3.4: Overview of the datasets adopted in the experimental evaluation of S-DIMε.

module on both the execution time and the memory consumption. Then, we performed a quantitative evaluation to analyze the differences in terms of RFDs discovered by the S-DIMε and DIMε. Finally, to better assess the effectiveness of the latter, we performed a qualitative evaluation to verify if the novel RFDs involving textual attributes discovered by S-DIMε are meaningful.

For the experimental evaluation, we considered 8 datasets that include, among their attributes, columns containing more elaborate sentences or even whole texts such as long reviews. The main characteristics of the considered datasets are shown in Table 3.4. It is worth pointing out that the datasets were preprocessed in order to eliminate instances with missing values, as they negatively effect the discovered RFDs and we wanted to avoid disturbing factors external to our assessment. In addition, we removed most of the attributes that were not relevant for our analysis, such as *IDs*, timestamps, and coordinates that uniquely identified the datasets' tuples. For each dataset, we launched S-DIMε and DIMε to discover hybrid RFDs with different coverage measure thresholds (i.e., 0.2, 0.4, 0.6, and 0.8). In addition, we also performed the discovery on RFDs relaxing only on the comparison criterion (i.e. RFD_cs). As concerns the thresholds used for the comparison criteria, we tested three different configurations (namely C_1, C_2 and C_3). Finally, for the semantic module of S-DIMε, we evaluated 3 different distance measures (i.e., *cosine*, *manhattan*, and *euclidean*). All the experiments have been executed on an iMac Pro with an Intel Xeon CPU at 3.20 GHz, 18-core, and 128GB of memory, running macOS Mojave, Python 3.8, and OpenJDK 17 as Java environment.

3.2.2.1 *Performances analysis*

The first experimental evaluation aims to analyze the performance of the preprocessing step of S-DIMε in terms of time and memory using different semantic distances. These performances have also been compared with those achieved by DIMε, which adopts only traditional syntactic distances. Figure 3.3 compares the execution times of the preprocessing steps of S-DIMε and DIMε, where the acronyms *cos*, *man*, and *eu* denote the S-DIMε configurations with *cosine*, *manhattan*, and *euclidean* distances.

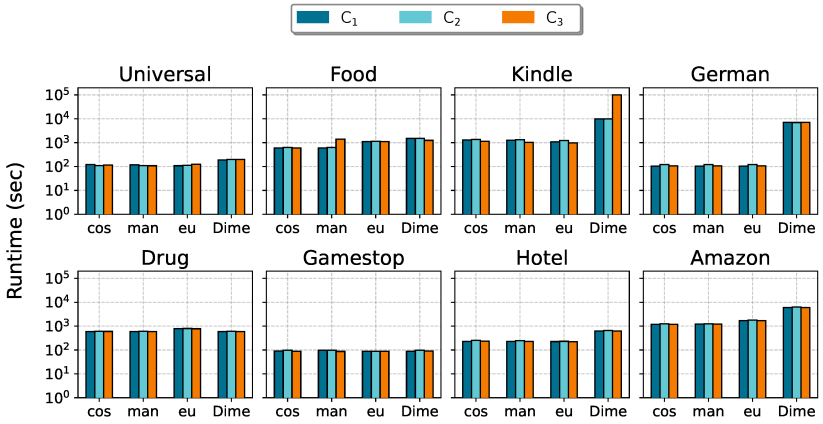


Figure 3.3: Comparison of S-DIMε and DIMε preprocessing times.

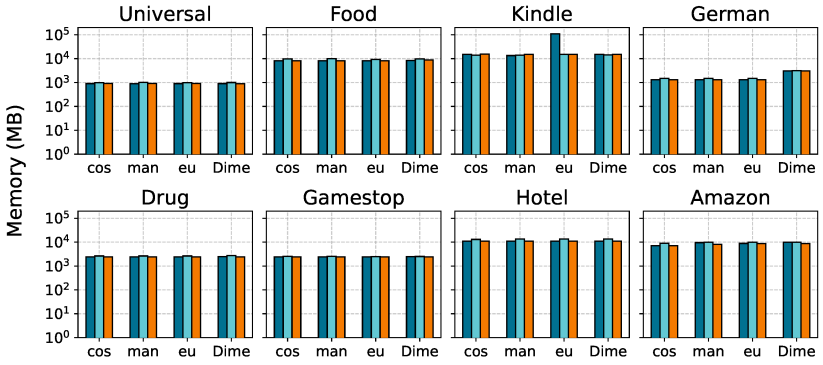


Figure 3.4: Comparison of S-DIMε and DIMε memory consumption.

By comparing the three S-DIMε configurations, we can see that the execution times are quite similar, meaning that the choice of different distance functions to compare the embeddings does not affect efficiency. Nevertheless, the most efficient distance measures was the *cosine* distance, which, on average, improved the execution times of 4.47% with respect to the *manhattan* distance and of 19.41% with respect to the *euclidean* distance. The latter was negatively affected by the higher execution times on the *Amazon* and *Drug* datasets, whereas on the other ones it showed similar execution times. On the other hand, by comparing the S-DIMε configurations with DIMε, it is possible to notice more significant differences. In particular, we can see that S-DIMε is generally

faster, while only in a few cases it was slightly outperformed by DIMε. More specifically, on the *Gamestop* and *Drug* datasets, the three S-DIMε versions and DIMε are generally aligned, with only the *eu* configuration being slower on the *Drug* dataset. Instead, S-DIMε was significantly faster on the other datasets, with average gains ranging from 21.16% to 98.42%. The slowdowns of DIMε are due to the overhead required by the application of the *levenshtein* distance on long texts. This observation is supported by the results recorded with the *German* dataset, which contains the longest textual values with respect to the ones of all other considered datasets. These characteristics challenged the performance of the *levenshtein* distance, leading to a significant slowdown. As a result, the S-DIMε versions significantly outperformed it, with average gains ranging from 98.41% to 98.43%. Moreover, DIMε was significantly slower also for the *Kindle* dataset, which includes the largest number of tuples (and thus, the largest number of comparisons to perform) among the tested datasets. With S-DIMε, we were able to significantly reduce the execution times, with average gains ranging from 96.84% to 97.26%.

We also evaluated the memory consumption of the preprocessing step, comparing it with the ones recorded with DIMε (see Figure 3.4). In general, we can see that all the compared versions exhibit a similar behaviour on all datasets, with the exception of *Kindle* and *German*. In the former dataset, the version of S-DIMε with the *euclidean* distance recorded a higher memory consumption, while on the latter, all three S-DIMε versions outperformed DIMε. This is again probably due to the fact that this dataset is more challenging for the traditional *levenshtein* distance. By considering all experiments, the versions of S-DIMε with the *manhattan* and *cosine* distance improved the overall memory consumption of DIMε of 5.35% and 7.29%, respectively, while the version of S-DIMε with the *euclidean* distance showed a slightly higher consumption with respect to all the compared versions.

3.2.2.2 *Quantitative analysis*

We conducted a quantitative analysis to evaluate the differences among the sets of RFDs discovered by S-DIMε and DIMε. More

specifically, given two sets of RFDs Σ and Σ' discovered by DIMε and S-DIMε, respectively, we considered:

- The percentage of RFDs in Σ' that exactly match the ones in Σ (RFDs *found*);
- The percentage of RFDs in Σ that does not belong in any form to Σ' (RFDs *not found*);
- The percentage of RFDs in Σ' not included in Σ. This means that an RFD in Σ does not exactly match, does not generalize, and does not specialize any RFD in Σ' (*New* RFDs).

Figure 3.5 shows the results achieved by the different variants of S-DIMε and DIMε across all datasets and coverage measure thresholds. In general, it is possible to notice that S-DIMε was able to discover a certain amount of *New* RFDs in most of the configurations. Among the different distance measures compared in our evaluation, the *Manhattan* (denoted with *M* in the figure) and the *Cosine* (denoted with *C* in the figure) distance allowed the discovery of more *New* RFDs with respect to the *Euclidean* (denoted with *E* in the figure) distance. Specifically, by averaging all experiments, the sets of RFDs discovered with version of S-DIMε with the *manhattan* distance included 39.53% *New* RFDs, while the one with the *cosine* distance included 20.1% *New* RFDs and the one with the *euclidean* distance only 7.30%. As a matter of fact, the RFD sets of the latter version were the ones that more closely resembled the RFD sets discovered with DIMε, with an overall percentage of RFDs *Found* of 91.41%. On the other hand, the versions of S-DIMε with the *manhattan* and *cosine* distance recorded an overall RFDs *Found* of 82.63% and 62.91%, respectively.

By considering the different coverage measure thresholds, which increase from 0 (i.e., Figure 3.5(a)) to 0.8 (i.e., Figure 3.5(e)) we also noticed some differences in the discovery results. In general, with RFDs relaxing only on the comparison criterion (i.e., coverage = 0), it was possible to extract on average the highest percentage of *New* RFDs (i.e., 32.67%). By increasing the coverage measure threshold, we noticed that the *New* RFDs tend to gradually decrease, with percentages of 27.6%, 19.2%, 17.45%, 14.53% for coverages $0.2, 0.4, 0.6, 0.8$, respectively. This result can be explained by the fact that, with higher coverage thresholds,

more violations are admitted, allowing the DiMε algorithm to discover more RFDs that involve textual attributes. As a matter a fact, the main reason why traditional discovery algorithms are not able to discover RFDs when dealing with long textual values is that they are not capable of properly assess similarities. Thus, during the discovery process, when trying to validate an RFD it is likely that two tuples that are similar on the LHS of the RFD does not match on the RHS if the latter involve a long textual attribute, leading to a violation. However, if more violations are admitted, it is possible to validate at least a part of these RFDs. This also explains the fact that the number of RFDs *Found* by S-DiMε raises as the coverage measure threshold increases. Nevertheless, even with the highest coverage measure, S-DiMε was still able to find a significant number of *New* RFDs, proving that the semantic-aware strategy is able to capture new relationships. To better analyze the meaningfulness of the novel RFDs discovered by S-DiMε, in the next section we provide a qualitative analysis of the results.

3.2.2.3 *Qualitative evaluation*

As the final part of our evaluation, we analyzed in detail the RFDs discovered with S-DiMε, with the aim of demonstrating its effectiveness in capturing meaningful dependencies. To this end, we analyzed the *New* RFDs (i.e., the RFDs discovered by S-DiMε and not by DiMε), focusing on dependencies that involved at least one of the attributes selected for the semantic comparison. In what follows, we will provide some examples of the novel meaningful RFDs discovered by S-DiMε.

Considering the *Gamestop* dataset, S-DiMε discovered the following RFD:

$$\varphi_1 : \mathsf{Name}_{\leq 2}, \mathsf{Review\ Description}_{\leq 16}, \mathsf{Recommended}_{\leq 2} \xrightarrow{\Psi \leq 0,6} \mathsf{Review\ Title}_{\leq 16}$$

which states that tuples with similar products, recommendations and reviews often have similar review titles. This RFD was not captured by the original DiMε algorithm because of the presence of Review Description and Review Title in the LHS and the RHS, respectively. For example, consider Table 3.5, which contains a snippet extracted from the *Gamestop* dataset. In this

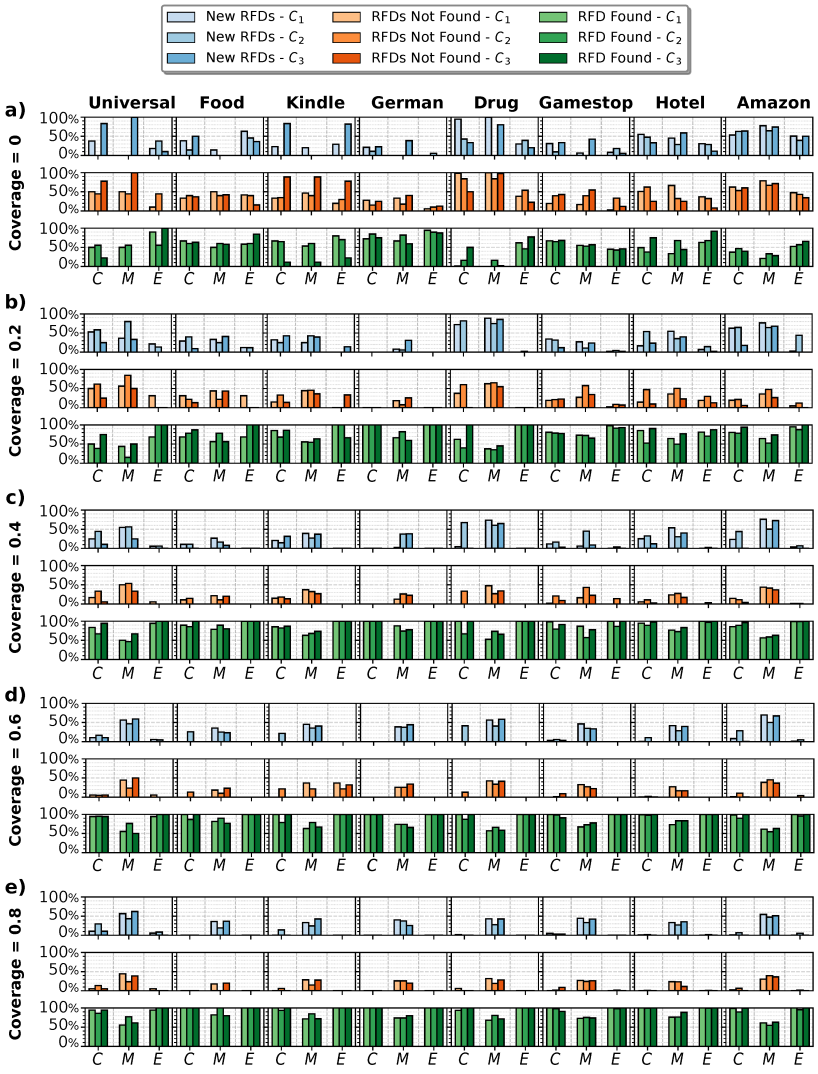


Figure 3.5: Categorization of the RFDs discovered by S-DIMε with different coverage values (0 (a), 0.2 (b), 0.4 (c), 0.6 (d), 0.8 (e)) and different threshold configurations (C_1, C_2, and C_3).

case, even if both the review descriptions and titles show obvious similarities, with the *levenshtein* distance, it was not possible to capture their closeness. For example, by considering the Review Description values of the third and fourth tuples, the *levenshtein* distance between the descriptions is 82, despite both reviews

Name	Review Description	Recommended	Review Title
Microsoft Xbox Elite Series 2 [...]	Great controller love the way it feels and all the little halo touches it has.	Yes: I recommend this product	Awesome Controller
Microsoft Xbox Elite Series 2 [...]	I love the pro controller and this one is absolutely beautiful. Must buy	Yes: I recommend this product	Best Controller
Microsoft Xbox Elite Series 2 [...]	AXYB buttons do no register unless they are 100% fully pressed in. Annoying there is no pressure sensitivity!	No: I don't recommend this product	Buttons don't register
Microsoft Xbox Elite Series 2 [...]	controller is awesome. but my Y button doesn't work all the time. its very frustrating.	No: I don't recommend this product	Y button doesn't work all the time.

Table 3.5: A snippet from the Gamestop dataset.

complain about the controls. As previously discussed, the inability to assess any of the similarities on the LHS of the RFD makes the latter a *key dependency*, which can be considered trivial and thus not meaningful. Moreover, in the unlikely scenario in which it is possible to find a similarity on the LHS, the inability to assess the similarity on the RHS would lead to an invalidation. For instance, even if the titles in the example are rather short, the difference between the values of the third and fourth rows for the attribute Review Title is already 21. Similarly, another new RFD found on the same dataset was the following:

$$\varphi_2 : \text{Review Title}_{\leq 0.2}, \text{Review Description}_{\leq 0.2} \xrightarrow{\Psi_{err(0)}} \text{Recommended}_{\leq 2}$$

which means that similar reviews imply whether a product is recommended or not. This can be observed in the same snippet of the *Gamestop* dataset shown in Table 3.5. In this case, both long textual attributes are on the LHS of the RFD, and by using the *levenshtein* distance, it was not possible to find two tuples with similar values, leading φ_2 to be considered a *key dependency*.

Considering the *Hotel* dataset, S-DIMε discovered the following RFD:

$$\varphi_3 : \text{Address}_{\leq 6}, \text{Ctry}_{\leq 6}, \text{Name}_{\leq 6}, \text{Prov}_{\leq 6}, \xrightarrow{\Psi_{err(0)}} \text{Review Text}_{\leq 0.4}$$

which states that tuples referring to similar hotels in similar places have also similar reviews. φ_3 was not captured by the original DIMε algorithm due to the presence of Review Text in its RHS. For example, consider Table 3.6, which contains a snippet extracted from the *Hotel* dataset. In this case, even if the reviews

Address	Ctry	Name	Prov.	Review Text
1 Wenonah Park Pl	US	Doubletree By Hilton Hotel Bay City - Riverfront	MI	We stay here for a weekend [...] The rooms are clean and quiet, the bed comfortable and the view of the river very nice.
1 Wenonah Park Pl	US	Doubletree By Hilton Hotel Bay City - Riverfront	MI	This is a great place to stay if you want a quiet place with a great view [...]
100 S 4th St	US	Comfort Inn Huntingdon	PA	The train whistle kept me up all night long [...]
100 S 4th St	US	Comfort Inn Huntingdon	PA	The train behind the hotel was horrible when trying to sleep.

Table 3.6: A snippet from the Hotel dataset.

of the two hotels in the example are similar, their syntactic distance is high, leading to the invalidation of φ_3. For instance, the *levenshtein* distance between the Review Text values of the first and second tuples is 241, despite both reviews praise the quietness of the place and the view. Notably, the distance between the Review Text values of the first and fourth tuple is even lower (i.e., 195), despite the content of the reviews being completely unrelated.

As a final example, let us consider the *Kindle* dataset and the following RFD discovered by S-DIMε:

$$\varphi_4 : \mathsf{Asin}_{\leq 2}, \mathsf{Summary}_{\leq 16}, \xrightarrow{\Psi_{err(0)}} \mathsf{Review}_{\leq 16}$$

where *Summary* is the review's title and *Asin* represents the book's ID. φ_4 was not discovered by DIMε due to the presence of Summary and Review in the LHS and RHS, respectively. For example, consider Table 3.7, which contains a snippet extracted from the *Kindle* dataset. As we can see, also in this case there are clear similarities between the reviews of each book. However, with the *levenshtein* distance, it is not possible to capture such

similarities. For example, the distance between the reviews of the third and fourth tuples is 379, which is even higher than the distance between the reviews of the first and fourth tuples (i.e., 319), which refers to two different books.

Asin	Summary	Review Text
B000JMKX4W	Very Funny	I enjoyed this book a great deal - very witty and a pleasant read. I actually did laugh out loud a couple of times. Thanks, free classics!
B000JMKX4W	Fun read	It's funny. Definitely worth the price (free). Had me laughing. [...] Great for reading on your phone while waiting for anything.
B00LAQ5JWC	Delicious Mediterranean Meals	This is an absolutely delicious read that'll get you motivated and hungry for some Italian pasta! [...] Great and recommendable! People should really try reading this book.
B00LAQ5JWC	The Mediterranean Diet for Beginners	Mediterranean Diet - An Ideal Beginner's Guide [...] is great! The book is very well organized [...] I would recommend this book to those who want to change to a more healthy life style.I love it...Great Job!

Table 3.7: A snippet from the *Kindle* dataset.

3.2.2.4 *Limitations*

The evaluation of S-DIMε is limited to a set of 8 real-world datasets, primarily selected to assess the impact of the semantic module when dealing with long textual values. Moreover, it has been tested with 3 specific threshold configurations for the comparison criteria, on the basis of the value distribution of each attribute. Finally, S-DIMε employs a specific sentence embedding model, primarily selected for its wide language support and general-purpose effectiveness.

To summarize, in this work we explored the feasibility of using semantic distances instead of syntactic distances for RFD discovery. To this end, we considered the DIMε discovery algorithm and extended it with the ability to perform semantic-aware comparisons for attributes characterized by long texts. To do this, we exploited sentence embedding models to obtain a vectorized representation of textual values, and we used three different distances to compare the embeddings (i.e., Cosine, Manhattan, and Euclidean). We then compared the results of S-DIMε and DIMε execution times, memory consumption, and discovered RFDs. As concerns the execution times, S-DIMε outperformed DIMε, since the comparison between vector embeddings resulted to be faster than the calculation of the *levenshtein* distance between long texts. Moreover, when compared on the dataset with the longest textual values, S-DIMε outperformed DIMε also in terms of memory consumption, whereas on the other datasets the two algorithms showed a similar behaviour. In terms of RFDs discovered, S-DIMε enabled to discover new meaningful dependencies, with percentages of *New* RFDs ranging from 7.30% to 39.53%. Among the three considered distance measures, we noticed that the Manhattan distance is the one that differs the most from the syntactical discovery, as it leads to the extraction of more novel RFDs. In contrast, the Euclidean distance provided fewer novel dependencies; finally, the Cosine distance proved to be a halfway point between the other measures.

3.3 SOLVING CHALLENGE 2: RYAN [41]

As discussed in Section 3.1, one important research direction in the field of Data Profiling lies in the introduction of visual tools capable of visualizing and explaining the discovered metadata. This task becomes even more difficult in dynamic scenarios, in which metadata evolves according to data updates (e.g., specializations and generalizations). To address this problem, we proposed the RYAN (**RFD explainabilitY & ANalysis tool**) tool. RYAN allows for comparing metadata discovered on a set of data before and after data changes. It provides an interactive dashboard with new visual metaphors to summarize changes into metadata according to such an evolution of data. Moreover, it enables users to obtain explanations on updated RFDs, aiming to understand if they map the natural evolution of the application

domain or if they are due to the inclusion of erroneous data in the dataset. The following sections will detail the components underlying RYAN, as well as the user study performed to evaluate it (Sections 3.3.1 and 3.3.2).

3.3.1 *The* RYAN *tool*

RYAN emerges as one of the first AI-based tools that analyzes how RFDs discovered on a dataset evolve as the data changes. It is able to categorize the discovered RFDs, enabling users to understand how data changes affect the validity of RFDs, combining visual components and detailed explanations. This is useful in scenarios in which it is necessary to monitor the impact of changes, such as maintaining integrity, checking the quality, or identifying anomalies in the data. In this section, we detail the architecture of RYAN, its components, and their interactions for analyzing discovered RFDs. A demo of RYAN is available on YouTube[2] whereas the source code and further details are available on GitHub[3].

3.3.1.1 *System architecture*

RYAN is built upon a client-server architecture, leveraging a Flask web server configured ad-hoc to provide both interacting with LLMs and processing of RFDs holding on a given dataset. Figure 3.6 shows the architecture and the modules underlying RYAN.

Starting from the client side, RYAN exposes an interactive visual interface that enables users to load the dataset on which they performed the discovery step with their changes and the RFDs discovered before and after applying the changes. After loading this data, the interface shows several components to analyze the set of RFDs and obtain explanations using LLMs. All visual components were developed using the Data-Driven Documents (D3.js)[4] framework that allowed us to obtain flexible and interactive components capable of easily processing and visualizing large amounts of information. The visual components of RYAN communicate with the server-side modules using API

2 YouTube: https://www.youtube.com/@dastlabunisa2379
3 Repository: https://github.com/Roby46/RYAN-RFD-explainability-Tool
4 D3 Library: https://d3js.org/

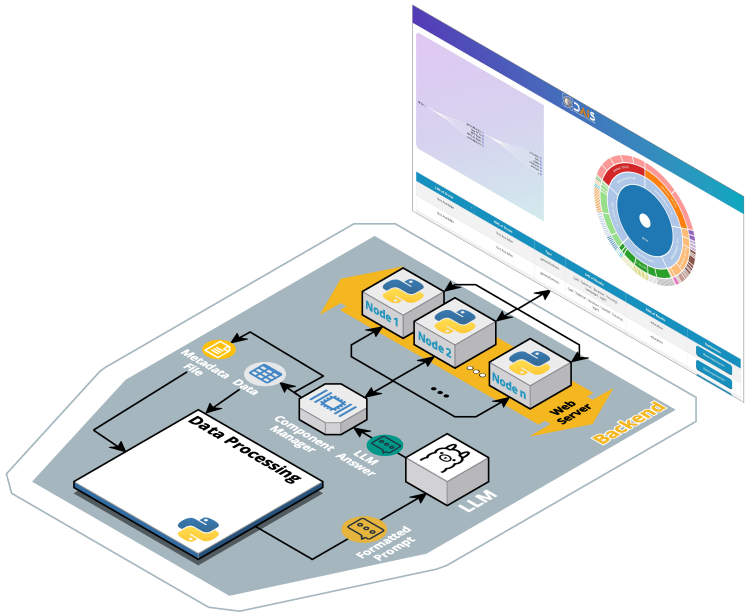


Figure 3.6: Overview of the architecture of RYAN.

mechanisms, which enable efficient data exchange and analysis. The server relies on a multi-threaded mechanism to enhance its scalability and responsiveness, allowing it to handle concurrent requests from multiple components without significantly affecting performance. In addition, we defined a component manager responsible for receiving data from the client side, interacting with the Data Processing module and the LLM. The former represents the core of RYAN's operations, as this module is responsible for identifying the update operations that led to the evolution of an RFD, and for interacting with LLMs to generate the natural language explanations.

3.3.1.2 *Data Processing Module*

Among the modules of RYAN, the Data Processing module is responsible for identifying and synthesizing the reasons that led to a generalization or specialization of an RFD φ valid at time τ to pass them on to the LLM for explanation. As shown in Figure 3.7, for each original RFD φ, the module receives as input φ and its generalized/specialized version (φ'/φ'', resp.). To perform its

operations, the module considers a representation of the dataset containing the original data and all update operations (insertions and deletions) performed on it. To keep track of these operations, for each tuple t_i, this representation contains an additional value that is set to 0 if t_i already exists within the original dataset. On the other hand, this value is set to j if t_i was inserted at time $\tau + j$, or to $-j$ if t_i was deleted at time $\tau + j$. For instance, in Figure 3.7, the original dataset contains the tuples ranging from t_0 to t_7, and two update operations are performed: the deletion of t_6 at time 1 and the insertion of t_8 at time 2. According to this representation and to the type of RFD evolution, the Data Processing module will perform different computations to identify the reasons for the considered evolution by leveraging the concept of *Similarity Class Partitions*.

Definition 14 (Similarity Class Partition). *Given a relation schema R, an instance r of R, and a set of attributes X, two tuples t and u are similar on X if $t[A]$ is similar to $u[A]$ $\forall A$ in X, according to a conjunction of predicates $\Phi = \bigwedge_{A_i \in X} \phi_i[A_i]$. Any attribute set X partitions the tuples of the relation into similarity classes. The similarity class of a tuple $t \in r$ with respect to a given set $X \subseteq R$, denoted by $[t]_X$, is defined a as $[t]_X = \{u \in r \mid t[A]$ is similar to $u[A] \mid \forall A \in X\}$. A similarity partition is defined as a set $\pi_x = \{[t]_X \mid t \in r\}$. In other words, π_x is a collection of sets (similarity classes) of tuples, such that each set has a unique value for the attribute set X and the union of these sets equals the relation r. Notice that, unlike equivalence class partitions, similarity partitions are composed of non-disjoint sets.*

Example 3.3.1. Consider the snippet of the dataset in Figure 3.7 and its initial set of tuples ranging from t_0 to t_7. The similarity partition of the Citation attribute when considering 2 as a similarity threshold is as follows:

$$\pi_{\text{Citations}} = \{28 : \{1,7\}, 30 : \{1,7\}, 58 : \{0,4\}, 60 : \{0,4\}\}$$

For example, t_1 and t_7 have similar values for the attribute Citations, since the difference between their values (i.e., 30 and 28, respectively) is less than or equal to the threshold defined on Citations, which is 2.

Thus, the partitions store, for each attribute, the set of tuples that are similar to it according to the similarity threshold defined

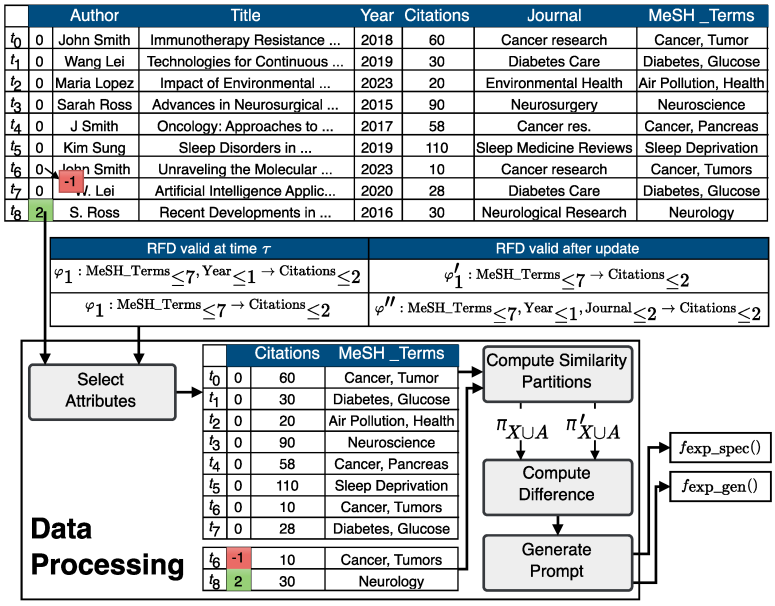


Figure 3.7: Overview of the Data Processing module of RYAN.

on that attribute. In what follows, we detail how these partitions are leverage by RYAN to analyze *Generalizations* and *Specializations* of RFDs.

a) Generalizations. In case of generalization, the processing module receives the original RFD φ: $X_{\Phi_1} \xrightarrow{\Psi \leq \varepsilon} A_{\phi_2}$ and its generalized version φ': $X'_{\Phi'_1} \xrightarrow{\Psi \leq \varepsilon} A_{\phi_2}$. Thus, the module has to identify which operations removed tuples that, in the original data, invalidated φ', leading to the discovery of φ. To accomplish this, we compute similarity partitions [49] for determining which tuples are similar to others for a given attribute value. These partitions, computed on the dataset in relation to the attributes involved in φ', can be analyzed to evaluate the impact of each deletion operation. It is important to note that similarity partitions do not satisfy the transitivity property due to the inherent definition of similarity [49]. At first, the processing module filters the attributes of the dataset, keeping only the attributes $B \in X' \cup A$. Thereafter, it computes similarity partitions of the dataset before deletions, namely $\pi_{X \cup A}$, and after deletions, namely $\pi'_{X \cup A}$. Then, to understand the impact the changes have had on φ, the mod-

ule computes the differences between the two sets of partitions, i.e., $\pi_{X \cup A}$ and $\pi'_{X \cup A}$. The resulting set of partitions contains the changes that occurred with respect to the original data. Then, for each deleted tuple t_i, the module identifies the set v_i of all tuples t_j, with $j \neq i$, which belong to the same similarity partition of t_i for the attributes in X' but not for the attribute A. The case where v_i is not empty indicates that tuple t_i had similar values on all the attributes in X' with respect to the tuples in v_i but different ones on A, leading to the violation of φ'. After checking all the deleted tuples, the module generates an ad-hoc prompt with all the identified tuples t_i and their associated sets v_i (see Section 3.3.1.4).

Example 3.3.2. Let us consider the snippet of the dataset in Figure 3.7 and its initial set of tuples ranging from t_0 to t_7. It is possible to notice that the RFD:

$$\varphi : \mathsf{MeSH_Terms}_{\leq 7}, \mathsf{Year}_{\leq 1} \xrightarrow{\Psi_{err(0)}} \mathsf{Citations}_{\leq 2}$$

holds on this set of tuples. Now, let us examine the deletion of t_6 at time $\tau + 1$: it is possible to notice that it leads to the validation of:

$$\varphi' : \mathsf{MeSH_Terms}_{\leq 7} \xrightarrow{\Psi_{err(0)}} \mathsf{Citations}_{\leq 2}$$

making the previously discovered φ no longer minimal. Let us suppose that φ and φ' are given as input to the data processing module to analyze this evolution. The module filters the dataset's columns by keeping only those involved in φ'. Then, it computes the similarity partition $\pi_{X \cup A}$ by considering tuples ranging from t_0 to t_7 and the partition $\pi'_{X \cup A}$ by excluding t_6:

$\pi_{X \cup A}$ = {Citations: {28: {1, 7}, 30: {1, 7}, 58: {0, 4}, 60: {0, 4}}, MeSH_Terms: {"Cancer, Tumor": {0, 6}, "Cancer, Tumors": {0, 6}, "Diabetes, Glucose": {1, 7}}}

$\pi'_{X \cup A}$ = {Citations: {28: {1, 7}, 30: {1, 7}, 58: {0, 4}, 60: {0, 4}}, MeSH_Terms: {"Cancer, Tumor": {0}, "Cancer, Tumors": {0}, "Diabetes, Glucose": {1, 7}}}

It is worth pointing out that for the calculation of these partitions, we consider the thresholds defined in φ'. After the computation of $\pi_{X \cup A}$ and $\pi'_{X \cup A}$, the processing module considers their difference:

$\pi_{X \cup A} \setminus \pi'_{X \cup A}$ = {Citations: {}, MeSH_Terms: {"Cancer, Tumor": {0, 6}, "Cancer, Tumors": {0, 6} }}

At this point, we can see that t_6 appears in a similarity partition together with t_0 for all the attributes in the LHS of φ' (i.e., MeSH_Terms) but not for the attribute in its RHS (i.e., Citations). This implies that t_0, due to the presence of t_6, invalidated φ'. However, the deletion of t_6 has led to the validation of φ' that is minimal with respect to φ. Thus, the data processing module identifies the deletion of t_6 as the reason for the generalization of φ, and adds t_0 to the set v_6 associated with t_6. This information will then be leveraged to fill the prompt template for explanation.

ii) Specializations. In case of specialization, the processing module receives the original RFD φ: $X_{\Phi_1} \xrightarrow{\Psi_{\leq \varepsilon}} A_{\phi_2}$ and its specialized version φ'': $X''_{\Phi''_1} \xrightarrow{\Psi_{\leq \varepsilon}} A_{\phi_2}$ and it has to identify which operations added tuples that invalidated φ, leading to the discovery of φ''. Thus, to assess the impact of each insertion operation, the processing selects from the dataset only the data of the attributes in $X \cup A$. Then, it computes similarity partitions of the dataset before and after all insertions and the difference between the two resulting sets $\pi_{X \cup A}$ and $\pi''_{X \cup A}$ to obtain the variations from the original partition. For each inserted tuple t_i, the module identifies the set v_i of all the tuples t_j, with $j \neq i$, which belong to the same similarity partition of t_i for the attributes in X but not for the attribute A. The case where v_i is not empty indicates that the tuple t_i had similar values on X with respect to other tuples but different ones on A, leading to the violation of φ. After checking all the inserted tuples, the module generates the prompt with all the identified tuples t_i and their associated sets v_i.

Example 3.3.3. Let us consider the dataset in Figure 3.7, which was updated after the deletion operation described in Example 3.3.2, and examine the insertion of t_8 at time $\tau + 2$. This insertion causes the invalidation of:

$$\varphi' : \mathsf{MeSH_Terms}_{\leq 7} \xrightarrow{\Psi_{err(0)}} \mathsf{Citations}_{\leq 2}$$

and the consequent validation of:

$$\varphi'' : \mathsf{MeSH_Terms}_{\leq 7}, \mathsf{Year}_{\leq 1}, \mathsf{Journal}_{\leq 5} \xrightarrow{\Psi_{err(0)}} \mathsf{Citations}_{\leq 2}.$$

The data processing module, after filtering the dataset's columns related to the attributes in φ', computes the similarity partitions $\pi_{X \cup A}$ and $\pi''_{X \cup A}$ considering the new tuple t_7. Let us suppose

φ' and φ'' are given as input to the data processing module to analyze this evolution. As described, the module filters the dataset's columns by keeping only those involved in φ'. Thereafter, it computes the similarity partition $\pi_{X \cup A}$ by considering the tuples ranging from t_0 to t_7 with the exception of t_6, since it was deleted at time $\tau + 1$. Finally, the module computes $\pi''_{X \cup A}$ by also considering t_7. The two obtained similarity partitions are the following:

$\pi_{X \cup A}$ = {Citations: {28: {1, 7}, 30: {1, 7}, 58: {0, 4}, 60: {0, 4}}, MeSH_Terms: {"Cancer, Tumor": {0}, "Cancer, Tumors": {0}, "Diabetes, Glucose": {1, 7}}}

$\pi''_{X \cup A}$ = {Citations: {28: {1, 7, 8}, 30: {1, 7, 8}, 58: {0, 4}, 60: {0, 4}}, MeSH_Terms: {"Cancer, Tumor": {0}, "Cancer, Tumors": {0}, "Diabetes, Glucose": {1, 7} "Neuroscience": {8, 3} "Neurology": {8, 3} }}

For the generation of these partitions, we consider the thresholds defined in the original RFD, i.e., φ'. After the computation of $\pi_{X \cup A}$ and $\pi''_{X \cup A}$, the data processing module considers their difference:

$\pi''_{X \cup A} \setminus \pi_{X \cup A}$ = {Citations:{28:{1, 7, 8}, 30:{1, 7, 8}}, MeSH_Terms:{"Neuroscience":{3, 8}, "Neurology":{3, 8}}}

As we can see, t_8 appears in a similarity partition together with t_3 for all the attributes in the LHS of φ' but not for the attribute in its RHS. This implies that the insertion of t_8 causes the invalidation of φ' due to the presence of t_3. Hence, the processing module identifies this insertion as the reason for the invalidation and adds t_3 to the set v_8 associated with t_8.

3.3.1.3 *Interaction with Visual Components*

At startup, RYAN presents a simple interface that allows users to load a dataset and a JSON file that includes the RFDs discovered before and after the data updates. The starting page is depicted in Figure 3.8.

In addition, a tutorial page with details on both RFDs and the functionalities of RYAN is always available by clicking on a *info* icon at the bottom left of the interface (see Figure 3.9). After loading the two required files, RYAN automatically switches to the analysis page, which includes all the visual components and the functionalities of the tool, as shown in Figure 3.10. In what

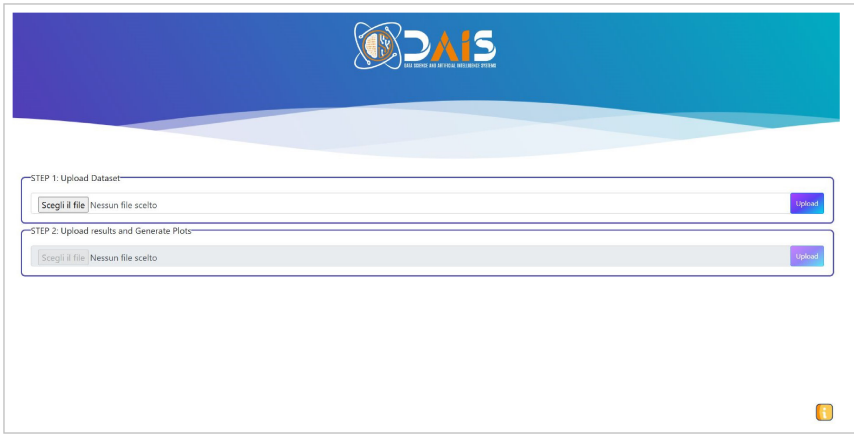


Figure 3.8: RYAN's homepage.

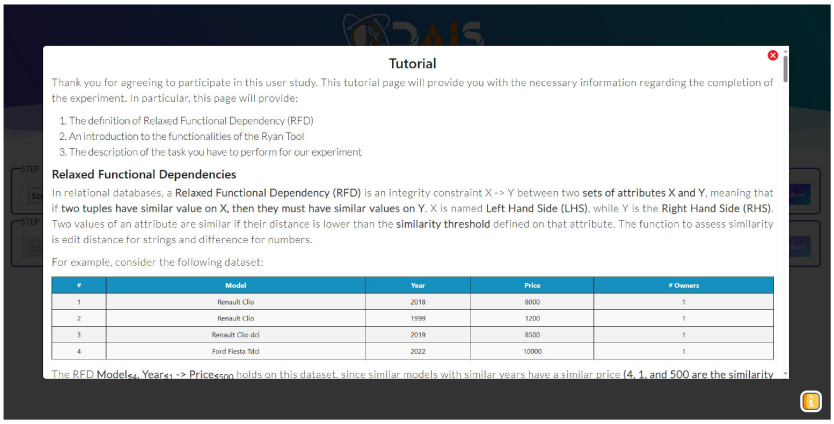


Figure 3.9: Tutorial page.

follows, we detail all the visual components provided by RYAN.

i) RFD *Tree.* The RFD tree (on the left of Figure 3.10) introduces a collapsible representation of all RFDs discovered on the updated data, i.e., at time $\tau + 1$, and their relation with those discovered at time τ. The RFD tree contains a root node, labeled "RFDs", linked to five child nodes, i.e., *Generalizations*, *Specializations*, RFDs *Found*, RFDs *Not Found*, and *New* RFDs. The *Generalizations/Specializations* represent all the RFDs discovered at time $\tau + 1$ that generalize/specialize at least one RFDs discovered at time τ. The RFDs *Found* nodes are all valid RFDs both at time τ and $\tau + 1$,

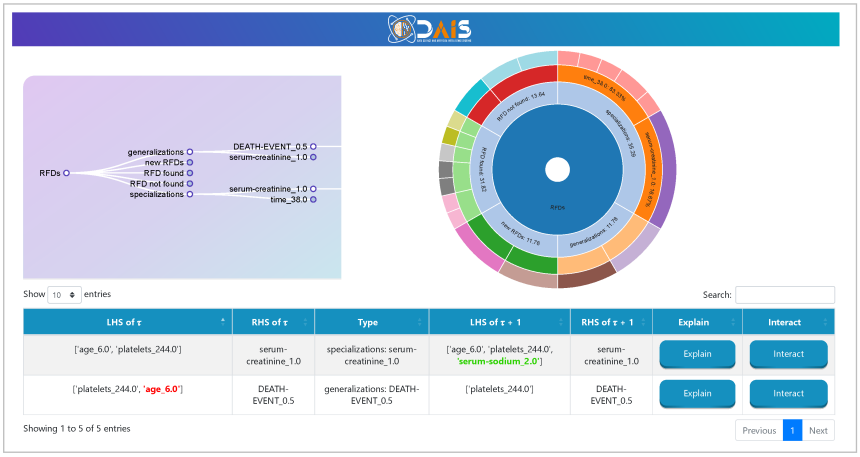


Figure 3.10: Overview of RYAN's visual components.

whereas the RFDs *Not Found* nodes are all RFDs that were discovered at time τ but were not discovered at time $\tau + 1$. The *New* RFDs nodes are all RFDs discovered at the time $\tau + 1$ that were not discovered at the time τ. A click on a node causes the tree to show new nodes, each labeled with an RHS attribute, which, when clicked further, shows the child nodes labeled with the LHS attributes associated with the chosen RHS. This representation is useful for systematically analyzing all RFDs since it allows users to analyze only the categories of interest and gradually drill down into the details of the most relevant RFDs.

ii) RFD *Sunburst.* Similar to the RFDs tree, the Sunburst chart provides a hierarchical representation of RFDs, considering a higher level of abstraction to provide statistics about the five categories of RFDs. With the Sunburst chart, the user has an immediate view of the entire set of RFDs discovered, without having to explore the tree through interaction with its nodes. As shown in Figure 3.10, the RFD Sunburst is composed of different rings. The innermost ring (i.e., "RFDs") is the root, while the second one contains the categories of RFDs. Finally, the third and fourth rings contain the segments of the RHS and respective LHS of the RFDs discovered upon data changes, respectively. Each segment contains both a label of the represented value (i.e., Category, RHS, or LHS) and the percentages of the categories computed based on the totality

of RFDs extracted at time τ (or the totality of RFDs extracted at time $\tau + 1$ only for the category *New* RFDs).

iii) Comparative DataTable. The comparative DataTable aims to provide a more detailed representation of RFDs and to get an explanation of the impact that data changes have had on the holding RFDs. It is composed of five columns: the first two represent the LHS and the RHS of an RFD φ valid at time τ; the third is the motivation with which the RFD discovered at the time $\tau + 1$ is related to φ, while the last two contains the attributes on the LHS and the RHS of an RFD validated at time $\tau + 1$. As shown in Figure 3.10, when displaying a specialized RFD, the additional attributes are highlighted in green, while when showing a generalized RFD, the removed attributes are highlighted in red. In both cases, the DataTable displays an "Explain" button that will interact with LLMs by defining a textual prompt (see Section 3.3.1.4), and will redirect to a page containing the textual explanation (see Figure 3.11). In addition, the DataTable provides an "Interact" button, which allows to customize the textual prompt for performing tailored analyses. These features are the most distinctive characteristics of the DataTable, allowing users to exploit LLMs to provide a descriptive analysis of why a RFD discovered at time τ was specialized/generalized at time $\tau + 1$. Moreover, the DataTable offers ordering, searching, and pagination features to improve the accessibility of the displayed RFDs.

3.3.1.4 *Interacting with LLMs*

Interacting with LLMs for specific tasks requires designing tailored prompt engineering strategies for obtaining reliable results. Since no LLM has been empirically proven to outperform others for specific tasks, we equipped RYAN with various frameworks for interacting with LLMs. In particular, RYAN is able to interact with several open and proprietary LLMs, i.e., those based on GPT-J, Llama, Replit, StarCoder, or Falcon architectures, GPT-4o, Claude 3 Haiku, Mixtral, and Llama 3.1 using the Gpt4all framework for open LLMs and the DuckDuckGo and OpenAI APIs for proprietary LLMs.

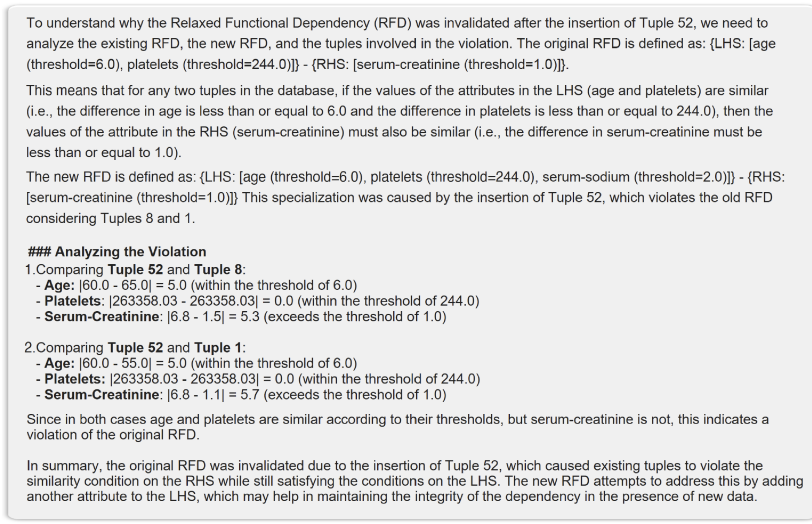


Figure 3.11: Overview of RYAN interfaces.

The interaction with LLMs for specific tasks requires designing ad-hoc prompt engineering strategies to achieve reliable results. For RYAN, we propose three different ad-hoc prompting functions to interact with LLMs to *i*) provide a context with base concepts about RFDs, *ii*) explain the reasons why a specific RFD is specialized, and *iii*) explain the reasons why an RFD gets generalized. To this end, we employed the Manual Template Engineering approach to generate prompts, considered the most intuitive method for crafting templates derived from human insights [124].

For the first prompt, we have defined a prompting function $f_{\text{context}}()$ that aims to provide a context $x' = f_{\text{context}}()$, with the fundamental concepts about RFDs. This prompt will then be involved with the second/third one to generate the explanation for specializations (generalization, res.). The template of the first prompt was defined as follows:

Template of $f_{\text{context}}()$

In relational databases, a Relaxed Functional Dependency (RFD) is an integrity constraint X $\rightarrow$ Y between two sets of attributes X and Y, meaning that if two tuples have similar values on X, then they must have similar values on Y. X is named Left Hand Side (LHS), while Y is the Right Hand Side (RHS). Two values of an attribute are similar if their distance is lower than the similarity threshold defined on that attribute. The function to assess similarity is edit distance for strings and difference for numbers.

This first prompt provides a context to the LLM in order to obtain more precise answers, as demonstrated by several studies in the literature [43, 64].

For the second prompt, we have defined a prompting function $f_{\text{exp_spec}}(x)$ that aims to complete the sentence x in order to achieve the prompt sentence $x'' = f_{\text{exp_spec}}(x)$. The template of this prompt was defined as follows:

Template of $f_{\text{exp_spec}}(c, \varphi, \varphi'', t_1, \ldots, t_m, v_1, \ldots, v_m)$

[C] After the insertion of a new tuple, an existing RFD can be invalidated only if the new tuple has similar values on the LHS with respect to other tuples but it has different values on the RHS. In this case, a specialized RFD with additional attributes on the LHS may be valid on the dataset. You will be provided with an RFD that gets invalidated after the insertion of a batch of tuples, and with the tuples that caused the violation and their values. The RFD [Φ] was invalidated and specialized by the new RFD [Φ″]. This happened because:
- The insertion of [T_1] caused a violation considering [V_1]
...
- The insertion of [T_m] caused a violation considering [V_m].
Basing on this information, provide an extensive explanation of why the RFD was invalidated, by analyzing the attribute values and by considering the similarity thresholds. [H]

where [C] is the slot for the context provided by the first prompt, [Φ] the slot for the invalidated RFD, [Φ″] the one for its specialized version, [T_i] is the slot for the inserted tuple that, due to the presence of the tuples listed in the slot [V_i], caused the violation of [Φ].

For the third prompt, we have defined a prompting function $f_{\text{exp_gen}}(x)$ that aims to complete the sentence x in order to achieve the prompt sentence $x''' = f_{\text{exp_gen}}(x)$. The template of this sentence was defined as follows:

Template of $f_{\text{exp_gen}}(c, \varphi, \varphi', t_1, \ldots, t_m, v_1, \ldots, v_m)$

[C] After the deletion of a tuple, an existing RFD can no longer be minimal. Indeed, a generalized RFD with a subset of attributes on the LHS may be valid on the dataset if the deleted tuple had similar values on the LHS with respect to other tuples, but it had different values on the RHS. You will be provided with an RFD that gets generalized after the deletion of a batch of tuples, and with the tuples that caused the generalization and their values. The RFD [Φ] was generalized by the new RFD [Φ′]. This happened because:
- [T_1], which caused a violation considering [V_1], has been removed.
...
- [T_m], which caused a violation considering [V_m], has been removed..
Based on this information, provide an extensive explanation of why the RFD was generalized. To do this, analyze the attribute values and consider the similarity thresholds. [H]

where [Φ] is the slot for the RFD that is no longer minimal, [Φ′] the one for its generalized version, [T_i] is the slot for the deleted tuple that, before the deletion operation, was invalidating [Φ′] due to the presence of the tuples listed in the slot [V_i].

3.3.2 *User Study*

To demonstrate the effectiveness of RYAN, we defined a user study concerning the analysis of RFD evolution in the healthcare domain. This section first details the users involved in our study and the survey they completed, followed by a discussion of the results.

3.3.2.1 *Experimental Settings*

We recruited 25 participants from the Department of Computer Science at the University of Salerno. Statistics about participants have been collected through a background survey, as shown in Table 3.8 (Q1-Q4). In particular, participants included 12 students, 4 research fellows, 8 Ph.D. students, and an associate professor. All participants stated that they had some level of knowledge of databases, i.e., 12 high, 9 medium, and 4 low. Regarding RFDs, 8 participants were unfamiliar with the topic, 4 had a basic understanding, 4 had medium knowledge, and 9 were domain experts.

Before the evaluation, each participant underwent a 20-minute tutorial on RFDs and RYAN. Each participant was assigned a task, which first consisted of loading a dataset relating to heart fail-

ure events[5], and RFD discovery results obtained at two different time instants following different insertion, deletion, and update operations. Afterward, participants were asked to interact with the sunburst chart to identify the total percentage of generalized/specialized RFDs. Then, participants were randomly given a target RFD discovered on the updated data, with the objective to identify the target dependency via the RFD tree and to verify from which original RFD it derived. Finally, each user was asked to exploit RYAN's explanation functionality to understand why the RFD evolution occurred.

The test was conducted under the supervision of an expert, who was available to answer any questions throughout the session. Additionally, the order of tasks varied among participants to minimize any effect of the order on the results. After completing the assigned tasks, participants were requested to fill out a questionnaire (see Table 3.8 (Q5-Q18)). These questions have been measured through a Likert scale, ranging from 1 (i.e., "Strongly disagree") to 5 (i.e., "Strongly agree"). In addition, the questionnaire ends with 3 open-ended questions (Q19-Q21), aimed at obtaining useful comments and hints.

3.3.2.2 *Results*

Figure 3.12 shows the answers provided by participants to the quantitative questions (Q5-Q18). As we can see, the answers obtained are generally positive for all questions, with only two negative answers among all participants, indicating that RYAN received positive feedback. Specifically, the aspects on which most positive feedback was obtained (i.e., answers 4 or 5 on the Likert scale) concerned the ease of use (Q5), the presence of all relevant information on the screen (Q8), the ease in identifying target RFDs (Q11), and the clarity in showing how it evolved (Q12). Furthermore, the majority of users claim to be willing to use the tool in the future (Q17). The other questions also registered positive feedback, including those concerning the quality of the LLM-related features (Q13-Q16). The effectiveness of RYAN's functionalities led to 96% of the participants saying that they had a good impression of the tool (Q18). As can be seen, only two questions had a single negative feedback (Q10 and Q13),

5 Heart Failure Clinical Data

	Alias	Question
Statistics	Q1	Gender
	Q2	Qualification
	Q3	Level of knowledge on Databases
	Q4	Level of knowledge on RFDs
Tool Evaluation	Q5	The tool is simple to use
	Q6	The tool is easy to learn
	Q7	The user interface is pleasant
	Q8	The user interface displays all relevant information.
	Q9	The RFD tree and the DataTable provide a clear overview of discovered RFDs.
	Q10	It was easy to understand the metadata syntax.
	Q11	It was easy to find the target RFD to analyze
	Q12	The user interface clearly shows how an RFD has evolved
	Q13	The explanation provided by the LLM was clear.
	Q14	The explanation provided by the LLM was thorough.
	Q15	The explanation provided by the LLM is clearer compared to the pre-compiled prompt.
	Q16	The model's explanation makes it easier to identify the values that caused the RFD evolution compared to the pre-compiled prompt.
	Q17	I would like to use the tool in the future.
	Q18	In general, I had a good impression of RYAN.
Final	Q19	Do you have any improvements to suggest?
	Q20	Which features of the tool did you like the most?
	Q21	Which feature of the tool did you like the least?

Table 3.8: The questionnaire proposed to participants.

both coming from one of the non-expert participants. Listening to his/her comments, the participant found it more difficult to understand RFDs, making it difficult for him/her to comprehend the syntax (Q10) and the model answer (Q13). For this reason, after completing all the tests, we have further enriched the tutorial available when using the tool, allowing even those who find it more difficult to understand RFDs to be able to use RYAN without issues.

Examining the results in relation to the level of expertise of the users, the ratings obtained by expert users are all between 4 and 5, indicating the effectiveness of RYAN in supporting domain experts in analyzing the evolution of RFD. Furthermore, the ratings

of users who were not familiar with RFD before the tutorial were also generally high (between 3 and 5, with the exception of the participant mentioned above), indicating that RYAN is characterized by ease of interaction that allows its use also by this category of users. The first open-ended question (Q19) revealed that some participants suggested making recently uploaded files immediately available on the home page and adding a filtering function for the RFD tree nodes. Other users suggested improvements to the formatting of the pre-compiled prompt and other minor changes to the tutorial page. Another useful comment from an expert participant was to include, in the explanation page, a table with the tuples involved in the violation, allowing the user to look at the data immediately after reading the explanation. As regards Q20, the majority of participants highly appreciated the exploration of the RFD tree (48%) and the LLM-related functionalities (52%). For instance, one of the expert participants remarked that "*the* RFD *tree visualization enables clear, intuitive exploration of RFD categories, while RHS partitioning facilitates* RFD *grouping, which is ideal for large datasets*". A non-expert user, instead, said that "*the model explanation is very clear, helping me understand the* RFD *specialization despite my lack of prior knowledge*". Another expert user affirmed that "*This tool provides significant support for* RFDs*, using LLMs to explain results effectively and add insights beyond those from human experts.*" Finally, in response to Q21, no participant identified any feature as less appreciated.

3.3.2.3 *Limitations*

The evaluation of RYAN is limited to a case study involving several participants of the Department of Computer Science at the University of Salerno. Moreover, the study primarily focused on evaluating the usability of RYAN and its ability to provide clear natural language explanations. Other task-related evaluations, such as a measurement of the success rate could be considered, as well as a comparison with different LLM models.

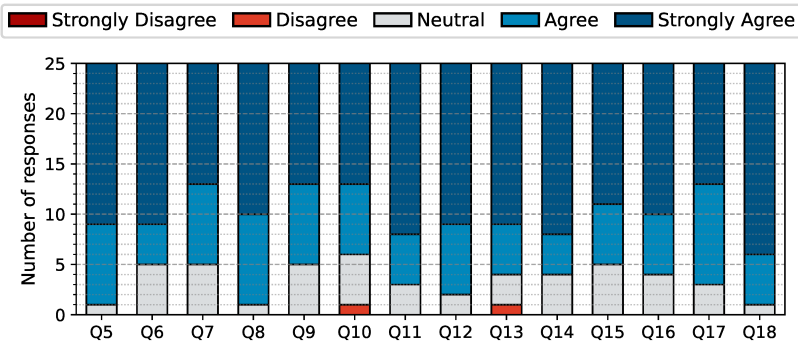


Figure 3.12: Answers provided by participants for Q5-Q18.

To summarize, we introduced the RYAN tool to compare RFD results from discovery processes performed at different time instants, allowing interactive analysis through new visual components. One of the most important features that distinguishes RYAN from other visualization tools is the integration of LLMs to obtain a textual explanation of how and why a particular RFD has evolved over time. To evaluate RYAN, we defined a user study by recruiting both expert and non-expert participants. The results highlighted RYAN's capability to provide effective visualizations of RFDs and to deliver detailed and correct explanations of their evolution over time.

SUMMARY OF PART I

In this part of the thesis, we first provided an overview of Data Profiling techniques, discussing the several types of metadata that can be discovered from data collections. Then, we formally introduced Functional Dependencies (FDs) and Relaxed Functional Dependencies (RFDs), which serve as the foundation for all the works proposed in this thesis. We discussed the wide range of use cases in which FDs and RFDs were successfully applied in recent years, and we overviewed the discovery algorithms that allow to automatically extract these types of metadata, enabling their practical application. Then, we highlighted some of the current challenges in the field of Data Profiling, focusing on those that limit the applicability of such techniques. To address the identified challenges, this part proposed two contributions. The first one is S-DiMε, a novel RFD discovery algorithm that enable semantic-aware comparisons when evaluating the similarities among long textual attributes. Through an experimental evaluation, we showed that S-DiMε enables the extraction of novel, meaningful RFDs, and that its execution time is significantly lower than traditional discovery algorithms, which employ syntactic distances. The second contribution discussed in this part is the RYAN tool, which allows to visualize RFD discovery results obtained at different time instants. Notably, RYAN is the first tool that introduces the possibility to interact with LLMs to provide explanations not only on the meaning of the discovered RFDs, but also on the reasons why a particular RFD has evolved over time. Though a user study, we showed that RYAN represents an user-friendly tool that facilitates the interpretation of discovery results for expert and/or non-expert users. Overall, the contributions presented in this part of the thesis aimed at broadening the applicability of Data Profiling, allowing to leverage metadata in a wider range of application scenarios.

Part II

METADATA-DRIVEN DATA IMPUTATION

4

DATA IMPUTATION

In contemporary research and industry, data have become the foundation of a wide range of data-driven processes. Therefore, the reliability and effectiveness of such processes heavily rely on the quality of the underlying data. As a matter of fact, several studies in recent years have highlighted the (potentially) severe effects of poor quality data in different contexts [83, 127]. Data quality is a multidimensional concept [148] which involves several aspects such as *Interpretability*, *Relevancy*, *Objectivity*, *Accessibility*, and *Completeness*. The latter aspect refers to the extent to which data is not missing, and represents one of the most important issues in Data Quality. To address this problem, Data Imputation techniques aim to automatically fill missing values by inferring plausible values from the data collection itself. This chapter discusses the Data imputation problem and how RFDs can be employed in this scenario (Section 4.1), and provides an overview of existing data imputation approaches in the literature (Section 4.2), ending with a discussion of current challenges and research gaps (Section 4.3).

4.1 PROBLEM OVERVIEW

The missing value problem is ubiquitous in real-world data [132], and severely affects any data-driven task. The presence of missing values can be due to different reasons, such as human errors, sensor failures, merging of unrelated data, or users refusing to fill the optional fields in forms. More formally, a missing value (MV) can be defined as follows:

Definition 15 (**Missing Value**). *Given a relation schema R, defined over a set of attributes $attr(R)$, an instance r of R, an attribute $A \in attr(R)$, and a tuple $t \in r$, a missing value of tuple t on the attribute A, denoted as $t[A] = _$, is such that $t[A]$ is null. Here, r is said to be an incomplete instance, and $\hat{r} \subseteq r$ is such that each $t \in \hat{r}$ is an incomplete tuple.*

Missing values can occur within data according to different missingness mechanisms, which can be categorized into three main groups [159]:

- **Missing Completely at Random (MCAR)**: The missing values are independent of both observed and unobserved data. In other words, the probability that a value is missing does not depend on any characteristic of the dataset.
- **Missing at Random (MAR)**: The missing values depend only on the observed data and not on the missing values themselves. For example, if older individuals are less likely to respond to a question, the missingness depends on the *Age* attribute.
- **Missing Not at Random (MNAR)**: When the missingness is related to the unobserved data itself or to features not included in the dataset. This is the most challenging scenario, as the causes of missingness cannot be fully explained by the available data.

To address the problem of missing values, Data Imputation approaches aim to fill incomplete instances by retrieving or inferring a plausible value from the same instances. More formally, the missing value imputation problem can be defined as follows:

Definition 16 (Data Imputation). *Given a relation schema R, and an instance r of R, for every tuple $t \in r$ and every attribute $A \in attr(R)$ for which $t[A] = _$, the imputation problem consists of finding a plausible value $a \in dom(A)$, resulting in an imputed instance r'.*

4.1.1 RFD$_c$-based Data Imputation

In the context of Data Imputation, dependencies such as FDs and RFDs can be used both to guarantee the verification of the semantic consistency of the imputed instance, and to drive the searching of meaningful candidates for all missing values [30]. Since RFD$_c$s represent the most general type of dependency used in data imputation processes, in what follows, we formally define the problem of imputing MVs by leveraging RFDs. Moreover, we particularly refer to the definitions provided by [30], since the

t	Name	Address	City	Phone Number	Type
t_0	arnie morton's of chicago	435 s. la ciniega blv.	-	310/246-1501	american
t_1	arnie morton's of chicago	435 s. la ciniega blvd.	los angeles	310/246-1501	steakhouses
t_2	art's delicatessen	12224 ventura blvd.	sherman oaks	818/762-1221	american
t_3	-	12224 ventura blvd.	studio city	-	-
t_4	hotel bel-air	701 stone canyon rd.	bel air	-	californian
t_5	trio cafe	1870 filmore st.	san francisco	415-563-2248	american
t_6	bel-air hotel	701 stone canyon rd.	bel air	-	california
t_7	cafe bizou	14016 ventura blvd.	sherman oaks	818/788-3536	french
t_8	cafe bizou	14016 ventura blvd.	sherman oaks	818-788-3536	french bistro
t_9	wa-ha-ka oaxaca mexican grill	2141 polk st.	san francisco	415-775-1055	mexican
t_{10}	3200 las vegas blvd. s.	3200 las vegas blvd. s.	las vegas	702/733-8899	asian
t_{11}	3200 las vegas blvd. s.	3200 las vegas blvd. s.	las vegas	702-733-8899	chinese

Table 4.1: A snippet of the *Restaurant* dataset.

latter represent the approach that takes more advantage of data dependencies for imputing MVs.

Definition 17 (Semantically consistent imputation). *Given a relation schema R, defined over a set of attributes attr(R), an instance r of R, and a set of* RFDs Σ *holding on r ($r \models \Sigma$), an instance r' of R resulting from an imputation process I over the instance r, denoted as $r' = I(r)$, is semantically consistent iff $r' \models \Sigma$.*

Definition 18 (Data imputation with RFD$_c$s). *Given a relational schema R, an instance r of R and a set of* RFD$_c$s Σ *holding on r ($r \vDash \Sigma$), an imputation process $I(r)$ consists of finding, $\forall A \in attr(R)$, $\forall t \in r$ s.t. $t[A] = _$, a tuple $t' \in r$ with $t'[A] \neq _$ s.t. (t, t') is similar w.r.t. Φ_1, according to $\varphi : X_{\Phi_1} \rightarrow A_{\phi_2}$, and s.t. the imputation $t[A] = t'[A]$ yields $r' = I(R)$ be a relation instance s.t. $r' \vDash \Sigma$.*

Example 4.1.1. Let us consider the snippet in Table 4.1 and the RFD$_c$ φ: $\mathsf{Name}_{\leq 2} \rightarrow \mathsf{City}_{\leq 8.5}$ holding on it. φ can be exploited to impute the missing value t_0[City] since t_1 has a similar value on the attribute Name. Thus, the value of its attribute City, "*Los Angeles*" is a plausible value for the imputation of t_0[City].

4.2 LITERATURE REVIEW

This section provides a detailed review of existing imputation approaches. In particular, it will first discuss imputation approaches tailored for relational data (Section 4.2.1), which represents the

most considered application scenario in existing works, then, it will discuss imputation approaches tailored for time series, which represents an emerging scenario in the context of Data Imputation (Section 4.2.2).

4.2.1 *Imputation of Relational Data*

Relational data is characterized by values arranged in a tabular form, with rows an columns. For this type of data, the most simple way to deal with missing values consists in the application of *Simple Imputation* approaches, which handle missing values by replacing them with the modal, mean, or median of available values. However, this very simple approach is not ideal; in fact, it can lead to unrealistic results and does not perform well with large modern datasets [70]. Thus, more complex and refined approaches are required to provide higher-quality imputations. To this end, researchers have introduced several techniques over the past years, which can be grouped in different categories according to the underlying methodology (see Table 4.2) and will be detailed in the following Sections.

4.2.1.1 *Clustering-based Approaches*

A category of data imputation approaches is based on clustering, which rely on finding plausible values belonging to complete instances that are part of the same cluster as the tuple to be imputed. In [189] the authors present CMI (Clustering-based Missing value Imputation), an approach that uses clustering, specifically the K-Means algorithm, to group similar instances in the dataset. When the dataset comprises both discrete and continuous attributes, the authors apply the simple matching method for discrete attributes and the Euclidean distance for continuous attributes to compute similarities. Following this, the authors adopt varying imputation methods based on the nature of the target attribute. When dealing with continuous target attributes, the authors leverage the kernel method for performing imputation. This technique harnesses the information within each cluster to estimate and fill missing values within instances characterized by continuous target attributes. Conversely, when the target attribute is discrete,

Category	Ref.	Year	Technique
Clustering-based	[189]	2008	Clustering instances with K-Means, then imputing continuous attributes via kernel estimation and discrete ones via nearest neighbors.
	[80]	2012	Clustering complete instances, assigning incomplete ones to clusters, and imputing using weighted kNN while updating clusters.
	[154]	2016	Assigning fuzzy memberships to clusters, estimating values within each cluster, and combining them through membership-weighted aggregation.
	[164]	2019	Computing similarity through Grey Relational Grade and predicting missing values using regression models within fuzzy clusters.
ML-based	[152]	2001	Using regression models tailored to variable types and enforcing plausibility constraints during imputation.
	[153]	2013	Identifying highly similar data segments, training decision trees or forests, and imputing values using segment-specific predictive models.
	[52]	2014	Performing kNN by using an adaptive value of k.
	[157]	2017	Learning a probabilistic graphical model and inferring the most likely repairs under integrity constraints.
	[90]	2018	Applying overcomplete autoencoders multiple times and averaging the estimations.
	[130]	2018	Training a variational autoencoder and iteratively updating missing values until convergence.
	[191]	2019	Transforming traffic data into images and reconstructing missing regions through a CNN encoder–decoder.
	[138]	2020	Using a variational autoencoder to impute missing values for mixed-type attributes.
	[171]	2020	Constructing a graph-based autoencoder with an adversarial critic to impute missing values.
	[185]	2020	Learning structural dependencies between features through attention mechanisms and generating context representations to impute.
Metadata-driven	[170]	2018	Exploiting DDs and maximizing imputations via ILP, randomized, and derandomized optimization.
	[178]	2020	Deriving values from FDs and association rules and integrating them to recover missing data at scale.
	[30]	2022	Leveraging RFDs to find candidate values and validating them through integrity-preserving checks.

Table 4.2: Summary of imputation approaches for relational data by category, year, and technique.

the authors adopt the nearest neighbor method (NN) to perform imputations, relying on neighboring instances within the same

cluster. A similar technique that also leverage K-Means is the one proposed in [80]. This approach starts by clustering complete instances and subsequently arranges the tuples requiring imputation in ascending order based on the number of missing values. The imputation process starts with the first instance that has to be imputed, which is assigned to the nearest cluster. Within this cluster, the algorithm identifies the k tuples most similar to the one under consideration. Then, the imputation is performed using a weighted variation of the k-Nearest Neighbors (kNN) method. Following imputation, the newly completed instance is included into the cluster, becoming available for supporting subsequent imputations. This aspect differentiates this approach from others, and the authors highlight that, in some cases, this approach outperforms methods that do not consider previous imputations. However, it is worth noting that errors made in earlier imputations can sometimes propagate into later stages of the process.

Other proposals are based on fuzzy clustering, a variant of traditional clustering in which a data point can belong to more than one cluster. The probability that a sample belongs to a certain cluster is expressed by a membership function. For instance, in [154] the authors introduce the Fuzzy Expectation Maximization and Fuzzy Clustering-based Missing Value Imputation Framework for Data Preprocessing (FEMI).FEMI operates by recognizing the inherent fuzziness within datasets. It acknowledges that records with missing values may exhibit fuzzy associations with multiple clusters, rather than an exclusive affiliation with a single one. These clusters represent groups of records sharing similar characteristics. FEMI estimates missing values separately for each cluster, potentially generating k imputed values if there are k clusters, and the final imputation is obtained through a weighted average of these k estimates, taking into account the membership degrees. Moreover, FEMI extends this notion of fuzziness to all records in the dataset, not only to those with missing values. It assumes that every record has fuzzy associations with all k clusters. Consequently, when imputing a missing value based on a given cluster, FEMI applies a fuzzy imputation technique that accounts for the membership degrees of all records in that cluster. This results in distinct imputed values across clusters,

reflecting the variations in membership degrees among records. A further clustering-based approach was proposed by Sefidian & Daneshpour [164]. The authors present a method that is based on a version of fuzzy-c-means utilizing concepts from *Grey System Theory* (GST), which is a mathematical framework that deals with systems characterized by limited information and uncertainty, aiming to model and analyze systems where traditional mathematical techniques may be inadequate due to incomplete or imprecise data. Specifically, the authors employ Grey Relational Grade (GRG), a metric used in GST that quantifies the degree of relationship or similarity between data points, instead of the distance metrics commonly used in the literature, such as Euclidean distance. To perform more accurate imputations, the algorithm performs feature selection based on Mutual Information (MI) within each cluster, aiming to select only relevant features. The imputation follows this procedure: first, the algorithm determines the importance of each missing attribute, then input instances are separated into fuzzy clusters. Subsequently, clusters meeting certain minimum conditions are selected, and highly dependent features are chosen within each group using MI. Finally, two predictive models (Multiple Linear Regression and Support Vector Regression) are applied to the selected features within each fuzzy cluster. The imputed value at the end is obtained by a weighted average of the values predicted by these models.

4.2.1.2 *Machine Learning-based Approaches*

Several imputation techniques are based on supervised machine learning approaches, among which many employ the k-Nearest Neighbors (kNN) algorithm. Cheng et al. [52] proposed a novel method to overcome the limitations of traditional kNN algorithms. Conventional kNN approaches often suffer from the use of a fixed k value for all test samples and the neglect of correlations among samples. To address these issues, the proposed Sparse Learning-based kNN (S-kNN) algorithm adapts the value of k for each test sample by considering both data correlations and local structural information. The core idea of S-kNN is to reconstruct test samples using information from the training samples to determine an optimal, sample-specific k. This adap-

tive k is then employed within the kNN framework to perform tasks such as classification, regression, and missing value imputation. The reconstruction process minimizes reconstruction error through a least-squares loss function, while a regularization term is introduced to enforce element-wise sparsity in the reconstruction coefficients, thereby enabling diverse k values across samples. Furthermore, to preserve the intrinsic local structure of the data, the algorithm integrates the Locality Preserving Projection (LPP) technique. Other methodologies employ regression-based imputation techniques. For instance, Raghunathan et al. [152] proposed an approach capable of imputing different types of variables by fitting a sequence of regression models—linear, logistic, Poisson, generalized logit, or combinations thereof. The choice of model depends on the nature of the variable to be imputed, specifically continuous, binary, categorical, count, or mixed. An additional feature of this method is the possibility to impose restrictions that define a subset of the population satisfying specific criteria, as well as to introduce logical bounds or constraints on the imputed values to ensure their plausibility and consistency with the data structure. A particular approach is the one proposed by [191], in which the authors proposed an imputation technique for traffic data based on Convolutional Neural Networks (CNNs). The key idea behind this approach is to transform the raw traffic data into spatial-temporal images. In this representation, the horizontal axis corresponds to sensor numbers, and the vertical axis represents time stamps. Valid traffic volume observations are represented as pixel values, while missing data points are set to -1. This transformation allows the traffic data imputation problem to be re-framed as an "image painting" problem, where the objective is to fill in the missing regions of the image. To accomplish this, the author employs a CNN-based context encoder-decoder pipeline. This pipeline first extracts high-level features from the incomplete data and then reconstructs a complete spatial-temporal image with the same dimensions as the original one. The CNN-based approach is capable of analyzing features at both macroscopic and microscopic levels, making it well-suited for capturing complex patterns in traffic data. The findings indicate that the proposed approach performs particularly well when the missing ratio

exceeds 30%. Another machine learning–based approach is proposed by Rahman et al. [153], who introduce two techniques for imputing both categorical and numerical missing values. These methods, namely DMI and SiMI, aim to improve imputation accuracy by identifying and exploiting horizontal segments within the dataset. The authors argue that real-world data often contain subsets of records exhibiting stronger similarity and higher attribute correlations than the dataset as a whole. DMI (Decision Tree–Based Missing Value Imputation) employs decision tree algorithms such as C4.5 in combination with the Expectation–Maximization Imputation (EMI) technique. It operates by detecting highly similar segments within the data and imputing missing values based on the relationships and correlations observed among records in each segment. SiMI, an extension of DMI, enhances this framework by integrating decision forest algorithms such as SysFor. Similar to DMI, SiMI identifies coherent data segments characterized by strong intra-segment similarity and attribute correlations. Its distinctive feature lies in its adaptive splitting and merging strategy: when a segment is too small to support reliable imputation, SiMI merges it with another suitable segment, thereby reinforcing similarity among records and strengthening attribute correlations within the resulting segment. A further machine learning-based approach is HOLOCLEAN [157], a comprehensive data-repairing framework driven by probabilistic inference techniques. This framework integrates qualitative data-repairing approaches, which rely on integrity constraints and external data sources, with quantitative data-repairing methods that utilize statistical properties of the input data. The primary goal of HOLOCLEAN is to automatically generate a probabilistic model for data repairing when given an inconsistent dataset as input. The workflow of HOLOCLEAN consists of three key steps. The first one is error Detection: this initial step involves identifying cells in the dataset that may contain inaccurate values. These cells are separated into "noisy" and "clean" cells. HOLOCLEAN allows flexibility in the choice of error detection methods, which could range from utilizing denial constraints and outlier detection mechanisms to external and labeled data. In the second phase, HOLOCLEAN leverages probabilistic semantics to express the uncertainty regarding the values

of noisy cells. Each cell is associated with a random variable, and a probabilistic graphical model is constructed to describe the distribution of these variables for all cells in the dataset. Data Repairing is the last phase: to repair the dataset, HOLOCLEAN employs statistical learning and inference techniques over the joint distribution of variables. It computes marginal probabilities for potential values of noisy cells and assigns the value that maximizes the probability to each cell. Empirical risk minimization is used to optimize the model parameters, with clean cell variables serving as evidence and noisy cell variables being inferred using methods like stochastic gradient descent and Gibbs sampling. Importantly, each repair, proposed by HOLOCLEAN, is associated with a marginal probability, which quantifies the confidence in the suggested repair. Furthermore, the consistency of repairs can be guaranteed by specifying a set of integrity constraints.

In recent years, researchers have also investigated more complex deep learning–based architectures for data imputation, including models built on autoencoders (AEs), generative adversarial networks, and attention mechanisms. In [185], the authors propose MIDA, a multiple imputation framework based on overcomplete deep denoising autoencoders (DAEs). The method builds on the standard DAE architecture, which reconstructs clean inputs from corrupted versions and therefore treats missing values as a particular form of noise. To enhance the capacity of the model to capture complex attribute interactions, the authors adopt an overcomplete architecture, where successive hidden layers contain more units than the input layer. This projection into a higher-dimensional latent space aims to support the recovery of missing information by enabling richer intermediate representations. Since DAEs typically require complete data for initialization, the method first applies a simple mean imputation to fill missing entries before training. During training, the model introduces additional stochastic corruption by randomly setting a portion of input components to zero, and it is optimized to minimize the reconstruction error on the observed values. Multiple imputation is achieved by running the DAE model several times, each one with a different set of randomly initialized weights, thus producing multiple reconstructions of the data. The final imputed dataset is obtained by averaging the imputations pro-

duced across these runs. The framework is designed to handle heterogeneous data types and varying missingness patterns, and it does not rely on the availability of fully observed instances for training. Another approach based on autoencoders is the Variational Autoencoder Imputation Method (VAEI) [130]. VAEI adopts a variational autoencoder (VAE), operating through two phases: model training and iterative imputation. In the training phase, missing components are initially replaced with zeros, after which the VAE is trained for a fixed number of iterations. The optimization objective combines the reconstruction error computed over observed entries with a regularization term enforcing a prior distribution on the latent variables, thereby constraining the encoder and stabilizing the learned representations. In the subsequent iterative imputation phase, missing components are newly set to zero and iteratively updated using the outputs of the trained VAE. At each iteration, the current imputed matrix is passed through the VAE to generate a reconstructed version; the discrepancy between the reconstructed and previous matrices, evaluated only on observed values, is then used to update the missing ones. The process continues until a convergence criterion is met or a predefined iteration limit is reached. Another variational-autoencoder–based technique is the Heterogeneous-Incomplete VAE (HI-VAE) [138]. This model extends the use of VAEs to scenarios where data may include mixed attribute types and incomplete observations. Similar to other VAE-based approaches, HI-VAE begins by replacing missing entries with zeros and employs an encoder capable of handling partially observed inputs. However, unlike standard VAE formulations, HI-VAE incorporates a decoder specifically designed to accommodate heterogeneous data by selecting appropriate likelihood models for different attribute types. To better capture dependencies across features, the latent representation produced by the encoder is further transformed into an intermediate homogeneous representation before reconstruction. The model is trained by maximizing a variational lower bound computed solely over observed components. Once the training phase is completed, the learned decoder is used to infer plausible values for missing entries, producing a single completed dataset. On the other hand, the work proposed in [171] leverages a generative adversarial framework for imput-

ing missing data. The Graph Imputation Neural Network (GINN) models each sample as a node in a similarity graph and applies a graph convolutional denoising autoencoder over this structure. Unlike previous autoencoder or VAE-based approaches, GINN introduces an adversarial component inspired by GANs, where a critic network is trained to distinguish imputed values from observed ones. The generator and critic are simultaneously trained, with the generator aiming to reconstruct missing values while also "fooling" the critic. This combination allows GINN to exploit both the relational structure among samples and the adversarial training process, improving the quality of imputed values, particularly when complex dependencies exist in the data. Finally, another recent approach, namely AimNet [185], introduces an attention-based mechanism for imputing missing values in mixed datasets containing both continuous and discrete attributes. Unlike the previous autoencoder or GAN-based methods, AimNet focuses on learning structural dependencies between features through a variation of the dot product attention mechanism. The model processes raw tabular data by combining continuous projections and contextual embeddings for discrete values, and it generates a unified context representation for each attribute to guide imputation. AimNet is trained end-to-end using a multi-task loss that simultaneously addresses regression for continuous features and classification for discrete ones. This attention-driven approach enables the model to effectively generalize, particularly in scenarios with systematically missing data, outperforming several state-of-the-art baselines, including ensemble tree models and deep generative architectures.

4.2.1.3 *Metadata-driven Approaches*

While the HOLOCLEAN algorithm (described in the previous Section) leverage metadata to support the imputation process, other approaches exploit them through deterministic and/or fully metadata-driven methodologies. For example, Wang et al. [178] presented a three-stage approach for addressing missing data in Big Data interfaces. In the initial stage, a Big Data model is designed to handle missing data recovery, accommodating both structured and semi-structured data. The second stage in-

volves the development of a missing data recovery method based on functional dependencies and association rules. This method augments data by generating values from domain-based and functional dependencies and integrates them into the existing training data. In the final stage, the approach estimates algorithm complexity for missing data recovery, enabling the analysis of diverse data types. The proposed method outperforms traditional techniques like Expectation-Maximization (EM) and Random Forest (RF) by achieving a 12% enhancement in performance when dealing with 30% missing data. It also supports parallel execution in distributed databases, resulting in a substantial speedup, particularly for datasets with numerous attributes. In [170], instead, the authors rely on Differential Dependencies (DDs), a specific type of RFDs [46], which exploits similarity rules in order to offer tolerance on value variations. To enrich the data imputation process, the authors studied the problem of maximizing the number of imputed missing values. In particular, the authors presented 4 different algorithms for treating missing values, i.e., an integer linear programming, its approximation, a randomized algorithm, and its derandomized version, called DERAND. The latter is identified by the authors as the ideal solution for imputing multiple missing values with a deterministic bound of approximation. A comparable approach proposed in the literature is RENUVER [30], which is designed to work with incomplete instances and can handle numerical, textual, and categorical data. Its core reliance lies on RFDs to identify potential tuples for imputation and to uphold dataset integrity following the imputation process. The algorithm takes as input the dataset and a set of RFDs discovered from it. Before initiating the imputation, a data preparation phase is executed by isolating the tuples requiring imputation and by eliminating all *key dependencies* from the initial set of RFDs. This omission is performed to optimize the execution time, as these dependencies do not contribute to the imputation process. Then, for each attribute with at least one missing value, the algorithm generates a map of dependencies that can be used for imputation. Within each collection, the RFDs are ordered in ascending order with respect to the similarity constraint associated with the considered attribute. Thereafter, all missing values within the dataset are subjected to the imputation procedure. Based on the estab-

Category	Ref.	Year	Technique
Interpolation	[86] [11]	2004 2018	Filling gaps between the time instants of a series by estimating values based on neighboring points.
ML-based	[177]	2000	Iteratively imputing each variable by training a regression model on the remaining ones and cycling through all dimensions until convergence.
	[32]	2018	Modeling temporal dependencies with RNNs and treating missing values as learnable variables.
	[79]	2020	Encoding incomplete sequences into a latent space via a VAE and modeling temporal evolution using a Gaussian process to capture smooth multi-channel dynamics.
	[17]	2021	Estimating each missing value through a temporal transformer that applies self-attention over fine-grained and coarse-grained temporal signals while incorporating correlated series via kernel regression.
	[68]	2023	Applying a self-attention-based architecture that reconstructs missing segments through stacked temporal attention blocks designed to capture long-range dependencies efficiently.
Dynamic / Incremental	[82]	2019	Incrementally updating fuzzy rules and imputing missing values in evolving data streams.

Table 4.3: Summary of imputation approaches for time series data by category, year, and technique.

lished order, each dependency map is considered, and potential candidates for imputation are sought by considering similarity patterns. Imputation occurs if a valid candidate is found, but it is retained only after checking that the integrity of the dataset is not compromised.

4.2.2 *Imputation of Time Series Data*

An important scenario in the context of Data Imputation that is increasingly gaining interest is represented by the imputation of time series, which often have to deal with a high number of missing values due to sensor failures. As observed also for tabular data, a simple strategy is to replace MVs with statistics, such as the mean or median. However, with time series there are other simple approaches such as *Forward Fill* and *Backward Fill*, which propagate the last or the next observed value to fill MVs. However, the need of more refined approaches led to

the introduction of novel techniques, which can be grouped in different categories according to the underlying methodology (see Table 4.3) and will be detailed in the following Sections.

4.2.2.1 *Interpolation Approaches*

A category of approaches specifically tailored for time series consists of *interpolation* techniques, which aim to fill gaps between the time instants of a series by estimating values based on neighboring points. These methods rely on the assumption that the local structure of the series provides sufficient information to produce meaningful intermediate values, preserving the overall trend and dynamics of the data. The most simple approach is the Linear interpolation [86], which fills each gap within a data series by fitting a straight line between the observed endpoints. While computationally efficient and effective for approximately linear series, this method may fail to capture more complex or non-linear patterns in the data. To address such limitations, interpolation can be extended to higher-order polynomial methods [116, 134], allowing for a more flexible modeling of local trends. This approach can be further extended to higher degrees [116, 134] to better capture non-linear trends. Among these, cubic splines are particularly popular [11]: they construct piecewise cubic polynomials between consecutive data points while ensuring continuity and smoothness up to the second derivative. This results in a smooth approximation that can accurately reflect curvatures and local variations without introducing the oscillations often observed in global polynomial interpolation.

4.2.2.2 *Machine Learning-based Approaches*

Among ML-based approaches, a simple and commonly used method is MICE [177], which uses an iterative approach to impute MVs by building a model for each dimension, using the other ones as predictors. However, recent years have witnessed the introduction of more complex approaches based on deep learning-based. Among these, BRITS [32], focuses on imputing missing values in multiple correlated time series. This method leverages bidirectional recurrent neural networks (RNNs) to model the temporal dependencies in both forward and backward

directions, without imposing strong assumptions on the underlying data. BRITS treats missing values as variables within the RNN computational graph, allowing gradients to flow through them during backpropagation, which improves the accuracy of imputation through consistency constraints. In addition, BRITS jointly performs imputation and downstream tasks such as classification or regression within the same network, reducing the propagation of errors from the imputation step to predictive tasks and enhancing overall performance in time series applications. Instead, GP-VAE [79] uses a combination of variational autoencoders (VAEs) and Gaussian processes (GPs). The method first maps the incomplete time series into a lower-dimensional latent space where missingness is effectively removed. Temporal dynamics are then modeled in this latent space using a GP prior, allowing the model to capture smooth, interpretable temporal evolution across multiple channels that may operate at different frequencies. The approach employs structured variational approximations to efficiently perform inference, incorporating correlations across both time and features while maintaining linear time complexity. Finally, recent approaches have investigated the use of self-attention mechanisms. Among these, DeepMVI [17] models each missing value as a distribution conditioned on both fine-grained and coarse-grained temporal signals within the series, as well as information from correlated series at the same time. The approach employs a modular deep learning architecture that explicitly leverages self-attention through a temporal transformer to capture temporal dependencies across the series, combined with kernel regression to incorporate information from correlated series. To improve generalization, the model is trained using simulated missing blocks around observed values, which mimic real missing patterns. Another approach that relies on attention mechanisms is SAITS [68], which learns missing values from a weighted combination of two diagonally-masked self attention blocks.

4.2.2.3 *Dynamic/Incremental Approaches*

Finally, other recent approaches start to consider dynamic scenarios in which the dataset is updated over time with new instances.

In [82] the authors propose an evolving granular fuzzy-rule-based model designed for function approximation and time-series prediction in online contexts, where missing data can occur. This model, known as the evolving fuzzy granular predictor (eFGP), incorporates an incremental learning algorithm that not only imputes missing data but also adapts its model parameters and structure as the data evolves over time. The eFGP model is capable of addressing both single and multiple missing values within data samples, making it suitable for scenarios where data may be missing at random (MAR) or missing completely at random (MCAR) in nonstationary data streams. Unlike many traditional statistical and machine-learning methods, the eFGP is well-suited for environments characterized by concept drifts and shifts over time. It incrementally builds its rule base and computes fuzzy granule parameters and local linear functions based on the incoming data streams, even in the presence of missing data.

4.3 OPEN CHALLENGES AND RESEARCH GAPS

While recent metadata-driven imputation approaches have shown promising results, several research gaps still need to be addressed. Among these, the possibility to exploit RFDs to impute other types of data, such as time series, has not yet been explored. In particular, as in the relational databases scenario RFDs can be used to infer plausible candidates by describing relationships among columns/attributes, they could also be leveraged in multivariate time series imputation scenarios, in which data must be processed in terms of time steps and dimensions. However, to enforce RFDs in these scenarios, it is necessary to deal with two issues: (*i*) discovering RFDs is exponential with respect to the number of attributes, which corresponds to the number of dimensions in multivariate time series; and (*ii*) finding suitable imputation candidates is strictly related to the dataset. Consequently, while ensuring accurate imputations, RFD-based strategies leave a certain number of MVs unfilled, which is not suitable for several time series analysis approaches that require a fully observed dataset. These gaps will be addressed in the work discussed in Chapter 5.

In addition, as highlighted in Section 4.2, recent data imputation proposals are starting to consider incremental scenarios, in which datasets grow and evolve over time. However, the application of RFD-based imputation in these scenarios has not yet been explored. While in static scenarios, profiling metadata, such as Functional Dependencies (FDs) or Relaxed Functional Dependencies (RFDs), can be discovered from a complete batch of data and used to guide the imputation process, in incremental scenarios this approach is no longer sufficient. Both data and metadata evolve, and previously valid assumptions about data completeness and integrity may become outdated. This introduces two key challenges: (*i*) imputation becomes less reliable, as the metadata used to guide it might be incomplete or outdated, due to evolving data relationships, and (*ii*) profiling metadata discovery becomes more complex, as the presence of MVs in newly arriving data distorts such relationships. These issues will be investigated in the work discussed in Chapter 6.

5

DATA IMPUTATION FOR TIME SERIES DATA

The problem of missing values is particularly challenging in the data series scenario, since this type of data is typically collected from sensors. While recent years witnessed the introduction of several imputation methods tailored for this context, the application of RFD-based imputation has not yet been explored. As discussed in the previous chapter, leveraging RFDs in this scenario is not trivial, in particular due to the scalability issues of discovering RFDs on a large set of dimensions. This chapter will present a novel RFD-based imputation approach that includes optimizations tailored for the time series scenario, addressing the scalability issues of RFDs. More specifically, Section 5.1 will formalize the problem, while Section 5.2.1 will detail the proposed approach and the results of its evaluation.

5.1 PROBLEM OVERVIEW

Time Series data consists of sequences of ordered observations recorded over time, and can be categorized according to the dimensionality of data [25]. *Univariate* time series involve a single dimension, while *Multivariate* involve multiple ones:

Definition 19 (Multivariate Time Series). *A multivariate time series $T \in \mathbb{R}^{(D,n)}$ is defined as a sequence of n observations across D*

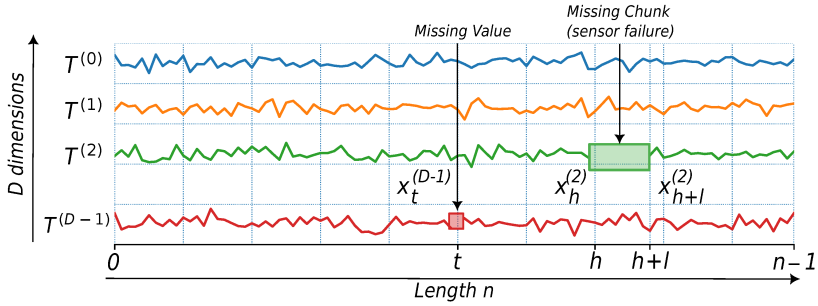


Figure 5.1: Time series with D dimensions and length n.

variables (i.e., dimensions), written $T = [T^{(0)}, T^{(1)}, \ldots, T^{(D-1)}]$*, where* $T^{(i)} = [T_0^{(i)}, T_1^{(i)}, \ldots, T_{n-1}^{(i)}]$ *represents the i-th dimension of* T*. The* t*-th observation of* T*, namely* x_t*, consists of* D *values observed at a timestamp* t*, i.e.,* $x_t \in \mathbb{R}^D$*.*

In real-world scenarios, some values of x_t may be missing due to communication errors or sensor failures. Specifically, given a dimension $d \in [0, D-1]$ and a time instant $t \in [0, n-1]$, the t-th observation on the d-th dimension, denoted as $x_t^{(d)}$, can be either *observed* (i.e., $x_t^{(d)} \in \mathbb{R}$) or *missing* (i.e., $x_t^{(d)} = _$). For instance, Figure 5.1 shows a multivariate time series in which the dimension $T^{(D-1)}$ has an MV at time t, i.e., $x_t^{(D-1)} = _$. Moreover, it can be noticed that there are consecutive MVs in $T^{(2)}$ between times h and $h+l$.

Multivariate time series often involve dimensions that exhibit underlying relationships among them. Thus, given a missing value on a dimension, such relationships, described through RFD$_c$s, can be leveraged to provide more precise imputations. In particular, we formalize the time series imputation with RFD$_c$s as follows:

Definition 20 (Imputing Multivariate Time Series with RFD$_c$s). *Given a multivariate time series* $T \in \mathbb{R}^{(D,n)}$ *and a set of* RFD$_c$s Σ *holding on* T *(*$T \vDash \Sigma$*), an imputation process* $I(T)$ *consists of finding,* $\forall A \in [0, D-1]$*,* $\forall t \in [0, n-1]$ *such that* $x_t^{(A)} = _$*, a time instant* $t' \in [0, n-1]$ *with* $x_{t'}^{(A)} \neq _$ *such that* $(x_t, x_{t'})$ *is similar w.r.t.* Φ_1*, according to* $\varphi : X_{\Phi_1} \rightarrow A_{\Phi_2}$*, and such that the imputation* $x_t^{(A)} = x_{t'}^{(A)}$ *yields* $T' = I(T)$ *to be semantically consistent, i.e.,* $T' \vDash \Sigma$*.*

5.2 RIFT: MULTIVARIATE TIME SERIES IMPUTATION [172]

Applying RFD$_c$s in multivariate time series scenarios is non-trivial due to the high volume of data and the large number of involved dimensions, as these scenarios typically incorporate readings from multiple sensors. To introduce a tailored solution to enable the effective use of RFD$_c$s, we consider 3 key observations and characteristics of time series:

- **Observation** 1**:** *Certain dimensions may be strongly correlated, while others can display substantial heterogeneity.* Thus, han-

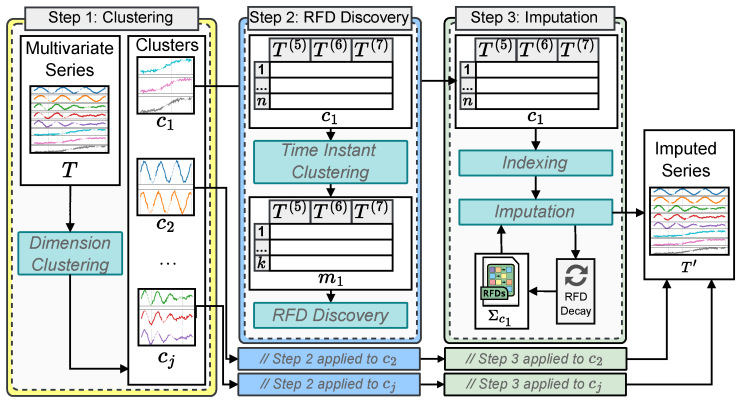


Figure 5.2: Overview of the RIFT Framework.

dling dimensions all together might be inefficient, redundant and, inaccurate.

- **Observation** 2**:** *Repetitive patterns frequently occur over time, possibly enabling a more compact representation of the series.* Considering all timestamps may lead to handling duplicated information, leading to inefficient computation.
- **Observation** 3**:** *The underlying phenomena captured by the time series may evolve over time, potentially altering data distributions and dependencies.* Therefore, dependencies initially leveraged to impute MVs might not be useful to impute the following time intervals.

We leverage these 3 observations to introduce a framework that enables the use of RFD$_c$s to inject high-quality imputations.

5.2.1 *The proposed approach*

This section describes the proposed approach (see Figure 5.2), detailing its three main steps for achieving efficiency and effectiveness, namely *Dimension Clustering*, RFD$_c$ *Discovery*, and *Imputation*. Throughout the section, we will refer to the running example illustrated in Figure 5.3, which depicts a small sample of the *Opportunity (Act.5)* dataset.

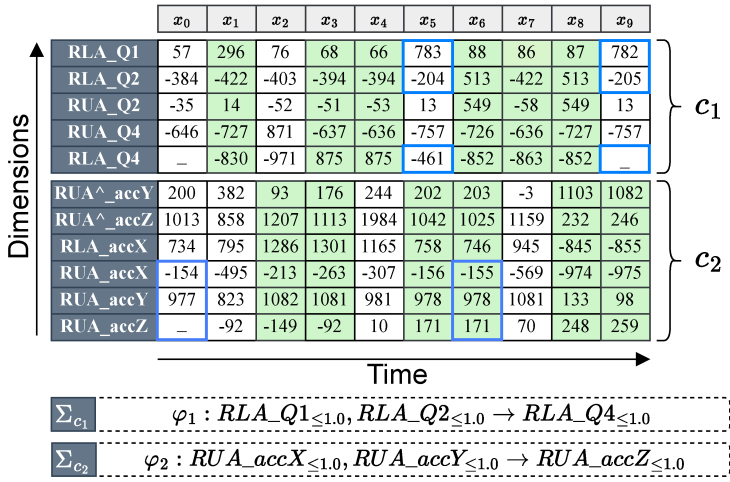


Figure 5.3: Snippet of the *Opportunity (Act.5)* dataset used for the Running Example.

5.2.1.1 *(Step 1) Dimension Clustering.*

One of the most important limitations of RFD$_c$ discovery algorithms is their lack of scalability on datasets with a large number of dimensions. However, according to **Observation** 1, it is possible to partition dimensions in groups with high intracorrelation and low intercorrelation. For this reason, the first step of the proposed approach, which is depicted in the yellow box of Figure 5.2, splits multivariate time series into clusters to perform both RFD discovery and imputation on a limited number of correlated dimensions. Formally, consider a multivariate time series $T \in \mathbb{R}^{(D,n)}$, the first step of RIFT entails partitioning T into j clusters $\mathcal{C} = \{c_1, c_2, \ldots, c_j\}$, where each $c_i \subseteq \{T^{(0)}, T^{(1)}, \ldots, T^{(D_i-1)}\}$ is a subset of the dimensions of T, and the union of all clusters covers the entire set of dimensions, i.e.,

$$\bigcup_{i=1}^{j} c_i = \{T^{(0)}, T^{(1)}, \ldots, T^{(D-1)}\}, \quad c_i \cap c_h = \emptyset \text{ for } i \neq h. \tag{5.1}$$

Since RFD$_c$s capture relationships among columns/dimensions, in this step we aim to obtain clusters composed of dimensions that are strictly correlated with each other, thus enabling the

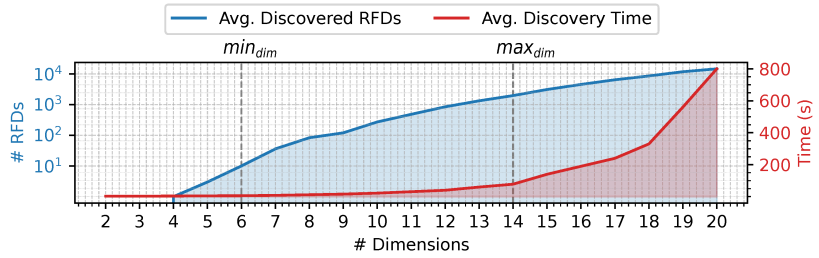


Figure 5.4: Average number of discovered RFDs and discovery time by varying the number of dimensions.

discovery of RFD_cs. To achieve this, we consider all the complete time instants to compute the *Pearson* correlation between pairs of dimensions $T^{(d)}$, $T^{(d')}$ (with $d, d' \in [0, D-1]$), and use the complement of these correlations as distances.

After computing the *Pearson* correlation among dimensions, we perform a hierarchical agglomerative clustering process based on the Ward linkage criterion [181], adapted to consider correlations as distances. This kind of point-wise clustering approach is more suited for multivariate time series, in which data is usually collected from different sensors in a synchronized manner, rather than using shape-based approaches [94], which instead are better at capturing similarities between series with temporal shifts. Nevertheless, we empirically evaluated other alternatives to the *Pearson* correlation, including the non-linear rank-based *Kendall* correlation, as well as the *Shape-Based Distance* (SBD) and the alignment-based *Dynamic Time Warping* (DTW). The results, available in Section 5.2.2, confirmed that RIFT provides more precise imputations when employing the *Pearson* and *Kendall* correlations for its first Step, denoting a better partitioning of the dimensions, with the former measure obtaining the best results. This outcome can be motivated by the fact that the imputations performed by RIFT are also point-based, making these correlations more suitable for partitioning dimensions.

After applying clustering, there might be clusters with a large number of dimensions and others with only a few of them. In the first case, the scalability problems of RFD_c discovery algorithms still persist, while in the second one, it might be unlikely to infer meaningful relationships. Thus, we introduce two parameters, namely max_{dim} and min_{dim}, which define the maximum

and minimum number of dimensions allowed in each cluster, respectively. Formally, each cluster c_i must satisfy the condition $min_{dim} \leq |c_i| \leq max_{dim}, \forall i \in [1, j]$. Thus, if one or more clusters contain a number of dimensions that exceed max_{dim}, we further split these clusters. On the other hand, small clusters are merged with neighboring clusters, provided that the max_{dim} constraint is satisfied. To define min_{dim} and max_{dim}, we performed a preliminary experiment over different datasets commonly used in Data Profiling[1] by randomly varying the number of dimensions from 2 to 20, considering the average number of discovered RFDs and the average discovery time (see Figure 5.4). It is possible to notice that with 6 or more dimensions several RFDs start to emerge, and that with more than 14 dimensions the discovery times start to rise more steeply. As a result, for our approach, we fix $min_{dim} = 6$ and $max_{dim} = 14$. The output of the described *Step1* consists of j clusters of dimensions (i.e., c_i, $i = 1, 2, \ldots, j$), which can be independently processed in the following two steps.

Example 5.2.1. Consider the sample of the *Opportunity* dataset shown in Figure 5.3. For the sake of illustration, we do not consider the min_{dim} constraint given the reduced size of the example. Suppose that, after applying the clustering step, RIFT partitions the time series into two clusters c_1 and c_2. The former contains quaternion-based features, capturing movements of the right upper (RU) and right lower (RL) arm, while the latter includes accelerometer measurements from the right arm, reflecting coordinated linear motions along different axes.

5.2.1.2 *(Step 2)* RFD_c *Discovery.*

For each of the j clusters, the second step of RIFT, which is depicted in the blue box of Figure 5.2, aims to discover the set of RFD_cs holding on it. For performing the discovery of RFD_cs, we leverage INDIBITS [28], a recent approach that exploits an efficient binary representation of data similarities. However, the proposed strategy is not tied to any specific discovery algorithm. To further optimize discovery times, according to **Observation** 2, RIFT extracts RFD_cs only on a subset of representative time

1 https://github.com/DastLab/TestDataset

	Opportunity (Act. 3)			Opportunity (Act. 5)			Telemetry		
ID	1	2	3	4	5	6	7	8	9
Length	10.000	15.000	20.000	10.000	15.000	20.000	10.000	15.000	20.000
Dim.	30	60	130	30	60	130	8	8	8
Type	Human Activity			Human Activity			Sensor (Temperature)		
MVs (5%)	15.000	45.000	130.000	15.000	45.000	130.000	4.000	6.000	8.000
MVs (10%)	30.000	90.000	260.000	30.000	90.000	260.000	8.000	12.000	16.000
MVs (20%)	60.000	180.000	520.000	60.000	180.000	520.000	16.000	24.000	32.000
	Genesis			Air Quality			CATS		
ID	10	11	12	13	14	15	16	17	18
Length	15.538	15.538	15.538	3.000	6.000	9.000	10.000	13.000	16.000
Dim.	10	15	18	13	13	13	17	17	17
Type	Sensor (Industrial)			Sensor (Air Pollution)			Sensor (Industrial)		
MVs (5%)	7.769	11.654	13.984	1.950	3.900	5.850	8.500	11.050	13.600
MVs (10%)	15.538	23.307	27.968	3.900	7.800	11.700	17.000	22.100	27.200
MVs (20%)	31.076	46.614	55.937	7.800	15.600	23.400	34.000	44.200	54.400

Table 5.1: Details of the datasets employed in the evaluation and number of MVs injected in each configuration.

instants that better represent the overall dynamics of the series. Thus, we use the KMedoids [146] algorithm, which considers actual data points, namely *Medoids*, as cluster representatives. We apply it by considering only complete time instants in each cluster, in order to perform the discovery process on a clean portion of data. Formally, for each cluster $c_i \in \mathcal{C}$, we select k representative time instants and form a multivariate time series $m_i = \{T_{m_i}^{(0)}, T_{m_i}^{(1)}, \ldots, T_{m_i}^{(D_i-1)}\}$, where each $T_{m_i}^{(j)}$ is a series of length k extracted from the corresponding dimension $T^{(j)}$ and D_i is the number of dimensions in m_i. For RIFT, we set $k = 500$ (see Section 5.2.2.3). Finally, we perform RFD_c discovery on the subsets m_i to obtain j sets of RFD_cs, i.e., Σ_{c_i}, $i = 1, 2, \ldots, j$, where j is the number of clusters.

Example 5.2.2. Let us consider the clusters c_1 and c_2 obtained from the Step 1 clustering (see Example 5.2.1). For the sake of illustration, we set $k = 6$ given the reduced size of the example. RIFT applies the KMedoids algorithm on each cluster, considering only the complete time instants. Thus, the timestamps x_0 and x_9 are not considered for c_1, while x_0 is not considered for c_2.

Suppose that the following instants are selected as the most representative ones:

$$m_1 = \{x_1, x_3, x_4, x_6, x_7, x_8\} \quad m_2 = \{x_2, x_3, x_5, x_6, x_8, x_9\}$$

For each cluster, we perform RFD_c discovery only on its representative time instants (i.e., m_1 for c_1 and m_2 for c_2), which are highlighted in green in Figure 5.3, to obtain the sets of RFD_cs Σ_{c_1} and Σ_{c_2}. Suppose that, at the end of the discovery process for m_1, the set Σ_{c_1} is composed of a single dependency:

$$\varphi_1 : \mathsf{RLA_Q1}_{\leq 1.0}, \mathsf{RLA_Q2}_{\leq 1.0} \rightarrow \mathsf{RLA_Q4}_{\leq 1.0}$$

Similarly, suppose that, at the end of the discovery process for m_2, the set Σ_{c_2} is composed of a single dependency:

$$\varphi_2 : \mathsf{RUA_accX}_{\leq 1.0}, \mathsf{RUA_accY}_{\leq 1.0} \rightarrow \mathsf{RUA_accZ}_{\leq 1.0}$$

5.2.1.3 *(Step 3)* RFD_c*-based Imputation.*

The third step of RIFT, which is depicted in the green box of Figure 5.2, aims to impute each cluster c_i by leveraging the RFD_c set Σ_{c_i} and by employing the RENUVER imputation algorithm. To improve the execution times of the imputation process, we introduce two optimizations. The first one is the use of an indexing strategy to efficiently retrieve potential candidates. In particular, we build a B-tree index for each dimension, in order to find the time instants that are similar according to the considered RFD_c. This strategy is not applied with boolean values, as we observed significant slowdowns. This is due to the fact that the values of a boolean dimension can be of only two types, leading the B-tree to have only two leaves with a large list of pointers. As a result, there is no advantage in retrieving candidates through the index, since the lookup cost becomes comparable to, or even higher than, a sequential scan. The second optimization is derived from **Observation 3**. According to it, while an RFD_c may initially contribute to the imputation process, its relevance can diminish over time. In such cases, iterating through all RFD_cs for each MV may become inefficient. To address this, we introduce a *time decay* mechanism, which gradually reduces the lifetime of RFD_cs during imputation. Formally, after RENUVER attempts to

impute a dimension A of a time step t_j (i.e., $t_j[A] = _$), all the RFD$_c$s $\varphi : X_{\Phi_1} \rightarrow A_{\phi_2}$ that were not used to impute it will have their lifetime reduced by a value α, namely the *decay rate*. If the lifetime of an RFD$_c$ reaches zero, then it is removed from Σ_{c_i}. Instead, whenever an RFD$_c$ is used for imputing a missing value, its lifetime is reset to 1. For RIFT, we set $\alpha = 0.05$ (see Section 5.2.2.3). After all the clusters are imputed, we merge them to form the (partially) imputed time series. RIFT can be used as preprocessing for injecting high-quality values before using any other imputation approach.

Example 5.2.3. Consider the clusters c_1 and c_2 obtained from the Step 1 clustering (see Example 5.2.1, and their associated sets of RFD$_c$s Σ_{c_1} and Σ_{c_2} obtained from Step 2 (see Example 5.2.2). Since the *Opportunity* dataset is composed of real values, before starting the imputation process RIFT builds an index for each dimension, in order to improve the execution times of the candidate retrieval. Then, it starts the imputation process by considering each cluster and its associated RFD$_c$ set, searching for suitable imputation candidates for each missing value. For example, during the imputation process of c_1, RIFT first considers the missing value $x_0^{(\mathsf{RLA_Q4})} = _$, and searches for a candidate according to the only RFD$_c$ that belongs to Σ_{c_1}, i.e., $\varphi_1 : \mathsf{RLA_Q1}_{\leq 1.0}, \mathsf{RLA_Q2}_{\leq 1.0} \rightarrow \mathsf{RLA_Q4}_{\leq 1.0}$. However, no suitable candidate can be inferred from c_1, since there not exists a time instant x_i that has similar values on RLA_Q1 and RLA_Q2 w.r.t. x_0. Thus, in this case the lifetime of φ_1 is decreased by 0.05. Then, RIFT considers the missing value $x_9^{(\mathsf{RLA_Q4})} = _$. In this case, by querying the indexes built on RLA_Q1 and RLA_Q2, RIFT identifies x_5 as a suitable candidate. Thus, it performs the imputation of $x_9^{(\mathsf{RLA_Q4})}$ by using the value in $x_5^{(\mathsf{RLA_Q4})}$, i.e., $x_9^{(\mathsf{RLA_Q4})} = -461$. Thus, in this case the lifetime of φ_1 is set to 1. Similarly, during the imputation process of c_2, RIFT considers the missing value $x_0^{(\mathsf{RUA_accZ})} = _$, and searches for a candidate according to the only RFD$_c$ that belongs to Σ_{c_2}, i.e., $\varphi_2 : \mathsf{RUA_accX}_{\leq 1.0}, \mathsf{RUA_accY}_{\leq 1.0} \rightarrow \mathsf{RUA_accZ}_{\leq 1.0}$. In this case, by querying the indexes built on RUA_accX and RUA_accY, RIFT identifies x_6 as a suitable candidate. Thus, it performs the

imputation of $x_0^{(\text{RUA_accZ})}$ by using the value in $x_6^{(\text{RUA_accZ})}$, i.e., $x_0^{(\text{RUA_accZ})} = 171$.

5.2.1.4 *Complexity.*

The complexity of RIFT is tied to the ones of RFD-discovery algorithms and RFD-based imputation approaches. However, the introduced optimizations allow for reducing the terms involved. Specifically, given a multivariate time series T of length n with D dimensions, the RFD discovery algorithm has to consider all possible z combinations of attributes, with $z \in [2, D]$. By partitioning T into j clusters, the discovery complexity for each cluster $c_i \in [1, j]$ is:

$$O(\sum_{z=2}^{D_i} \binom{D_i}{z} \cdot z)$$

As for the imputation, the complexity is :

$$O(n^2 \cdot D_i \cdot (|\Sigma_{c_i}| - w) \cdot (v \cdot D_i \cdot (|\Sigma_{c_i}| - w) \cdot v \log v))$$

where $|\Sigma_{c_i}|$ is the number of RFD_cs holding on c_i, v is the average number of candidates for each MV, and w is the average number of RFD_cs discarded by the decay mechanism. The latter can significantly reduce imputation times in scenarios involving a large number of RFD_cs, enabling to retain only the ones contributing to the imputation (see Section 5.2.2.1).

5.2.2 *Experimental Evaluation*

This section details the evaluation of RIFT[2]. All experiments were conducted on a system with an AMD Ryzen R9 5950X processor and 128GB of RAM. For our evaluation, we consider 6 datasets covering different domains (detailed in Table 5.1). These datasets represents scenarios with both a small (i.e., *Telemetry* and *Air Quality*) and a large (i.e., the two *Opportunity* datasets) number of dimensions, in order to evaluate RIFT in different conditions.

2 Supplementary materials related to this work are available on the official repository

Moreover, for each dataset, we consider 3 configurations by varying the length and/or the number of dimensions. For each configuration, we artificially injected MVs by following the concept of Missing Completely at Random (MCAR), considering missing rates of 5%, 10%, and 20% with respect to the total number of values of each dataset, for a total of 54 configurations. No additional preprocessing steps (e.g., normalization) are applied, as the primary goal is to impute MVs based on the original data. Finally, to evaluate a more realistic and challenging scenario, we also conducted a supplementary experiment on the *Opportunity (Act. 5)* dataset by considering a different missing mechanism. In particular, we simulated sensor failures for random periods of time (see the missing chunk in Figure 5.1), leading to the injection of consecutive MVs until reaching a missing rate percentage of 5%, 10%, and 20%

In the **first part of our experiments**, we use all configurations to measure the execution time improvements of RIFT with respect to a traditional RFD-based approach. We first evaluate the effect of our clustering strategy (Step 1 and 2). We then evaluate the imputation times with and without the indexing strategy and the *time decay* mechanism of RIFT. For these experiments, we set a *Time Limit* of 5 hours. For the following parts of our evaluation, we only consider the larger configurations of each dataset (IDs 3, 6, 9, 12, 15, 18). In the **second part of our experiments**, we validate the choice of using the *Pearson* correlation for RIFT's Step 1 clustering against other alternatives, namely *Kendall*, SBD, and DTW, comparing the imputation performance of RIFT when adopting these measures. The **third part of our experiments** aims to investigate how the parameters k and α impact on the discovery times and on the imputations performed by RIFT. For evaluating imputations, we consider *Precision*, *Recall*, and *Root Mean Square Error* (*RMSE*). *Precision* represents the percentage of MVs correctly imputed with respect to the number of imputations performed. *Recall* measures the percentage of MVs correctly imputed with respect to the total number of MVs. *RMSE* quantifies the magnitude of errors between the original and imputed values. Formally, let y_i and $\hat{y}_i$ be the original and the imputed value, tr be the number of correct imputations, imp the number of imputations performed and $miss$ be the number of MVs orig-

inally present in the dataset. *Precision*, *Recall* and *RMSE* can be defined as follows:

$$\textbf{\textit{Precision}} = \frac{|tr \cap imp|}{|imp|}, \quad \textbf{\textit{Recall}} = \frac{|tr \cap miss|}{|miss|}, \quad \textbf{\textit{RMSE}} = \sqrt{\frac{1}{imp}\sum_{i=1}^{imp}(y_i - \hat{y}_i)^2}$$

It is worth pointing out that *imp* and *miss* are two different variables. As a matter of fact, not all the imputation algorithms provide estimations for all the missing values. For example, it is not possible to apply the *Mean* to impute a dimension with textual values. Similarly, as previously explained, RFD$_c$-based imputation provides estimations only if it finds a suitable, high-quality candidate found through the imputation process. To compute the number of correct imputations (i.e., *tr*) in all the performed evaluations, we employed the same rule-based framework used in [30], which allows to introduce a small margin of error defined according to the value distribution of each dimension. This is due to the fact that time series data is typically characterized by the presence of continuous numbers, and determining the correct imputations through exact matching may result in an underestimation of *Precision* and *Recall*. For instance, on the *Telemetry* dataset, we defined a margin of error of 0.005 for the *lpg* dimension, which reports the gas concentration registered by the sensors. While this solution enables to provide a more reliable quantitative evaluation that helps to understand the general behaviour of the compared configurations, with the *RMSE* we also analyze the distance between the imputations and the ground-truth values, leading to a complete evaluation.

Finally, the **fourth part of our experiments** evaluates the support that RIFT provides to other state-of-the-art time series imputation approaches. Among these, we considered popular imputation methods belonging to different categories. Among *Simple Imputation* approaches, we included Mean, Median, and the commonly used Backward and Forward Fill. Among *Machine Learning* methods, we considered MICE, while for *Interpolation* techniques we employed the cubic Spline. Finally, we included some recent popular deep learning approaches provided by the PyPOTS library [67], i.e., TIMESNET [184], BRITS [32], and SAITS [68]. Since these models require the definition of several hyperparameters, we relied on the ones provided in the PyPOTS documentation, and we set the number of training epochs to 180.

5.2.2.1 *Scalability Evaluation and Ablation Study*

We first evaluate the scalability of RIFT with an ablation study to evaluate the effect of each of its optimizations on (i) RFD_c discovery and (ii) RFD_c-based imputation.

DISCOVERY. Figure 5.5 shows the RFD_c discovery times with and without applying the two optimizations of RIFT (i.e., Step 1 and 2), as well as the ones when applying them individually. For the configurations that apply Step 2, we set k=500. We can observe that the "Step 1 only" configuration allows for significantly reducing the execution times for datasets with a large number of dimensions (i.e., *Opportunity*, *Genesis* and *CATS*), with average improvements from 48.29% to 72.59%. On the other hand, this optimization does not affect *Telemetry* and *Air Quality*, since they have a limited number of dimensions that are arranged in a single cluster. As regards the "Step 2 only" configuration, we can observe that it has even more impact on the discovery times on all datasets, with average improvements from 68.51% to 98.05%. Finally, when combining Step 1 and Step 2, we can observe a further reduction of execution times, with average improvements w.r.t. the discovery without optimizations from 68.51% to 98.92%. Figure 5.5 also reports the average scale-up ratio (discovery without optimizations vs RIFT) on the different missing rates. We observed that it tends to rise as the size of the series increases, demonstrating that RIFT is significantly more scalable than a traditional RFD discovery process. Moreover, since we perform discovery only on the complete portion of data, the discovery times tend to decrease when missing rate increases. Finally, since we were able to extract RFD_cs in all clusters, we can say that RIFT's clustering strategy preserved relevant relationships while significantly improving the efficiency of the discovery process.

IMPUTATION. Figure 5.6 shows the imputation times with and without applying the two optimizations of RIFT (i.e., indexing and time decay), as well as the ones when applying them individually. For this comparison, we perform imputation by using the RFD_cs extracted by applying the clustering strategy, with k=500. In addition, it is also worth pointing out that, for the configu-

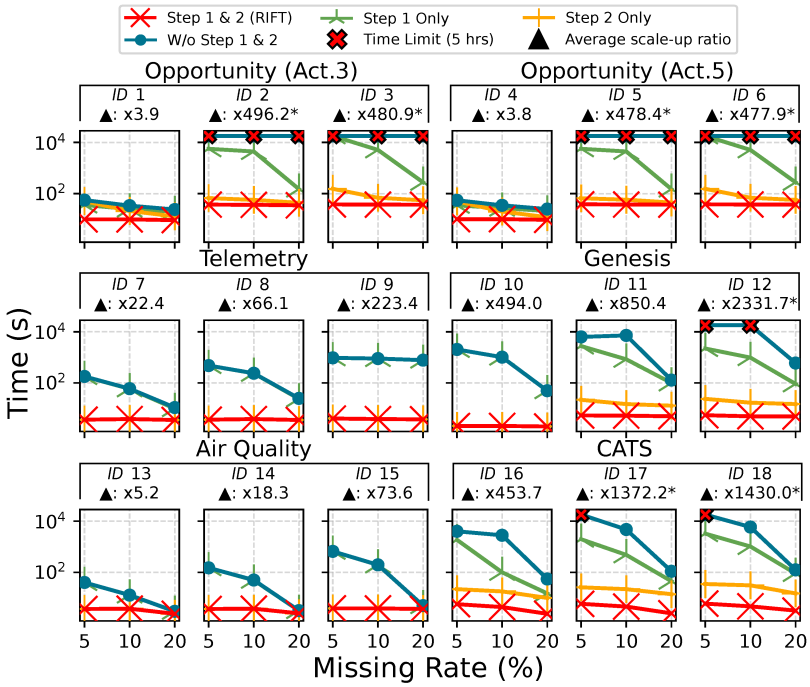


Figure 5.5: Discovery times with and without RIFT's optimizations. Scale-ups computed between RIFT (red) and RFD_c discovery without optimizations (blue). Asterisks mark scale-ups computed considering the TL.

rations that leverage indexing, the values reported include the time required for building the indexes. We can observe that the "Indexing only" configuration had a significant impact in almost all datasets, with average time improvements ranging from 25.52% to 65.64%. The only exception is for *Genesis*, in which most dimensions are boolean values and thus, as discussed in Section 5.2.1.3, no indexes are built upon them. As regards the "Decay only" configuration, we can observe that its impact varies according to the datasets. For *Telemetry* and *Genesis* the impact of time decay is limited, with an average reduction of 13.87% and 3.12%, respectively. This occurred because there were not many holding RFD_cs, and the majority of them were regularly employed throughout the imputation process. Instead, as shown in Figure 5.6, the decay mechanism led to noteworthy improvements for the other datasets. The latter had a large number of holding RFD_cs that, without decay, were constantly considered

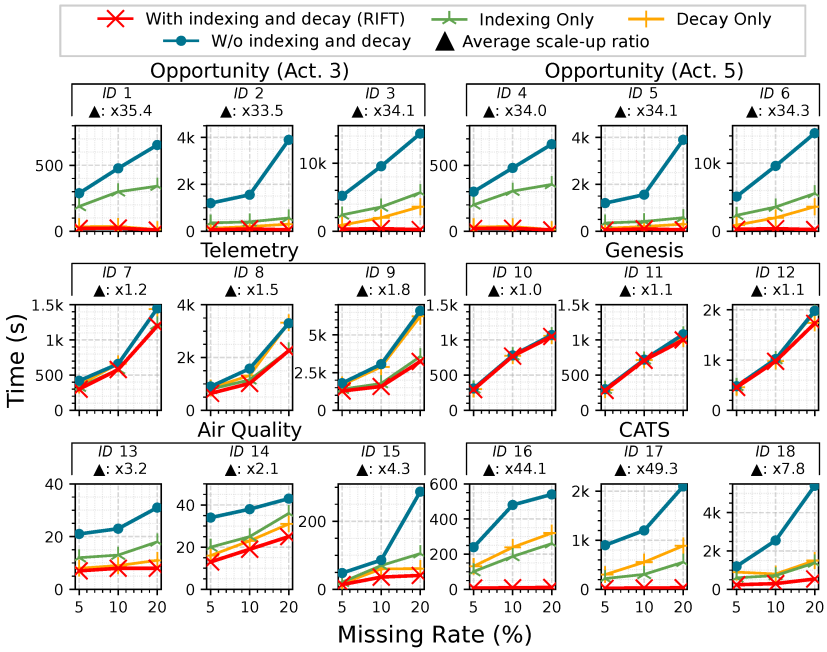


Figure 5.6: Imputation times with and without applying RYAN's optimizations. The scale-ups are computed between RYAN (red) and the imputation without optimizations (blue).

when searching for a suitable candidate, even if most of them never led to an imputation. With the time decay mechanism, we were able to discard those RFD_cs that were not contributing to the imputation, with average improvements ranging from 53.38% to 87.19%. Finally, when combining indexing and decay, we can observe a further reduction of imputation times, with average improvements (excluding *Genesis* and *Telemetry*) ranging from 63.82% to 95.9%.

5.2.2.2 *Point-based vs Shape-based clustering.*

As discussed in Section 5.2.1, the Step 1 of the RIFT framework consists in partitioning multivariate time series in clusters according to their correlation. For the latter, RIFT employs the *Pearson* correlation as distance used in the clustering process. To validate our choice, we compared RIFT with other variants that consider the Kendall correlation coefficient [106], the *Shape-Based Distance*

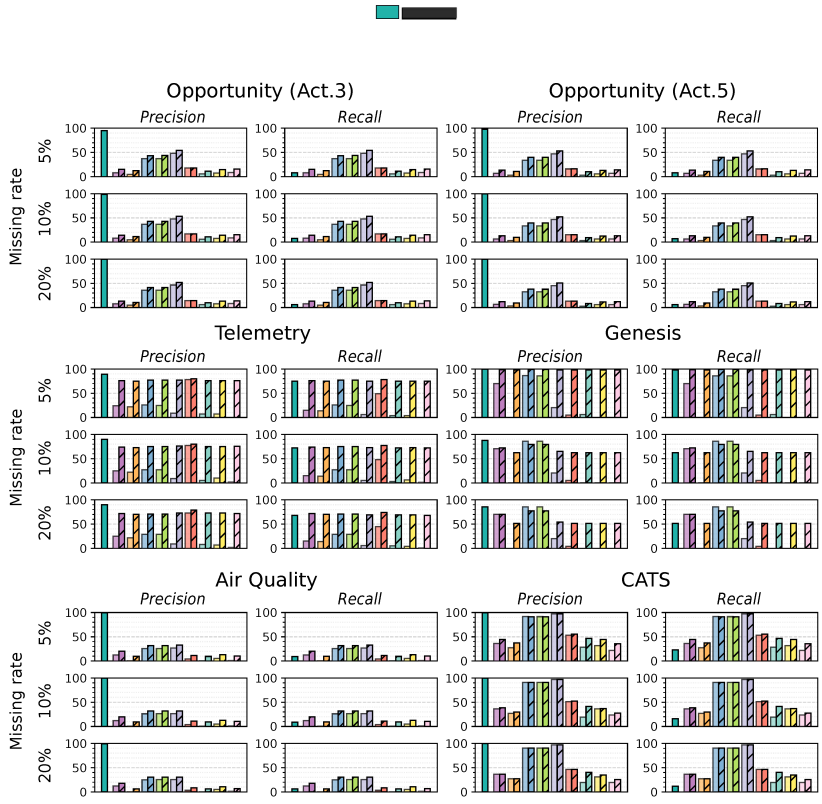


Figure 5.7: Impact of the proposed approach in terms of *Precision* and *Recall*.

(SBD) [142], and the Dynamic Time Warping (DTW) distance [22] with the Sakoe-Chiba bounding strategy [161], respectively. We compared the *Precision* and *Recall* of these variants with the ones achieved with the *Pearson* correlation across the *Opportunity*, *Genesis*, and *CATS* datasets, which are the ones whose number of dimensions is higher than the max_{dim} value defined in Section 5.2.1, requiring the application of the Step 1 optimization. The results are shown in Table 5.2. In general, while RIFT exhibited a similar behaviour in all the different configurations, it is possible to notice some differences. By considering the *Precision*, the variant of RIFT with the *Pearson* correlation RIFT achieved the highest scores (96.75% on average), denoting a better partitioning of the dimensions that allowed to find candidates of better

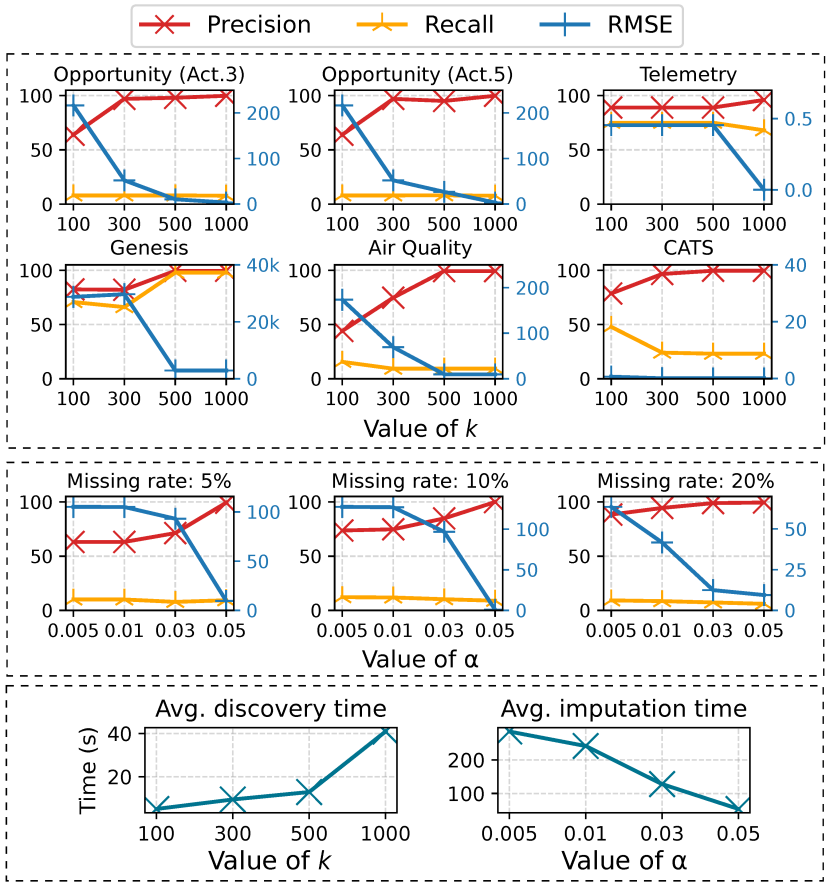


Figure 5.8: Influence of k (upper box) and α (middle box) on the imputation accuracy (the right axis in blue is for the RMSE), and on the execution times (lower box).

quality for the imputation. The second best variant was the one employing the *Kendall* correlation (90.93% on average), followed by the ones that leverage SBD and DTW (90.6% and 90.5%, on average, respectively). By considering the *Recall*, the *Pearson* correlation again enabled to achieve the highest scores (25.43% on average) with respect to Kendall, SBD, and DTW (22.4%, 19.74%, and 22.41%, on average), even if in some configurations the latter achieved better results. In these few cases, the variants provided an higher number of correct imputation but, as proved by the *Precision* values, at the cost of injecting more errors within the series. Overall, these results validate our choice of relying on the

Pearson correlation, as it allows RIFT to provide higher-quality imputations.

5.2.2.3 *Influence of the Number of Medoids (k) and decay rate (α)*

In this section, we evaluate the influence of the k and α parameters on both the efficiency and imputation accuracy of RIFT.

INFLUENCE OF k. We evaluate the discovery times and the imputation accuracy with different numbers of medoids, i.e., $k \in \{100, 300, 500, 1000\}$. Figure 5.8 shows the impact of k on both discovery times and imputation accuracy for the datasets with missing rate of 5% (we observed similar results for the other missing rates, the full results are available in our repository). As expected, the execution times increase with higher values of k. As for the imputation accuracy, we observe that, with lower values of k, the imputation algorithm provides more estimations but it is less accurate, resulting in lower *Precision* and slightly better *Recall*. Instead, higher values of k imply a higher *Precision* and slightly lower *Recall*. Moreover, *RMSE* tends to decrease with higher k values, indicating higher-quality imputations. Since there is no evident difference between k=500 and k=1000, we consider the former as a suitable trade-off between imputation performance and discovery times.

INFLUENCE OF α. We now evaluate RIFT imputation times and accuracy for different values of α, i.e., $\alpha \in \{0.005, 0.01, 0.03, 0.05\}$. As depicted in Figure 5.8, the imputation times decrease as α increases, as more RFD$_c$s are discarded during the imputation process. Moreover, higher values of α lead to more accurate imputations, improving both *Precision* and *RMSE*. This is because the discarded RFD$_c$s are typically those ranked lower in the selection process of RFD$_c$s. Since higher-ranked RFD$_c$s are prioritized, less relevant ones tend to remain unused for longer periods. With a more aggressive decay, these lower-ranked RFD$_c$s are discarded earlier, thereby reducing the chance of relying on less effective RFD$_c$s. Finally, *Recall* slightly lowers with higher values of α, as fewer RFD$_c$s lead to fewer estimations. Since the purpose of RIFT is to inject high-quality imputations, we adopt the highest α.

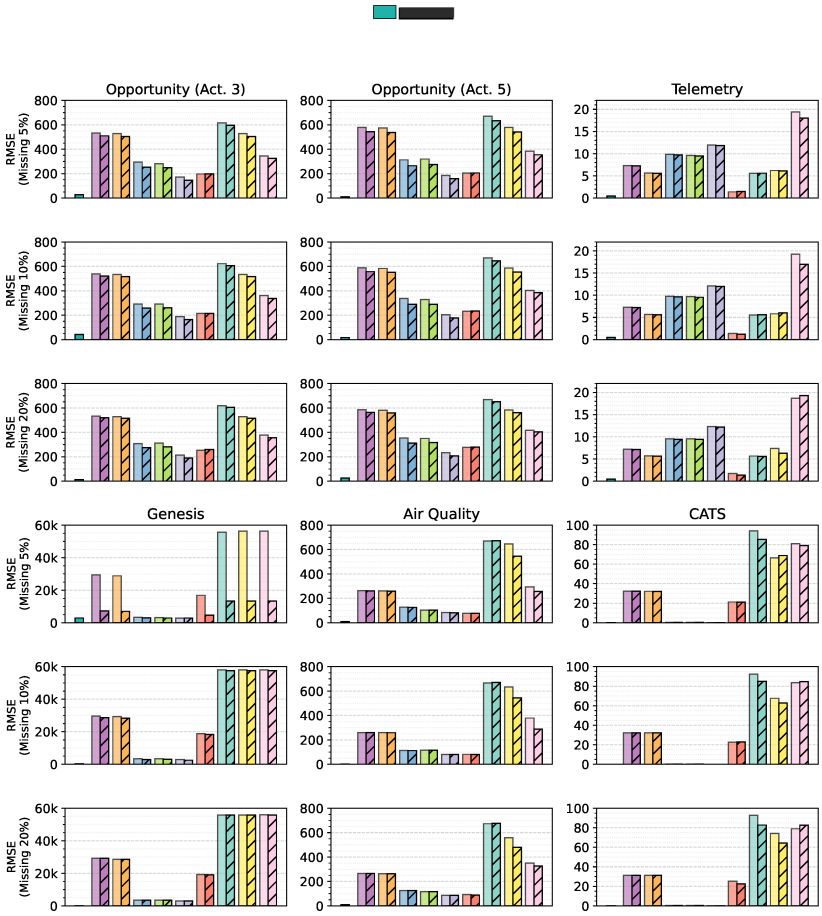


Figure 5.9: Impact of RIFT in terms of *RMSE*

Overall, although RIFT relies on two parameters (k and α), in our extensive evaluation, the adopted parameter values (k=500 and α=0.05) proved to be stable and effective across different experimental conditions, and do not require adjustment for different settings.

5.2.2.4 *Comparison with the State-of-the-art*

Figure 5.7 shows the *Precision* and *Recall* of the baselines with and without applying RIFT as a preprocessing step. Moreover, the performance of RIFT alone is also reported. By considering the latter, we can see that it always achieved higher *Precision*

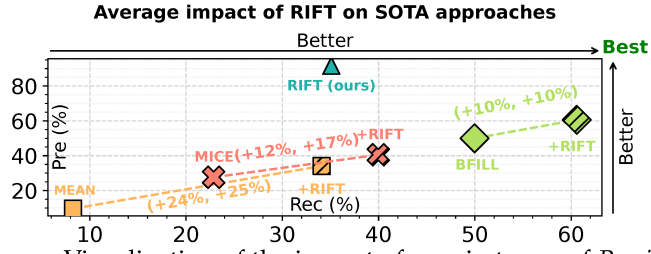


Figure 5.10: Visualization of the impact of RIFT in terms of *Precision* and *Recall*.

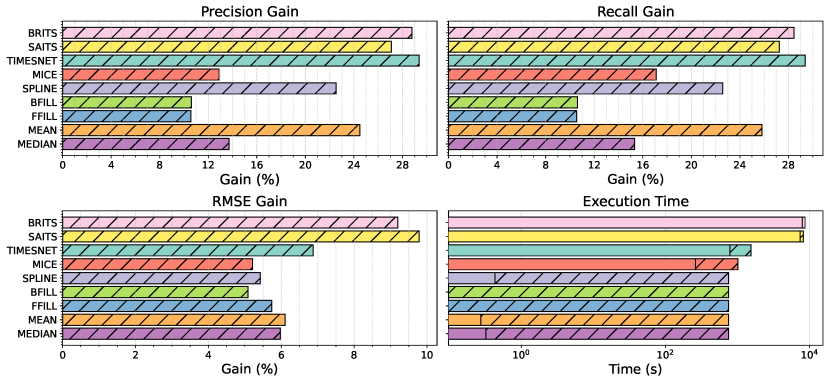


Figure 5.11: Average gain when applying RIFT as preprocessing step in terms of *Precision*, *Recall*, *RMSE*, and execution times.

than the baselines (from 85.78% to 99.9%). This suggests that the optimizations introduced did not affect the inherently high precision of RFD-based imputation, allowing to detect meaningful relationships and, as a consequence, high-quality candidates. As for the baselines, only few approaches could reach (without the application of RIFT as preprocessing) a *Precision* higher than 80%, but only in a limited number of experiments. SPLINE was the best one on both the *Opportunity* datasets (with maximum *Pre.* of 48.2%), while MICE was the best on *Telemetry* (with maximum *Pre.* of 78%). Finally, Backward and Forward fill were the best for *Air Quality* (maximum *Pre.* of 26.0%), *CATS* (maximum *Pre.* of 97.1%), and *Genesis* (maximum *Pre.* of 86.2%), while SPLINE was the best baseline for *CATS* (maximum *Pre.* of 97.1%). In general, it is possible to notice that none of the baselines exhibited a consistent behavior across the different datasets, showing important variations in their *Precision*. Notice that, the *Precision* and the *Recall* of baselines are always the same in each configuration,

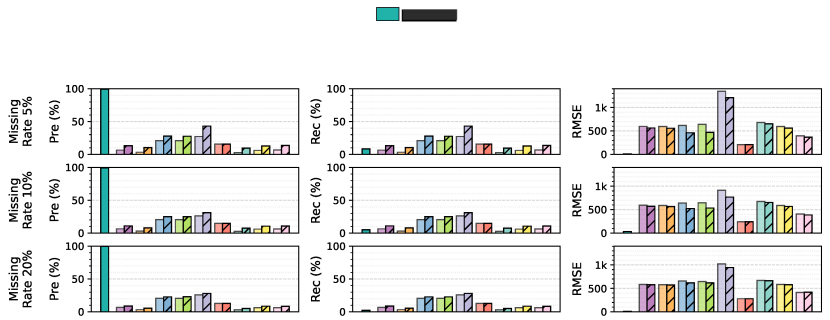


Figure 5.12: Impact of RIFT in terms of *Precision*, *Recall*, and *RMSE* when considering MVs caused by sensor failures on the *Opportunity (Act. 5) dataset*.

since they impute all the MVs within datasets. An exception is registered for the *Telemetry* dataset, which contains some dimensions with textual values that, in contrast to our approach, not all baselines could impute. On the other hand, the *Recall* achieved by RIFT is rather low. This is because RFD-based algorithms only impute a value if they find a suitable candidate, resulting in a low overall number of performed imputations. This behavior is particularly evident for *Opportunity* and *Air Quality*. However, by considering the perfomance of the baselines when applying RIFT as preprocessing step, we can already notice that this limited number of high-quality imputations allows to significantly improve both *Precision* and *Recall* in almost all configurations. This proves that RIFT is able to properly impute MVs that other approaches cannot correctly estimate. Moreover, in contrast to the baselines, RIFT exhibited a consistent behaviour in terms of *Precision*, meaning that it was always capable of providing such high-quality imputations. By averaging the improvements over all baselines, the application of RIFT increased both *Precision* and *Recall* by 5.34%, 5.48%, 45.19%, 6.83%, and 4.73% for *Opportunity (Act.*3), *Opportunity (Act.*5), *Genesis*, *Air Quality*, and *CATS*, respectively. For the *Telemetry* dataset, we recorded average improvements of 52.45% and 57.16% for *Precision* and *Recall*, respectively. Figure 5.10 summarizes the overall *Precision* and *Recall* improvement across all datasets, visually showing how RIFT shifts baseline approaches toward the optimal region. For

visual clarity, we show only a representative subset of baselines (i.e., *Mean*, MICE, and BFILL).

We also investigate the quality of the imputations by analyzing the *RMSE*. Figure 5.9 shows the results in terms of RMSE scores of the baselines with and without the application of the proposed framework, as well as the RMSE of RIFT alone. We can see that the *RMSE* of RIFT is significantly lower w.r.t. the other baselines, highlighting the closeness of its estimations to the ground truth values. While the *RMSE* of the baselines is very high, by applying RIFT as a pre-processing step we observed improvements in almost all configurations. On average, by applying RIFT, the best *RMSE* reductions were observed on the *Genesis* and *Opportunity* datasets, with improvements of 18.91% and 6.98%, respectively. Notably, even in datasets where RIFT performed fewer imputations (i.e., *Opportunity* and *Air Quality*), it still improves the baselines' performance.

To summarize, Figure 5.11 shows the average improvement of *Precision*, *Recall*, and *RMSE* per baseline. We can see that RIFT significantly enhanced both *Precision* and *Recall*, with gains ranging from 10.56% to 29.38%. Furthermore, RIFT brought significant improvement for *RMSE*, with gains ranging from 5.08% to 9.79%. Figure 5.11 also presents the average execution times of the baselines with and without RIFT across all datasets. The execution time of RIFT falls between the faster baselines and the two slowest methods, BRITS and SAITS. However, we claim that the additional overhead required to perform RIFT is justified by the significant improvement in imputation accuracy.

IMPUTATION PERFORMANCE WITH OTHER MISSING MECHANISMS. In addition to our main experiments conducted under the MCAR assumption, we performed a complementary evaluation by simulating sensor failures for a random period of time, until reaching a missing rate percentage of 5%, 10%, and 20%. Figure 5.12 shows the results in terms of *Precision*, *Recall*, and *RMSE*. By comparing the latter with those under the MCAR assumption, it is possible to generally notice a similar behavior of all approaches, even with a slight degradation due to the more complex scenario. However, some of the baselines showed a significant degradation of their

	Opportunity (Act. 3)						Opportunity (Act. 5)					
Missing Rate	5%		10%		20%		5%		10%		20%	
	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec
Pearson (ours)	95.0	8.1	98.4	7.4	99.8	6.3	98.0	8.0	98.1	7.2	99.8	6.1
Kendall	94.2	8.1	99.0	7.4	99.6	6.3	95.9	8.0	96.3	7.3	98.9	6.1
SBD	94.9	5.4	97.0	5.0	99.5	4.2	91.6	5.4	93.8	5.0	98.9	4.2
DTW	81.1	5.5	99.1	6.2	99.8	4.3	86.7	7.8	92.2	5.2	98.6	4.2

	Genesis						C.A.T.S					
Missing Rate	5%		10%		20%		5%		10%		20%	
	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec
Pearson (ours)	99.5	97.8	87.5	62.4	85.7	51.4	99.5	23.1	99.8	15.7	99.9	11.7
Kendall	78.8	64.4	81.0	65.4	82.0	51.3	87.5	15.5	87.6	14.9	90.4	4
SBD	79.5	56.9	78.2	56.7	85.6	37.6	90.4	22.3	87.1	19.2	90.8	15.0
DTW	80.6	71.0	78.4	69.9	85.4	44.6	94.7	18.2	95.0	17.5	94.5	14.6

Table 5.2: Comparison of RIFT's *Precision* and *Recall* when employing different measures to perform the Step 1 clustering. The best configuration is highlighted in green.

performance. In particular, SPLINE, BFILL, and FFILL showed a significant drop in both *Precision* and *Recall* and an increase of the *RMSE*. This degradation is due to the fact that, unlike the MCAR scenario where missing values are more evenly distributed, in this case, there are prolonged periods of missing data due to the simulated sensor failures. As a result, imputation methods such as BFILL and FFILL, which rely on propagating the nearest observed values, struggled to accurately reconstruct the data. Similarly, SPLINE was severely affected by the lack of regularly distributed reference points, leading to less precise interpolations. As a matter of fact, while the *RMSE* scores of SPLINE were the best among the baselines in the MCAR experiments, in this more challenging scenario these scores became the worst overall. As regards the proposed approach, RIFT showed a similar behavior, providing few but very precise imputations. Even in this scenario, by applying RIFT as a preprocessing step, we were able to improve the baselines' imputations, with an average improvement of 4.36% for *Precision* and *Recall*, respectively, and an average reduction of 6.74% for the *RMSE*.

5.2.2.5 *Limitations*

The RIFT framework is based on two specific strategies available in the literature, i.e., RENUVER for imputation and INDIBITS for RFD discovery, along with their specific configurations. Moreover, although RIFT has been evaluated on a rich set of time series, the results are limited to such a specific set of datasets. Finally, to properly evaluate imputations, the evaluation only considers datasets with artificially injected missing values.

To summarize, we introduced RIFT, an RFD-based preprocessing approach to inject high-quality values in multivariate time series and consequently enhance the quality of imputations of other techniques. RIFT introduces several optimization tailored for multivariate time series, enabling the use of RFDs to impute missing values also in these scenarios. Results showed that each introduced optimization significantly reduces execution times, and that RIFT is able to improve other imputation algorithms both from a quantitative and a qualitative perspective, with average gains in *Precision* and *Recall* up to 29.38%, and *RMSE* up to 9.79%.

6

DATA IMPUTATION IN INCREMENTAL SCENARIOS

One of the recently pursued research directions in the context of Data Imputation is the development of approaches tailored for incremental scenarios. As discussed in Section 4.3, the use of RFD-based imputation approaches in such scenarios has not yet been explored, and raises new challenges. As a matter of fact, profiling metadata discovery and data imputation must be rethought as a continuous process that can respond to changes in real time. This introduces two key challenges: (*i*) imputation becomes less reliable, as the metadata used to guide it might be incomplete or outdated, due to evolving data relationships, and (*ii*) profiling metadata discovery becomes more complex, as the presence of MVs in newly arriving data distorts such relationships. This Chapter will present an empirical study aimed at analyzing the mutual influence between metadata discovery and data imputation in such settings. In particular, Section 6.1 will provide an overview of the problem, while Section 6.2 will provide the details and the outcomes of the study.

6.1 PROBLEM OVERVIEW

This section discusses the problem of missing values in incremental scenarios from two perspectives. We first analyze the problem of imputing missing values leveraging dynamically evolving metadata (Section 6.1.1) and then we analyze the impact of missing values on discovery processes (Section 6.1.2).

6.1.1 *Missing value imputation in incremental scenarios*

Having a substantial number of representative metadata is essential for dependency-based imputation approaches to perform optimally. To fulfill these requirements, a simple approach involves employing automatic discovery algorithms, specifically

designed for static scenarios. This entails running a discovery algorithm on an available relation instance or on a portion of it, and then utilizing the resulting RFD$_c$s to perform MV imputation. However, such a solution is not viable in incremental scenarios, as modifying the underlying tuples would invariably render the previously identified RFD$_c$s no longer representative of the evolved instance. As a result, the usefulness of dependencies is greatly diminished, potentially affecting the ability of imputation processes to rely on them. To this end, it would be useful to update RFD$_c$s over time as the dataset evolves. This would enable imputation processes to rely on representative and up-to-date dependencies, potentially leading to better imputation results.

An imputation process over an incremental scenario must consider the change of the RFD$_c$s, since it does not have full knowledge of the complete set of data, but it gradually becomes available over time.

Definition 21 (Incremental Imputation). *Given a relation schema R, an instance r_τ of R and a set of* RFD$_c$s Σ_τ *holding on r_τ ($r_\tau \models \Sigma_\tau$). Let $t[A]$ be a value imputed at time τ according to $\varphi : X_{\Phi_1} \rightarrow A_{\phi_2}$ with $r_\tau \models \varphi$ and considering $r_{\tau+1}$ and $\Sigma_{\tau+1}$ s.t. $r_{\tau+1} \models \Sigma_{\tau+1}$ a time $\tau+1$, if $\varphi \in \Sigma_\tau$ and $\varphi \notin \Sigma_{\tau+1}$, then the imputation for $t[A]$ at time τ cannot be considered still valid. Thus, a missing value must be restored for $t[A]$, i.e., $t[A] = _$.*

Example 6.1.1. Let us consider the snippet in Table 6.1 and the RFD$_c$ φ: $\mathsf{Address}_{\leq 8.5}, \mathsf{City}_{\leq 8.5} \rightarrow \mathsf{Name}_{\leq 8.5}$ holding on it at time τ. Let us suppose that φ is used to impute the missing value in t_3[Name] with the value of t_2[Name]. At time $\tau+1$, φ gets invalidated due to the insertion of t_{12}, which has similar values to t_{10} and t_{11} for the attributes Address and City but a different value for the attribute Name. Since the imputation for t_3[Name] results in the invalidation of a previously valid RFD$_c$, we can assume that such an imputation might be wrong. Thus, the missing value is restored for t_3[Name].

6.1.2 *Impact of missing value imputation on* RFD$_c$ *discovery*

There is no single interpretation of missing values, even though interpretations can converge toward the following two meanings:

i) a missing value is a *non-existing value*, i.e., a value that does not exist; and ii) a missing value is an *unknown value*, i.e., a value that should exist but is not known. This dual interpretation leads to recognizing the presence of MVs as an important problem to tackle in different application contexts [19]. The presence of MVs has already been studied for the discovery of canonical FDs [23], but it is also needed to evaluate the problem in the context of RFD$_c$s. A commonly selected choice for handling the presence of MVs in RFD$_c$ discovery is to *assign maximum distance* when an observed value is evaluated with respect to a missing one. However, this choice yields invalidating possibly meaningful RFD$_c$s if the attribute containing the missing value appears on the RHS of the dependency. Formally, given a missing value $t[A] = _$, then each $\varphi : X_{\Phi_1} \rightarrow A_{\phi_2}$ can be invalidated by the presence of the missing value on $t[A]$, since the distance between t and any other tuple t' would be always set to the maximum one, leading to the violation of ϕ_2.

Example 6.1.2. Consider again the dataset snippet in Table 6.1 related to time τ and the RFD$_c$ φ: $\mathsf{Phone\ Number}_{\leq 1.0} \rightarrow \mathsf{City}_{\leq 8.5}$. We can notice that φ is not valid due to the presence of a single missing value for the attribute Phone Number in t_0. As a matter of fact, t_0 and t_1 are similar when considering the LHS of φ, but they violate the constraint on its RHS because the comparison with a missing value leads to the assignment of the maximum distance, which exceeds the defined similarity threshold.

6.2 DATA DEPENDENCY MEETS DATA IMPUTATION [26]

To investigate the challenges described in the previous section, we conducted an empirical study driven by two research questions. In particular, the first research question aims to study how the continuous update of RFD$_c$s can enhance the effectiveness of imputation processes:

> **RQ1:** To what extent does the continuous update of RFD$_c$s improve the quality of imputations in incremental scenarios?

	t	Name	Address	City	Phone Number	Type
r	t_0	arnie morton's of chicago	435 s. la ciniega blv.	-	310/246 – 1501	american
	t_1	arnie morton's of chicago	435 s. la ciniega blvd.	los angeles	310/246 – 1501	steakhouses
	t_2	art's delicatessen	12224 ventura blvd.	sherman oaks	818/762 – 1221	american
	t_3	-	12224 ventura blvd.	studio city	-	-
	t_4	hotel bel-air	701 stone canyon rd.	bel air	-	californian
	t_5	trio cafe	1870 filmore st.	san francisco	415 – 563 – 2248	american
	t_6	bel-air hotel	701 stone canyon rd.	bel air	-	california
	t_7	cafe bizou	14016 ventura blvd.	sherman oaks	818/788 – 3536	french
	t_8	cafe bizou	14016 ventura blvd.	sherman oaks	818 – 788 – 3536	french bistro
	t_9	wa-ha-ka oaxaca mexican grill	2141 polk st.	san francisco	415 – 775 – 1055	mexican
	t_{10}	3200 las vegas blvd. s.	3200 las vegas blvd. s.	las vegas	702/733 – 8899	asian
	t_{11}	3200 las vegas blvd. s.	3200 las vegas blvd. s.	las vegas	702 – 733 – 8899	chinese
$r_{\tau+1}$	t_{12}	palace court	3570 las vegas blvd. s.	las vegas	702/731 – 7547	continental
	t_{13}	four seasons grill room	99 e. 52nd st.	new york	212/754 – 9494	american
	t_{14}	palace court	3570 las vegas blvd. s.	-	702 – 731 – 7547	french bistro
	t_{15}	four seasons	99 e. 52nd st.	new york city	212 – 754 – 9494	american (new)

Table 6.1: A snippet of the *Restaurant* dataset with a new batch of tuples added at time $\tau + 1$.

On the other hand, the second research question aims to study how data imputation can enhance the results of RFD$_c$ discovery processes:

> **RQ2:** To what extent can imputation methods mitigate the effect of MVs in the discovery of RFD$_c$s in incremental scenarios?

To answer these questions, we devised two specific pipelines to enable the cooperation between incremental RFD$_c$ discovery and RFD$_c$-based imputation, whose details will be provided in the following sections, along with the experimental settings and the outcomes of our study.

6.2.1 *Experimental Settings*

In this section, we detail the datasets employed and their preparation, the pipelines used to enable the cooperation between incremental RFD$_c$ discovery and RFD$_c$-based imputation, the metrics used to assess the experimental results, as well as the analysis method.

Dataset	ID	Rows	Cols	Values
Cars	1	406	9	3.654
Restaurant	2	864	6	5.184
Boeing	3	898	10	8.980
Cats	4	1.071	10	10.710
Police	5	2.204	9	19.836
Telemetry	6	3.000	8	24.000
Actors	7	4.000	7	28.000
Medical Charges	8	2.500	12	30.000
F1	9	5.000	7	35.000
MotoGP	10	3.000	12	36.000
US Presidents	11	3.754	10	37.540
NBA	12	3.200	12	38.400
EV Vehicles	13	4.000	10	40.000
Superstore	14	4.500	11	49.500
Air	15	9.000	13	117.000

Table 6.2: Details of the datasets employed in the evaluation.

ID	1%	2%	3%	4%	5%	10%	20%	30%	40%	50%
1	37	73	110	146	183	365	731	1.096	1.462	1.827
2	52	104	156	207	259	518	1.037	1.555	2.074	2.592
3	90	180	269	359	449	898	1.796	2.694	3.592	4.490
4	107	214	321	428	536	1.071	2.142	3.213	4.284	5.355
5	198	397	595	793	992	1.984	3.967	5.951	7.934	9.918
6	240	480	720	960	1.200	2.400	4.800	7.200	9.600	12.000
7	280	560	840	1.120	1.400	2.800	5.600	8.400	11.200	14.000
8	300	600	900	1.200	1.500	3.000	6.000	9.000	12.000	15.000
9	350	700	1.050	1.400	1.750	3.500	7.000	10.500	14.000	17.500
10	360	720	1.080	1.440	1.800	3.600	7.200	10.800	14.400	18.000
11	375	751	1.126	1.502	1.877	3.754	7.508	11.262	15.016	18.770
12	384	768	1.152	1.536	1.920	3.840	7.680	11.520	15.360	19.200
13	400	800	1.200	1.600	2.000	4.000	8.000	12.000	16.000	20.000
14	495	990	1.485	1.980	2.475	4.960	9.900	14.850	19.800	24.750
15	1.170	2.340	3.510	4.680	5.850	11.700	23.400	35.100	46.800	58.500

Table 6.3: Number of injected missing values for each dataset.

6.2.1.1 *Datasets and Preprocessing steps*

For our evaluation, we considered 15 real-world datasets[1](see Table 6.2). Moreover, to guarantee that the analysis is performed under a controlled environment, i.e., properly track the validity of imputed values w.r.t. ground truth values, we performed several preprocessing operations. We first removed all tuples containing MVs already present in the data, and/or discarded attributes containing too many MVs, aiming at not losing too many tuples. Notice that the datasets' tuples have not been shuffled, so they follow the original ordering. After that, we artificially injected into the datasets a certain percentage of MVs by following the concept of Missing Completely at Random (MCAR). This procedure is required, since without knowledge of the original values, it would not be possible to objectively measure the correctness of imputations. We considered 10 different missing rates, i.e., 1%, 2%, 3%, 4%, 5%, 10%, 20%, 30%, 40%, 50%. The total number of injected MVs for each configuration setting is provided in Table 6.3. It is worth noting that the random injection of MVs can introduce a degree of variability in the results. To address this, we generated five different versions of the datasets for each of the 10 missing rates, resulting in 50 versions for each of the 15 datasets, for a total of 750 configurations. Thus, for each dataset and a specific missing rate, all the results will refer to the average of the outcomes corresponding to the 5 configurations. In addition, since in the real-world MVs are often not missing completely at random, we also considered two other missing mechanisms to perform supplementary experiments. The first one consists of injecting consecutive MVs within the same attribute across a random number of tuples (we will refer to this mechanism simply as MNAR - Missing Not at Random). On the other hand, the second mechanism employed is Missing Based on Unobserved Values (MBUV) [147], which entails generating a synthetic numeric feature that follows a normal distribution for each attribute. Observations corresponding to the extreme tails of this latent feature (e.g., the lowest and highest quantiles)

1 Supplementary materials related to this work are available on the official repository

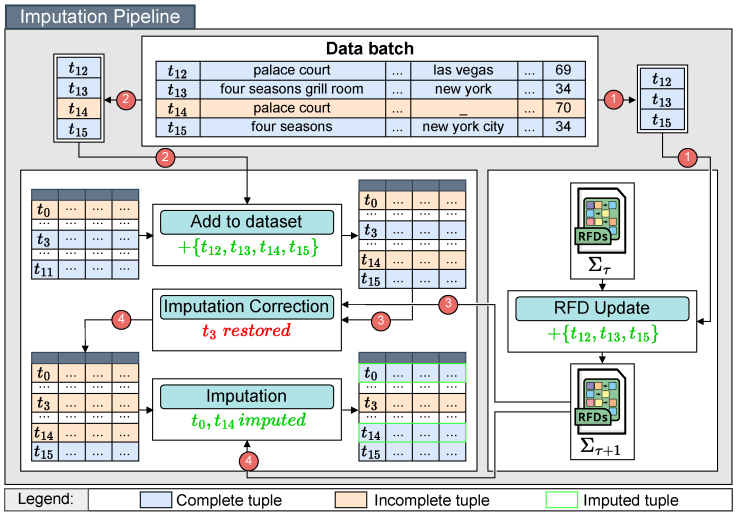


Figure 6.1: Overview of the *Imputation Pipeline* at a generic step, the batch size k is set to 4.

are then used to determine the MVs in the associated observed attribute.

6.2.1.2 *Proposed Pipelines*

To enable a proper evaluation of the relationship between imputation and RFD$_c$ discovery processes, we selected specific algorithms in the literature to independently perform such tasks. In particular, we considered the RENUVER imputation algorithm that uses RFD$_c$s to impute MVs in static relation instances [30]; and the INDIBITS RFD discovery algorithm [29], which enables the incremental update of sets of holding RFD$_c$s at different time instants, according to batches of operations performed over tuples, i.e., the insertion of new tuples, the update of existing tuples, and the deletion of old tuples. Moreover, incrementally updating dependencies is more efficient than processing all datasets at once, as shown in [29, 73].

Both INDIBITS and RENUVER evaluate tuples based on their similarity in attribute values: INDIBITS verifies the satisfiability of thresholds set for constraints on attributes, and RENUVER evalu-

ates the most similar candidate for the imputation. To achieve this goal, the absolute difference is employed to compute similarities in the case of numerical values, while the edit distance is utilized when string values are involved. Moreover, since INDIBITS is defined as a fixed-threshold incremental RFD_c discovery algorithm, meaning that the threshold values for each attribute must be pre-determined and set as input parameters, we considered specific attribute comparison thresholds based on the distribution of distances provided by pairs of values in each attribute.

For the evaluation of RQ1 and RQ2, we defined two distinct pipelines providing collaborative interaction between the discovery and imputation processes. The first pipeline, referred to as the *Imputation Pipeline*, is devised to assess the extent to which an incremental RFD_c discovery algorithm can enhance the data imputation process. Conversely, the second pipeline, known as the *Discovery Pipeline*, aims to evaluate the extent to which a data imputation algorithm can support RFD_c discovery processes.

The *Imputation Pipeline* harnesses dynamically inferred RFD_cs from the data to provide the imputation algorithm with updated metadata (see Figure 6.1). It is important to note that, to simulate an incremental scenario, our setup does not read the entire dataset in one operation. Instead, the dataset is processed in batches, with each batch consisting of a fixed size, denoted as k. As shown in Figure 6.1, upon receiving a new batch, the discovery algorithm updates the set of dependencies that were established in the previous step, taking into account the new data. Subsequently, before proceeding with the imputation phase, a check is performed on the imputations made in previous steps. This check is necessary because some imputations might have been based on RFD_cs that are no longer valid. Thus, the imputations made using outdated RFD_cs are discarded, and the original MVs are restored. Once the updated dataset is obtained, it serves as input for the imputation step, along with the updated RFD_cs. These three steps – updating dependencies, checking previous imputations, and performing imputations – are repeated until no further batch of data is available. To accurately evaluate the influence of RFD_cs on the proposed imputation process, the pipeline focuses solely on complete tuples when inferring RFD_cs during the discovery step.

This approach ensures that the RFD$_c$s are extracted without the "noise" introduced by the presence of MVs.

Let us consider the example in Figure 6.1, which refers to time $\tau + 1$ based on the snippet in Table 6.1. For this example, we assume that the batch size k is set to 4.

Example 6.2.1. Let us suppose that a batch of 4 new tuple insertions, namely t_{12}, t_{13}, t_{14} (which contains a MV for the attribute City), and t_{15}, is available. As the discovery step has to consider only complete tuples, only t_{12}, t_{13} and t_{15} are sent to the discovery algorithm (1), which updates the set of previously holding RFD$_c$s Σ_τ, obtaining the new set $\Sigma_{\tau+1}$. Conversely, all 4 tuples received are added to the current dataset (2). As per Definition 18, the RFD$_c$ φ: $\mathsf{Address}_{\leq 8.5}, \mathsf{City}_{\leq 8.5} \rightarrow \mathsf{Name}_{\leq 8.5}$ gets invalidated during the discovery process due to the insertion of t_{12}. Thus, φ does not belong to $\Sigma_{\tau+1}$ anymore. Supposing that φ was used to impute t_3[Name] at time τ, during the imputation correction step the original missing value is restored (3), i.e, t_3[Name] = _. The updated dataset and $\Sigma_{\tau+1}$ are then sent to the imputation algorithm (4). Instead, the RFD$_c$ φ: $\mathsf{Name}_{\leq 2} \rightarrow \mathsf{City}_{\leq 8.5}$ is still holding, and it can be used for the imputation of t_{14}[City]. In fact, t_{12} has a similar value for the attribute Name, and thus t_{12}[City] is a plausible value for t_{14}[City]. Furthermore, by means of $\Sigma_{\tau+1}$, also t_0 is imputed.

The *Discovery Pipeline* harnesses the data imputation process to provide the discovery algorithm with additional imputed tuples. As the *Imputation Pipeline*, the *Discovery Pipeline* processes data in batches of size k. As shown in Figure 6.2, at the beginning of each iteration, the *Discovery Pipeline* performs an imputation correction step, to discard imputations based on RFD$_c$s that have been invalidated in the previous iteration. Subsequently, the imputation step is performed by leveraging the previously discovered RFD$_c$s. Upon receiving a new batch, the discovery algorithm updates the RFD$_c$s according to the complete tuples included in it. However, with respect to the *Imputation Pipeline*, the *Discovery Pipeline* also considers the tuples completed by the imputation step (which are included as insertions in the discovery process) and the ones restored by the imputation correction step (which are included as deletions in the discovery process). These three steps - checking

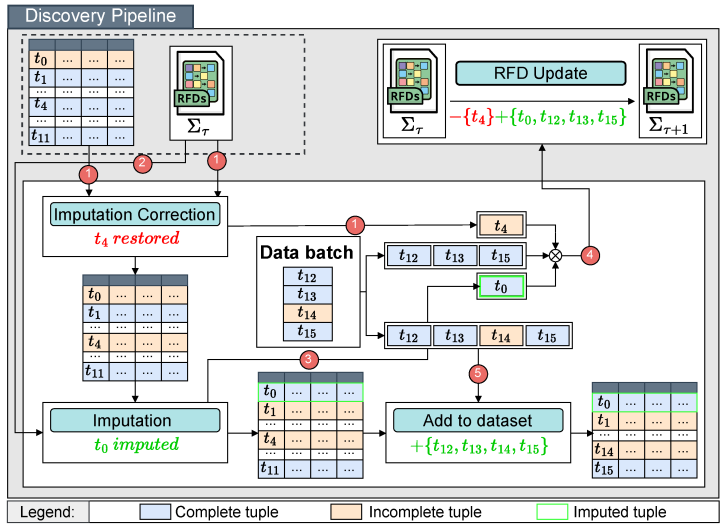


Figure 6.2: Overview of the *Discovery Pipeline* at a generic step, the batch size k is set to 4.

previous imputations, performing imputations, and updating dependencies - are repeated until no further batch of data is available.

Let us consider the example in Figure 6.2, which refers to time $\tau+1$ based on the snippet in Table 6.1. As before, we assume that the batch size k is set to 4.

Example 6.2.2. At time $\tau+1$, the pipeline performs the imputation correction step on the dataset consisting of the tuples $t_0, \ldots, t_{11}$ and the set Σ_τ of RFD$_c$s. Let us suppose that tuple t_4 has been previously completed by imputing a missing value on the basis of the RFD$_c$ φ: Type$_{\leq 1.0} \rightarrow$ Phone$_{\leq 1.0}$. Suppose that, after the discovery at time τ, this RFD$_c$ has been invalidated and, consequently, at time $\tau+1$, the imputation correction step restores the original MV, thereby making t_4 incomplete (1). Then, the imputation step is performed (2). By considering the example 4.1.1, let us assume that tuple t_0 is imputed over the attribute City, i.e., t_0[City]="*Los Angeles*" by means o f the RFD$_c$ φ: Name$_{\leq 2} \rightarrow$ City$_{\leq 8.5}$ (3). Then, a new data batch is available, and the discovery step updates the RFD$_c$ set Σ_τ by considering the

complete tuples included in the latter. In addition, the discovery process also considers the completed tuple t_0 as a new insertion, and the restored tuple t_4 as a deletion (4). Finally, the received data batch with both complete and incomplete tuples is added to the dataset (5) that will be used in the next iteration.

Notice that at the beginning of the *Discovery Pipeline*, there is neither a dataset nor a set of RFD$_c$s available. As a result, the imputation correction step and the execution of the imputation algorithm are not performed initially. Instead, an initial discovery step is conducted solely on the complete tuples obtained from the initial batch. This enables the extraction of an initial set of RFD$_c$s, which can be used for the subsequent imputation steps.

6.2.1.3 *Metrics*

To evaluate **RQ1**, we measured the quality of the performed imputations. To this end, we consider the *Root Mean Square Error* (*RMSE*), which quantifies the magnitude of errors between the original and imputed values.

However, the traditional *RMSE* is not suitable for textual values. Therefore, we devised an adapted version of RMSE that satisfies the following requirements:

- It returns normalized values in the range [0,1];
- It supports both numerical and textual attributes;
- For textual attributes, it takes into account their similarity.

For each numerical attribute, we apply min-max normalization using the minimum and maximum values computed from the oracle dataset. The same is then applied to the values imputed for that specific attribute, so as to bound the numerical errors computed during the evaluation of the imputation between 0 and 1. Formally, given a relation schema R defined over a set of attributes $attr(R)$, an instance r of R, for each attribute $A \in attr(R)$, s.t., $dom(A) \in \mathbb{R}$, let a_{max} and a_{min} be the maximum and minimum values in r for attribute A, respectively, each value $\hat{y}$ employed to impute a missing value over attribute A is normalized as follows:

$$\hat{y}_{\text{norm}} = \frac{\hat{y} - a_{min}}{a_{max} - a_{min}}$$

Finally, we compute the absolute difference between the normalized imputation and the normalized ground truth value y, i.e., $|y_{\text{norm}} - \hat{y}_{\text{norm}}|$. For clarity, no normalization was performed on the datasets prior to the execution of the pipelines. Rather, normalization was performed exclusively to evaluate the imputation results.

As for textual attributes, we define a binary distance function: if the imputed value is *equal* to the ground truth value, the error is 0; if they are different, the error is 1. However, to accurately assess the quality of the imputations made, it is necessary to evaluate the imputation of textual attributes by also considering similarities (e.g., the values *New York City* and *NYC* are different but represent the same entity). To this end, for each textual attribute, all possible values are grouped into *equivalence classes* by exploiting the same rule-based framework used in [30]. Thus, if the imputed value and the ground truth value belong to the same class, they are considered equivalent and the error is set to 0; otherwise, 1.

Drawing from the above definitions, we formalize the adapted RMSE metric used for evaluating imputation results. Given a relation schema R defined over a set of attributes $attr(R)$, an instance r of R, a tuple $t \in r$ s.t. $t[A] = _$, let $\hat{y}$ be the value employed to impute $t[A]$, i.e., $t[A] = \hat{y}$, y be the ground truth value for $t[A]$, and imp be the number of imputations, the adapted RMSE metric is:

$$\textbf{\textit{RMSE}} = \sqrt{\frac{1}{imp} \sum_{t[A]=\hat{y}} d(y, \hat{y})^2} \quad \forall t \in r$$

where $d(y, \hat{y})$ is a distance function defined as:

$$d(y, \hat{y}) = \begin{cases} |y_{\text{norm}} - \hat{y}_{\text{norm}}| & \text{if } dom(A) \in \mathbb{R} \\ 0 & \text{if } y, \hat{y} \text{ belong to the same equivalence class} \\ 1 & \text{otherwise} \end{cases}$$

To evaluate **RQ2**, we consider evaluation metrics to assess the quality of the discovered RFD$_c$s. Similarly to the imputation evaluation, where the imputed value is compared to its ground truth, we can assess the discovery results by comparing the set of discovered RFD$_c$s with a set of RFD$_c$s extracted from the oracle

dataset. Consequently, we name this set of RFD_cs, the oracle RFD_cs. More formally, given an RFD_c set Σ and an oracle RFD_cs Σ^*, we evaluate the performance of the pipeline by considering:

- The percentage of RFD_cs in Σ^* that exactly match the ones in Σ (RFD_cs *found*);
- The percentage of RFD_cs in Σ^* that are generalized with respect to the ones in Σ (*Generalized* RFD_cs);
- The percentage of RFD_cs in Σ^* that are specialized with respect to the ones in Σ (*Specialized* RFD_cs);
- The percentage of RFD_cs in Σ^* that does not belong in any form to Σ. This means that an RFD_c in Σ^* is not exactly matched, is not generalized, and is not specialized by the ones in Σ (RFD_cs *not found*);
- The percentage of RFD_cs in Σ not included in Σ^*. This means that an RFD_c in Σ does not exactly match, does not generalize, and does not specialize any RFD_c in Σ^* (*New* RFD_cs).

Notice that, the percentage of RFD_cs *not found*, RFD_cs *found*, *Generalized* RFD_cs, and *Specialized* RFD_cs are computed with respect to the total number of RFD_cs in Σ^*. Conversely, the percentage of *New* RFD_cs is computed with respect to the total number of RFD_cs discovered by the *Discovery Pipeline*. To quantify the degree of divergence between the RFD_cs within Σ^* and the ones discovered by the *Discovery Pipeline*, we also employed a recently proposed approach [44], which enabled us to aggregate all of the aforementioned metrics within a single value in the range [0,1] (this work is also proposed in this thesis, see Chapter 7). More formally, let *Spec* be the number of *Specialized* RFD_cs, *Gen* be the number of *Generalized* RFD_cs, *NF* be the number of RFD_cs *not found*, and *New* be the number of *New* RFD_cs, the divergence metric used in our evaluation is defined as follows:

$$div = \frac{New + (Gen \cdot 0.2) + (Spec \cdot 0.2) + NF}{|\ \Sigma \cup\ \Sigma^*|}$$

Notice that, this metric leverages coefficients to weight different types of RFD_c changes, assigning greater importance to more

substantial changes (i.e., RFD$_c$s *not found* and *new* RFD$_c$s) while attributing a lower contribution to moderate ones (e.g., *Specialized* RFD$_c$s and *Generalized* RFD$_c$s), aiming at accurately estimating the severity of changes among the two RFD$_c$ sets.

6.2.2 *Analysis Method*

In what follows, we introduce the two pipelines devised for addressing the two RQs considered in this study.

6.2.2.1 *Imputation pipeline*

To address **RQ1**, we compared the imputation results in terms of RMSE of the *Imputation Pipeline* against a baseline where RFD$_c$s are discovered only from the first 20% of complete tuples of the dataset. The same set of RFD$_c$s is employed for the imputation of MVs, also as the underlying dataset evolves. This scenario reflects a situation in which a limited number of analyzed tuples yield a generalized, non-representative set of RFD$_c$s. We also evaluated two other variants of the *Imputation Pipeline* as baselines: *PipelineNoRev*, which never discards previous imputations that are based on an invalidated RFD$_c$s, and *Hybrid*, which performs such a correction only if there is another new suitable candidate for imputing an MV previously imputed. Conversely, the *Imputation Pipeline*, always discards previous imputations that are based on an invalidated RFD$_c$. All these different pipelines allow us to compare different strategies for handling imputations in incremental scenarios. Moreover, to test the generalizability of the analysis, we included scenarios with different missing mechanisms (i.e., MNAR and MBUV), as well as a supplementary evaluation considering a different imputation approach, namely DERAND [170]. Since DERAND leverages the entire RFD$_c$ set rather than linking specific RFD$_c$s to each imputation, only the baseline and *PipelineNoRev* could be compared. This is because, once an RFD$_c$ is invalidated, it is impossible to trace which imputations used it, blocking any imputation correction step. Furthermore, as DERAND slows significantly on larger datasets and high MV rates, we limited tests to the three smallest datasets with missing

rates of up to 5%. This was still sufficient to assess the benefit of updated metadata in imputation.

6.2.2.2 *Discovery pipeline*

To address **RQ2**, we performed an evaluation comparing the *Discovery Pipeline* with a baseline approach, describing a scenario where the discovery process is performed on a dataset containing MVs. This baseline represents a naïve yet realistic scenario, where MVs are not explicitly handled before discovery, allowing us to assess their impact on the RFD_c quality. By comparing both strategies against the oracle RFD_cs obtained from the oracle dataset, we can measure the benefits of incorporating imputation into the discovery process. Our goal is to determine whether the synergy between RFD_c discovery and data imputation could lead to a higher number of discovered RFD_c that better approximate oracle RFD_cs than the baseline approach. To this end, we quantify possible improvements by using the metrics defined in Section 6.2.1.3. Therefore, the initial step in the analysis involved the classification of the discovered RFD_cs based on the nature of their alterations in relation to the oracle RFD_cs. Subsequently, the divergence measure was calculated in accordance with the formula provided in Section 6.2.1.3. In addition, as done with the *Imputation Pipeline*, we conducted a supplementary analysis to evaluate the compared strategies in scenarios with different missing mechanisms (i.e., MNAR and MBUV). Finally, to evaluate the generalizability of the results with a different type of profiling metadata, we conducted an evaluation using canonical FDs.

We also considered different values of the batch size k (i.e., 1%, 2%, 3%, 4%, and 5% of tuples in each dataset), and we observed that it does not impact the outcomes observed for both RQs. In what follows, we discuss the results observed with k=2%, while the ones related to other batch sizes are available online.

6.2.3 *Improving the imputation quality in incremental scenarios (RQ1)*

Figure 6.3 shows the imputation results in terms of *RMSE* for the *Imputation Pipeline*, the *PipelineNoRev*, the *Hybrid*, and the baseline

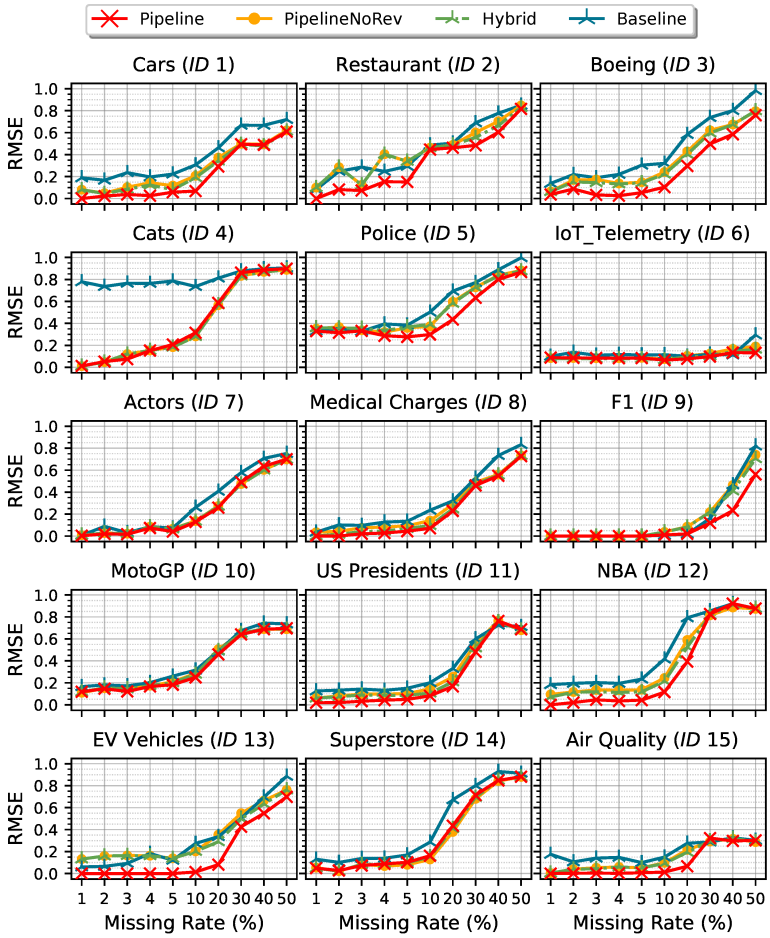


Figure 6.3: Comparison between the imputation results of the *Imputation Pipeline* and those of the baselines.

strategies across all experimental configurations. In general, it is possible to notice that all the compared strategies show an increasing *RMSE* trend as the missing rate grows. This can be attributed to two main reasons: first, the imputation process becomes more challenging, since the presence of a larger number of MVs hinders the possibility of finding good candidates for performing imputations; secondly, since the discovery is only performed on complete tuples, the RFD_cs used in the imputation process tends to become more generalized as the missing rate increases, further affecting the candidate selection process. This

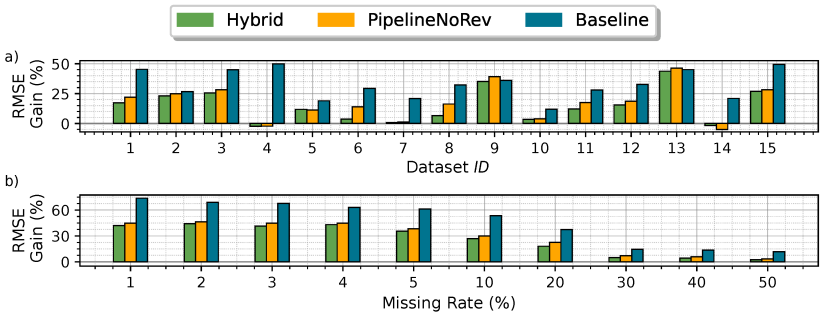


Figure 6.4: RMSE gain of the imputation pipeline w.r.t. the baselines for each dataset (a) and for each missing rate (b).

trend is particularly evident in almost all configurations when the missing rate exceeds 20% of the dataset values.

By comparing the different strategies, we can see that the baseline (blue line) achieved the worst results in almost all configurations. This is due to the fact that, although the dataset continues to evolve over time, the non-updated set of RFD$_c$s fails to capture the proper relationships between attributes since only the initial set of tuples was used for deriving these dependencies. As a result, the imputation process becomes more challenging, as generalized RFD$_c$s hinder the identification of suitable candidates for imputation and assessing their suitability. On the other hand, the *Imputation Pipeline* (red line) achieved the best results in almost all configurations, with a significantly lower *RMSE* with respect to the baseline. This improvement can be attributed to the fact that the *Imputation Pipeline* leverages more specialized RFD$_c$s discovered on the updated dataset. These RFD$_c$s enable a more precise identification of suitable candidates for imputation, leading to more accurate results. This difference is particularly evident in datasets showing a more significant RFD$_c$ evolution across different data batches (i.e., datasets with *IDs* 1, 2, 3, 4, 5, 12, and 13). While the *Imputation Pipeline* was able to update RFD$_c$s and restore the imputations made with outdated RFD$_c$s, the baseline kept performing imputations using the latter, leading to a significant degradation of the imputation quality.

As for the other baselines, the *PipelineNoRev* strategy (yellow line) achieved *RMSE* scores that fall between the *Imputation Pipeline* and the baseline without RFD$_c$ update. This is due to the

fact that, in contrast to the baseline, the *PipelineNoRev* considers RFD$_c$s that are updated over time, enabling correct imputations when performed on new batches of tuples. However, unlike the *Imputation Pipeline*, the *PipelineNoRev* does not re-evaluate the imputations performed at the beginning with RFD$_c$s that got invalidated in the subsequent steps. These lower-quality imputations affect the *RMSE* of the *PipelineNoRev* w.r.t. the *ImputationPipeline*. Regarding the *Hybrid* strategy (green line), we can see that it has a similar behavior with respect to the *PipelineNoRev* strategy, with a slightly lower *RMSE*. This improvement is due to the corrections made to the imputations performed with outdated RFD$_c$s if another suitable candidate is identified. However, the *RMSE* is still higher than the *Imputation Pipeline* in most of the experiments. This is because most of the imputations made by the *Hybrid* strategy were made on the basis of highly generalized RFD$_c$s that did not reflect the actual properties of the dataset. Thus, as RFD$_c$s got updated and specialized, it became more challenging to find a suitable candidate. As a result, the lower quality imputations performed were not corrected during the imputation process.

Moreover, in many of the analyzed datasets, the difference between the *Imputation Pipeline* and the other strategies tends to reduce with higher missing rates. This is probably due to the fact that, since all strategies discover RFD$_c$s on only complete tuples, the number of additional tuples that the *Imputation Pipeline* considers during the RFD$_c$ update is limited, thus resulting in less differences with the RFD$_c$s employed by the other strategies. However, in a real-world scenario in which data is generated continuously, the difference in the number of tuples considered by the strategies could be greater than the one observed in the experiments.

To summarize, Figure 6.4(a) shows the average *RMSE* gain achieved by the *Imputation Pipeline* with respect to the compared strategies. The highest improvements were observed with respect to the baseline, with gains ranging from 11.96% to 49.37% (with an overall gain across all experiments of 33.15%). As for the *Hybrid* and *PipelineNoRev* strategies, the gains achieved by the *Imputation Pipeline* range from 0.7% to 43.71% and from 1.2% to 46.21%, respectively (with overall gains across all experiments

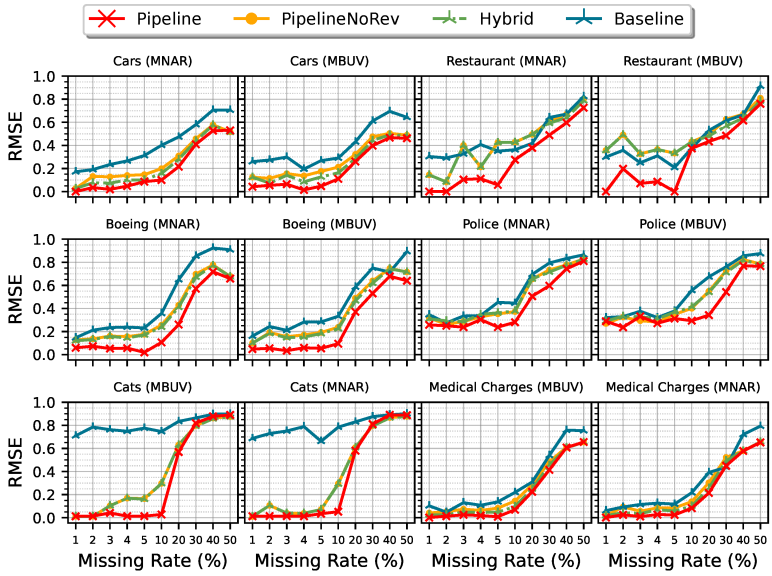


Figure 6.5: Imputation results with other missing mechanisms.

of 14.06% and 16.4%, respectively), with the only exception of *Cats* and *Superstore* (*IDs* 4 and 14) in which these two strategies achieve a slightly better *RMSE*. This may be due to the presence of a candidate that misled the re-imputation of an MV by the *Imputation Pipeline*, while the other variants did not consider it since they kept the previous imputation. Although the difference is small, these outliers highlight the need to devise more refined re-evaluation strategies.

Finally, Figure 6.4(b) shows the average *RMSE* gain per missing rate. As discussed, the gains tend to decrease with higher missing rates: for the baseline, the gains start from 73.69% and gradually drop to 11.8%, while for the *Hybrid* strategy they range from 42.12% to 2.54%. Finally, for the *PipelineNoRev* the gains start from 44.76% and gradually degrade to 3.33%.

Supplementary experiments

Figure 6.5 shows the imputation results observed on a subset of datasets with MNAR and MBUV values. In general, we can notice a similar behavior with respect to the experiments with MCAR MVs. As a matter of fact, the *Imputation Pipeline* achieves

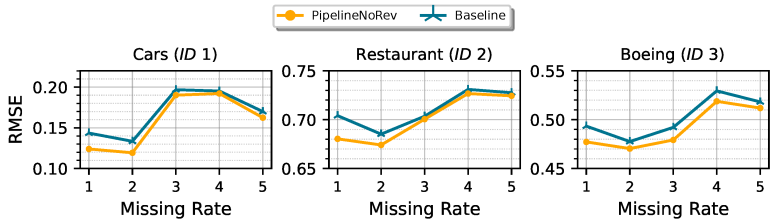


Figure 6.6: Imputation results when using DERAND.

lower *RMSE* scores than the other strategies, while the baseline recorded the worst *RMSE* scores in most of the experiments. The *PipelineNoRev* and *Hybrid* strategy achieved *RMSE* scores that often fall in between the other two strategies. These results prove that all the observations gathered in the previous experiments are generalizable to more realistic scenarios regarding how missing values occur within the data. Finally, Figure 6.6 shows the imputation results when using the DERAND imputation algorithm instead of RENUVER. As we can see, despite the difference being less pronounced than in the previous experiments, the *PipelineNoRev* strategy was able to achieve better RMSE scores than the baseline, also in this scenario. This proves that our findings related to the necessity of updating metadata over time are also generalizable to other metadata-driven approaches, even if they use metadata only to support the imputation process instead of leveraging them for selecting candidates.

6.2.4 *Mitigating the effect of missing values in the discovery processes (RQ2)*

Figures 6.7 and 6.8 show the discovery results of the *Discovery Pipeline* and the baseline, respectively, categorizing the type of RFD_cs discovered with respect to the oracle RFD_cs according to the definitions provided in Section 6.2.1.3. In general, we can see that the *Discovery Pipeline* achieves higher percentages of RFD_c *found*, being able to extract a fraction of the oracle RFD_cs in almost all configurations. On the other hand, the baseline was often unable to discover any oracle RFD_c. This occurred because, as discussed in Section 6.1.2, the presence of MVs significantly affects the discovery results. As a matter of fact, by considering the RFD_cs *not found*, we can notice that the baseline was unable to find a

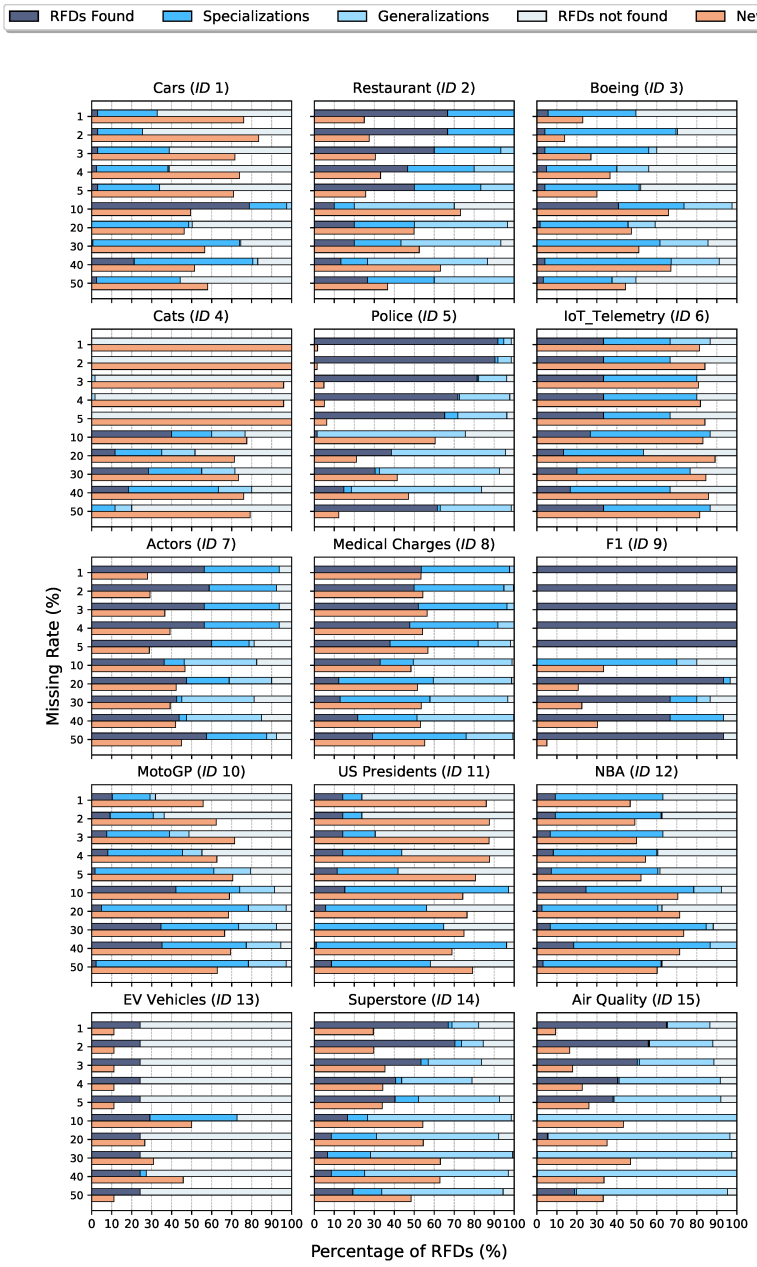


Figure 6.7: Comparison of the RFD_cs extracted by the *Discovery Pipeline* with those of the oracle.

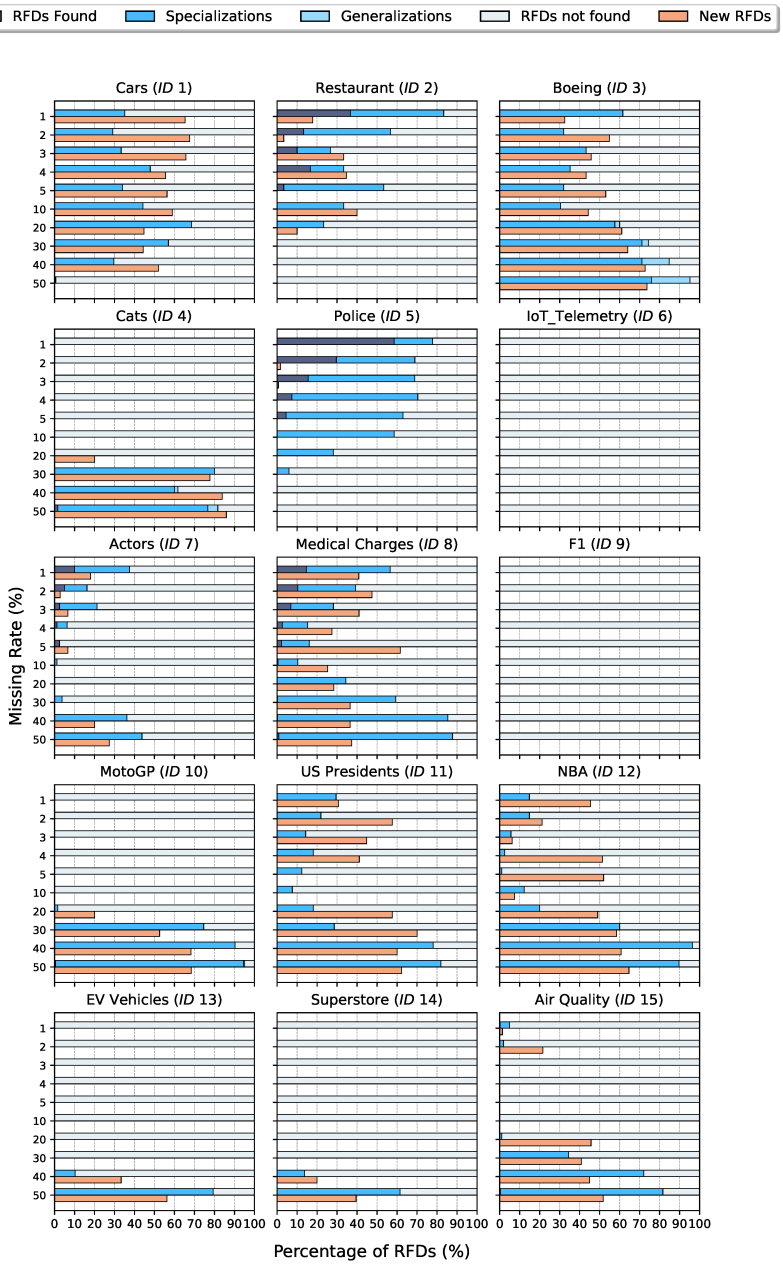


Figure 6.8: Comparison of the RFD_cs extracted by the baseline with those of the oracle.

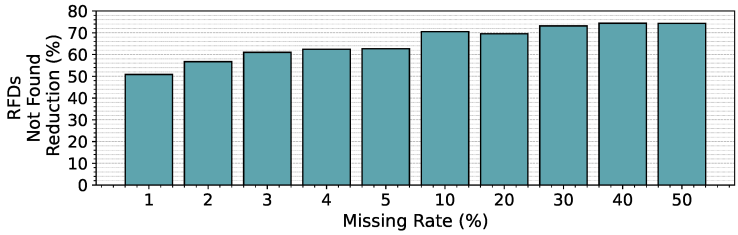


Figure 6.9: RFDs Not Found reduction (%).

significant portion of the oracle RFD$_c$s. Furthermore, on some datasets (i.e., datasets with *IDs* 6 and 9), the baseline is not even able to extract any type of RFD$_c$, thus returning an empty RFD$_c$ set. In this sense, we can see how the *Discovery Pipeline* obtains lower percentages of RFD$_c$s *not found* in most cases, with a drastic, and in some cases, total, reduction of this category of RFD$_c$s. The difference between the two strategies is particularly evident for the *F1* dataset (*ID* 9). The latter is characterized by low-level dependencies, i.e., RFD$_c$s with few attributes on the LHS. In this case, given an RFD$_c$, it is easier to find tuples that satisfy its LHS. However, with the baseline strategy, the discovery algorithm often found an MV when checking the RHS of the RFD$_c$, leading to an invalidation. On the other hand, the *Discovery Pipeline* enabled the recovery of several incomplete tuples, allowing the discovery of RFD$_c$s that often matched the oracle RFD$_c$s.

We now consider the percentage of *Specialized* and *Generalized* RFD$_c$s discovered by the two strategies. Although *Specialized* and *Generalized* RFD$_c$s does not match the oracle RFD$_c$s, they represent dependencies that are in some way related to them. Thus, their presence means that the oracle RFD$_c$s are not lost, but they continue to be valid in their specialized or generalized form. By comparing the *Discovery Pipeline* and the baseline, we notice that both strategies lead to the discovery of a certain amount of *Specialized* and *Generalized* RFD$_c$s. As for the baseline, we can notice that it shows a more heterogeneous behavior: for instance, on some datasets there is a higher percentage of *Specialized* and *Generalized* RFD$_c$s with low missing rates (i.e., datasets with *IDs* 2 and 5), while on others we can observe it on higher missing rates (i.e., datasets with *IDs* 4, 10, 13 and 14). On the other hand, the *Discovery Pipeline* showed a more stable behavior, maintaining

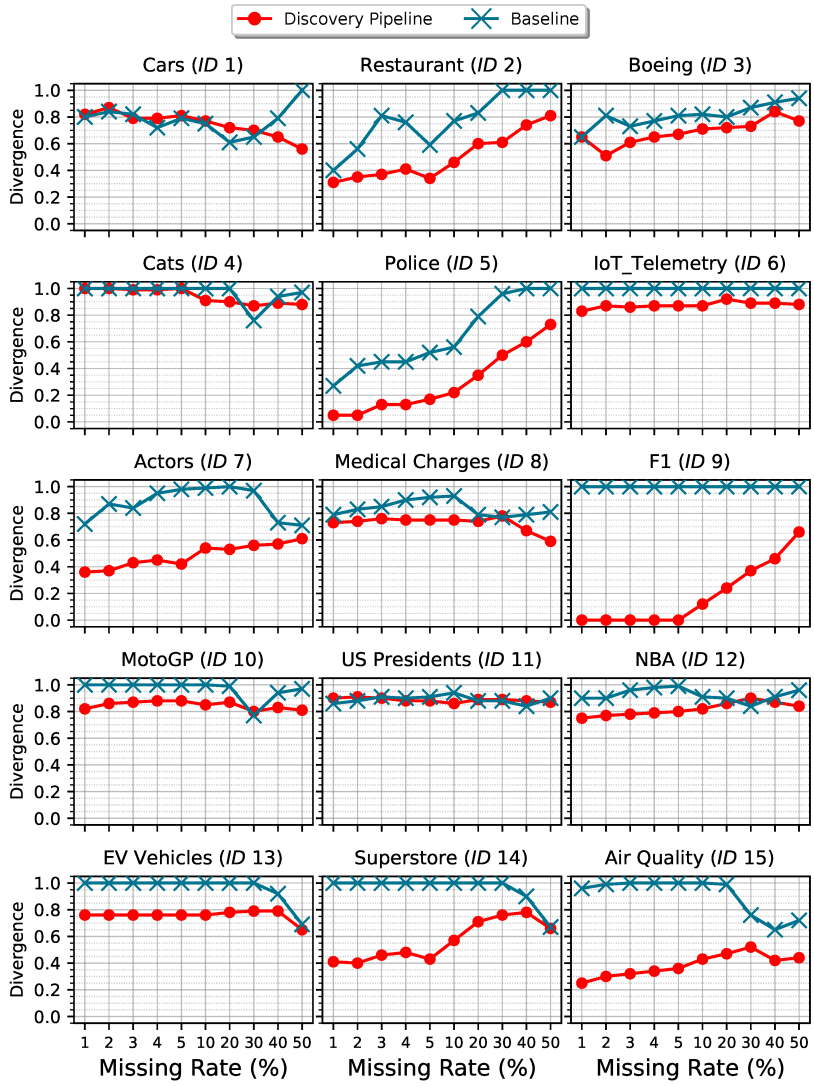


Figure 6.10: Comparing the baseline and the *Discovery Pipeline*.

a similar amount of specialized and generalized RFD$_c$s across all missing rates, showing some peaks only in a few datasets (i.e., datasets with *IDs* 4 and 11). One of the reasons why the pipeline exhibits a higher number of generalizations is that only complete tuples and/or those that are completed through the imputation process, are considered in the discovery process. This represents a scenario where a discovery process is performed on

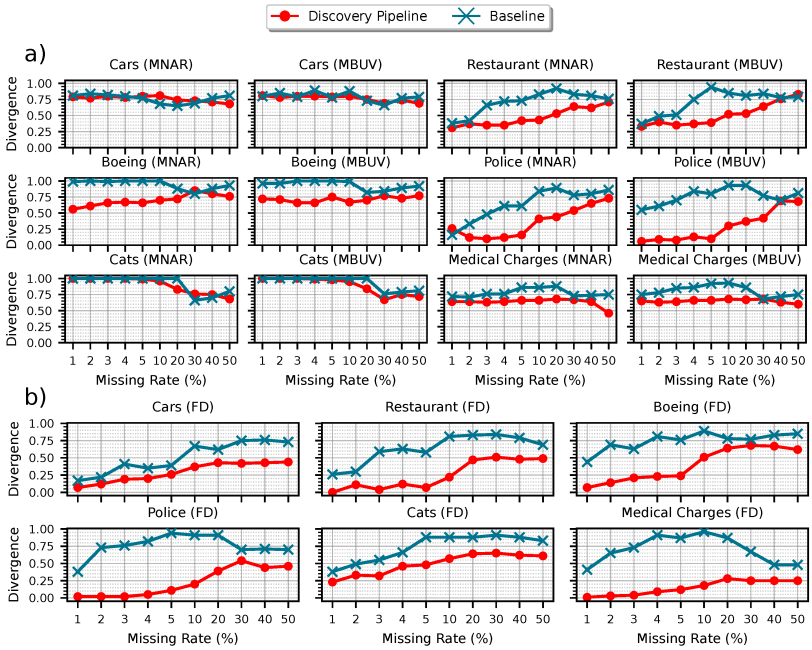


Figure 6.11: Comparing the baseline and the *Discovery Pipeline* with other missing mechanisms (a) and FDs (b).

a subset of tuples of the oracle dataset. In general, the higher percentage of both specialized and generalized RFD$_c$s guaranteed by the *Discovery Pipeline* highlights its ability to resemble more accurately the oracle RFD$_c$s.

Finally, the *Discovery Pipeline* allowed the discovery of *New* RFD$_c$s on most datasets. This trend increases in accordance with the missing rate. This indicates that the exclusion of incomplete tuples, despite reducing the available information, leads to fewer violations of RFD$_c$s, possibly yielding new valid RFD$_c$s on the dataset.

To summarize, these results demonstrate how the *Discovery Pipeline* is able to more accurately identify a set of RFD$_c$s that, in comparison to the baseline, is more similar to the set of oracle RFD$_c$s. In fact, one of the most important achievements of the pipeline consists in the mitigation of the overall percentage of RFD$_c$s *Not Found*. To provide a clearer perspective on this aspect, Figure 6.9 shows, for each missing rate, the average reduction, in percentage, of RFD$_c$s *Not Found* achieved by the *Discovery Pipeline*

with respect to the baseline. We can notice that, as the missing rate increases, the decrease of RFD$_c$s *Not Found* is more evident. In other words, more RFD$_c$s in the oracle have been identified by the *Discovery Pipeline* with respect to the baseline, either in their original or generalized/specialized version. This demonstrates how the *Discovery Pipeline* tends to retrieve dependencies that would be otherwise lost with the baseline strategy, especially when the data at hand is more challenging due to the high number of MVs. In particular, the reduction of RFD$_c$s *Not Found* ranges from 50.83% with 1% of missing rate to 74.35% with 50% of missing rate. As discussed in Section 6.2.1.3, we also considered a divergence measure to evaluate the difference between the sets of RFD$_c$s discovered by the compared strategies and the oracle RFD$_c$s. Figure 6.10 compares the results obtained by the two approaches (i.e. *Discovery Pipeline* in blue and the baseline in red). We observe that in most cases the *Discovery Pipeline* led to the discovery of sets of RFD$_c$s that were closer to the oracle RFD$_c$s in the majority of experiments, except only for a few datasets (i.e., *IDs* 1, 4, and 12) on a limited number of missing rates. In these cases, the *Discovery Pipeline* tends to discover a greater number of *New* RFD$_c$s due to the exclusion of incomplete tuples.

Supplementary experiments

As done with the *Imputation Pipeline*, we also considered scenarios with different missing mechanisms (i.e., MNAR and MBUV). Due to space constraints, we will only report the results obtained with the divergence metric. Figure 6.11(a) shows that the two strategies exhibited a similar behavior also in this more complex scenario. Finally, we also conducted supplementary experiments using another type of profiling metadata, i.e., the canonical FD. Figure 6.11(b) shows the results in terms of divergence from the oracle FDs. Also in this case, the *Discovery Pipeline* manages to extract a set of dependencies closer to the oracle FDs, and it is never outperformed by the baseline. This further highlights the usefulness of the *Discovery Pipeline*, demonstrating how the evaluated strategy can be generalized even with other types of metadata that have more stringent constraints than RFD$_c$s.

6.2.5 *Discussion*

In this section, we provide a discussion about the outcomes of our study for both RQs.

RQ1. The results described in Section 6.2.3 led us to the conclusion that, in incremental scenarios, it is crucial to update RFD$_c$s in accordance with changes in the data to effectively leverage them during the imputation process. As proved by the results of the baseline approach, failing to update the RFD$_c$s can result in significant performance degradations, particularly when RFD$_c$s become outdated and primarily exist in a highly generalized form. On the other hand, the results of the *Imputation Pipeline* demonstrated that updating RFD$_c$s lead to more representative and specialized dependencies, thus improving the overall imputation quality, with *RMSE* reductions up to 49.37%. One relevant example that highlights the importance of updating RFD$_c$s is represented by the *Cats* dataset (*ID* 4), in which we can notice a significant difference in terms of *RMSE* between the two strategies. By analyzing the dataset, we noticed that the first 280 rows of it represented instances of a single cat breed, while the remaining ones represented instances of other two breeds. Since the baseline considered RFD$_c$s discovered only in the first 20% of the dataset (i.e., 210 rows), its imputations were based on RFD$_c$s that were not completely reflecting the dataset domain. Our evaluation also compared the *Imputation Pipeline* and its variants (i.e., *PipelineNoRev* and *Hybrid*), showing that re-evaluating previous imputations made on the basis of outdated RFD$_c$s leads to significant improvements in the imputation quality, with *RMSE* reductions up to 46.21%. One example that supports this finding was observed on the *Police* dataset, which records police shooting incidents. Initially, the first batches of tuples led to the discovery of the RFD$_c$ $\mathsf{County}_{\leq 3.0} \rightarrow \mathsf{Classification}_{\leq 3.0}$ as a representative relationship, suggesting that similar counties imply the same type of incident. Such an RFD$_c$ led to the incorrect imputation t_{137}[Classification] ="Gunshot", based on a candidate with the same County. However, successive batches of tuples rectified the considered RFD$_c$, specializing it. Since such an RFD$_c$, despite being more representative, made more challenging the identification of

a suitable candidate for t_{137}, the *Hybrid* strategy left the previous incorrect imputation. On the other hand, by restoring the MV, the *Imputation Pipeline* avoided considering such a value for further iterations. This finding highlights the importance of developing appropriate methods to reassess imputations based on RFD$_c$s that have become invalid, paving the way for future research directions. Furthermore, we showed that the above findings still hold when adopting MNAR and MBUV approaches for injecting MVs, guaranteeing *RMSE* reductions up to 51.26% and 51.59%, respectively. Finally, we also proved that updating RFD$_c$s brings benefits even to other imputation algorithms, such as those leveraging RFD$_c$s only to support the imputation process (i.e., DERAND), but not for retrieving the candidate value for each imputation, which limits the reduction of *RMSE* up to 6.0%.

O1: Our empirical evaluation demonstrated that dynamically updating the RFD$_c$s employed in the imputation process significantly reduce errors (33.15% on average). Moreover, re-evaluating previous imputations based on outdated metadata further reduces errors (16.4% on average).

RQ2. The results described in Section 6.2.4 demonstrate that, in incremental scenarios, imputation strategies can support discovery processes by mitigating the impact of MVs on discovery results. With respect to the baseline, the *Discovery Pipeline* enabled the identification of more RFD$_c$s that exactly matched the oracle RFD$_c$s, with an overall average percentage of RFD$_c$ *Found* of 29.20%, although in some cases they were discovered in a *Generalized* (17.10%) or *Specialized* (27.46%) form. However, the presence of *New* RFD$_c$s (49.37%) poses a significant challenge, as this type of dependency does not reflect the actual data relationships. Nevertheless, the *Discovery Pipeline* significantly reduced the number of RFD$_c$s *Not Found* (i.e., with reductions up to 74.35%), resulting in the discovery of RFD$_c$ sets that more closely resemble the oracle RFD$_c$s compared to the baseline (i.e., with divergence reductions up to 81.5%). One relevant example that highlights the problem of MVs in discovery processes is represented by the dataset *Telemetry* (*ID* 6), in which the baseline is always unable to find any oracle RFD$_c$. Among the latter, we considered as an example the

RFD$_c$ φ: $\mathsf{CO}_{\leq 0.0001} \rightarrow \mathsf{Light}_{\leq 0.5}$, and we tried to assess its validity on the dataset with MVs. Within the latter, we considered the tuple t_{2860}, which has CO=0.0024 and an MV for the attribute Light. We observed that this single MV results in a violation of φ on over 220 tuples (i.e., the 7.33% of the dataset), leading to the invalidation of φ. Furthermore, the generalizability of these findings was proven by the experiments with MNAR and MBUV approaches for injecting MVs, in which the *Discovery Pipeline* achieved divergence reductions of up to 40.08% and 62.58%, respectively. Finally, the approach also proved to be effective with FDs, yielding divergence reductions of up to 67.66%, demonstrating that the benefits are not limited to a single type of metadata.

O2: Our empirical evaluation demonstrated that reducing missing values through imputation significantly enhances discovery and enables to obtain a set of dependencies closer to the ideal set (27.16% on average).

Overall, the above findings suggest that the integration of imputation and discovery processes constitutes a robust and mutually beneficial strategy in incremental settings. This also provides insights for time-series scenarios, where data continuously evolve with new sensor readings. While in Chapter 5 we considered this scenario in a static manner, the outcomes of this study suggest a way to apply RFD frameworks, such as RIFT, in dynamic settings.

6.2.5.1 *Limitations*

This study is limited on specific strategies found in the literature for performing imputation and incremental RFD$_c$ discovery (i.e., RENUVER/DERAND and INDIBITS) along with their specific configurations. The pipelines defined in this study incorporate tailored design choices to facilitate a comprehensive understanding of the mutual contribution between these processes. Moreover, the imputation correction step implemented in the pipelines is limited to a "naïve" restoration of MVs.

To summarize, we performed an empirical study, driven by two RQs, to analyze the usefulness of the cooperation between data profiling and RFD-based imputation algorithms in dynamic scenarios. To this end, we devised two pipelines to enable mutual support between the imputation and RFD_c discovery processes. The empirical evaluation yielded valuable insights into the problem at hand. In particular, it highlighted the need to update RFD_cs over time and to reassess imputed values based on outdated RFD_cs, achieving a trade-off between the potential propagation of errors resulting from evolving data and the desire to maintain consistent imputed values over time. A second notable finding is the impact of MVs on the data used for RFD_c discovery, as a high presence of them can lead to an abundance of RFD_cs that can distort the representation of data relationships.

SUMMARY OF PART II

In this part of the thesis, we discussed the problem of Data Imputation, and we showed how dependencies can be exploited to impute missing values. Then, we provided a detailed literature review of existing imputation techniques, categorizing the approaches for relational data and for time series data. Finally, we highlighted the current challenges related to the application of RFDs in this scenario. To address the identified challenges, this part proposed two contributions. The first one is RIFT, an RFD-based imputation framework tailored for time series data. RIFT deals with the scalability issues of RFD-based discovery and RFD-based imputation when dealing with high-dimensional data, which is the case of multivariate time series collected from several sensors. To address this problem, RIFT introduces several optimizations in both the ways RFDs are discovered and applied in the imputation process. Through an experimental evaluation, we show that RIFT significantly reduces execution times with respect to a traditional RFD-based approach, and that it can represent a useful preprocessing tool capable of improving state-of-the-art imputation approaches. The second contribution is an empirical study aimed at evaluating the synergies between incremental Data Profiling and metadata-driven imputation processes in dynamic scenarios. Our study highlighted the need of updating metadata over time and of re-evaluating imputations made on the basis of outdated metadata. Moreover, we showed that mitigating missing values through imputation approaches allows to significantly improve FD and RFD discovery results. Overall, the contributions presented in this part of the thesis pursued an existing and promising research direction, which entails the application of dependencies to impute missing values. The outcomes of these contributions pave the way for the identification of several future research directions, starting from the definition of a generalized framework capable of handling both dynamic contexts and heterogeneous data.

Part III

APPLYING DATA PROFILING IN OTHER USE CASES

7

APPLYING DATA PROFILING IN OTHER USE CASES

As discussed in Chapter 2, data profiling enables the discovery of useful patterns and relationships that can be leveraged in a wide range of use cases. Thus, we also provide further metadata-driven contributions in other application scenarios. In particular, this Chapter will discuss two works that explore the use of profiling metadata to detect concept drift (Section 7.1) and to perform query relaxation (Section 7.2).

7.1 CONCEPT DRIFT DETECTION

Machine Learning (ML) models are increasingly relied upon for a multitude of tasks. These include personalized recommendation systems [150], natural language processing applications, such as virtual assistants [118] and sentiment analysis applications [38], and for image and speech recognition tasks [7, 14]. Moreover, ML models are deployed for more delicate and critical tasks, where inaccurate predictions can lead to potentially severe consequences. As an example, anomaly detection is crucial across various domains, such as detecting fraud [100], monitoring business processes [162], ensuring network security [98], monitoring patients [160], and predicting maintenance [62]. However, as ML models transition from the training phase to real-world deployment, they face the challenge of sustaining their effectiveness. Initially, a model may exhibit robust performance, as it captures patterns and relationships within the training data. As time progresses, the underlying assumptions may no longer hold, possibly leading to wrong recommendations.

One of the main reasons for model degradation is *concept drift*, a phenomenon that refers to changes in the underlying function that generates data. Concept drift occurs due to various factors, including evolving user preferences, external environmental changes, and other alterations in the domain dynamics. Thus,

given the vast reliance on ML models we experience nowadays, it is crucial to detect changes to adopt countermeasures, such as retraining the model on a more representative, up-to-date set of data. Such detection task represents the goal of *Concept Drift Detection* approaches. In what follows, we provide an overview of existing concept drift detection approaches (Section 7.1.1) and a formal definition of concept drift (Section 7.1.2). Then, we present the first metadata-driven concept drift detection method (Section 7.1.3).

7.1.1 *Literature Review*

Concept drift detection techniques typically fall into two main categories: *performance-based* and *data distribution-based* [20]. The former relies on monitoring the model's error rate to detect potential drift. A well-known method in this category is the Drift Detection Method (DDM) [81]. It continuously monitors the probability of misclassification and its associated standard deviation. Over the years, several extensions of this approach have been proposed. For example, Baena et al. [13] proposed the Early Drift Detection Method (EDDM) by focusing on the distance between errors rather than just the error rate. This approach resulted more effective in handling gradual drifts, improving DDM especially with slower transitions. In [18], the authors defined a methodology to remove older instances linked to prolonged concepts, aiming to detect drifts earlier in the process, ultimately leading to improved model accuracy. Other proposals focus on comparing time windows for detecting drift [99, 125, 139].

On the other hand, data distribution-based approaches aim to detect significant shifts between the distributions of sampled data. In [88], the authors review distance measures for assessing numerical data shifts, recommending the Hellinger distance [97]. For instance, Ditzler and Polikar [65] used this distance to identify both gradual and abrupt changes across different data batches. Their strategy averages the Hellinger distance of individual features as drift measure and adopts an adaptive threshold to trigger warnings. Instead, Principal Component Analysis (PCA) has been conceived as a promising technique for high-dimensional data. The authors of [151] proposed a method

involving the creation of a reference window filled with initial data samples. Then, PCA is applied to extract the top k principal components (PCs), and both reference and test window data are projected onto them. The final change score is the maximum Kullback-Leibler divergence among the ones of each component. This approach was extended in a more recent work [89] by employing the Hellinger distance since it provides a more robust absolute measure. A variant of this measure is employed in [8], where a hybrid framework is proposed to adapt the parameters of drift detectors with tunable thresholds based on the characteristics of a data stream. This approach considers not only the model's accuracy and distributional changes but also additional information, such as the number of observations since the last detected drift.

In addition, there exist alternative approaches, such as Discriminative Drift Detector (D3) [92] and Label Dependency Drift Detector (LD3) [93], which do not rely on labels or feedback on predictions. D3 [92] employs a discriminative classifier to detect concept drift by monitoring changes in the feature space. A sliding window maintains the temporal order of samples, and logistic regression is applied to distinguish between old and new data. Concept drift is detected when the Area Under Curve (AUC) exceeds a threshold, indicating a significant difference between new and old data. Instead, LD3 [93] is tailored for multi-label data streams, and aims to detect concept drift by using a label influence ranking method, which exploits temporal relationships between labels.

In general, performance-based approaches have the advantage of being triggered only when the model performance is affected, but they require quick feedback on the predictions made. Instead, data distribution-based approaches consider only the distributions of data samples. However, since changes in distribution may not consistently affect the model, these approaches are susceptible to false alarms [20]. Moreover, we argue that existing distance measures may suffer a loss of effectiveness in detecting drift when this does not affect the data distribution or other statistics, but only the underlying relationships between features.

For this reason, the investigation of new approaches that consider other types of information/patterns within data can be

considered a challenging research direction. Another requirement that these approaches should satisfy is to avoid the reliance on qualitative descriptors like "abrupt" and "gradual", which have been shown to have limitations due to their dependence on arbitrary boundaries [182]. This leads to the necessity of estimating the *drift magnitude* by means of quantitative measures.

7.1.2 *Problem Overview*

Over the years, several definitions of *concept* and *concept drift* have been provided, leading to a lack of a standard definition. In this thesis, we comply with the definitions reported by Bayram et al. [20], representing a general and unified probabilistic view of concept.

In a supervised ML setting, each data instance is composed of a feature vector X and a target variable y. Formally, a concept drift occurs if there is a change in the joint distribution $P(X, y)$ between two time instants τ and $\tau + w$:

$$P_\tau(X, y) \neq P_{\tau+w}(X, y)$$

The latter can be decomposed by using the Bayesian Rule:

- $P_\tau(X|y)$ expresses the class-conditional probability density distribution, capturing the likelihood of encountering specific input data given a target label. It expresses how the features are distributed when considering a specific outcome;
- $P_\tau(y|X)$ is the posterior probability distribution of the target labels, capturing the likelihood of observing specific target labels given the features. It represents the probability of different outcomes after considering the available input data;
- $P_\tau(X)$ is the probability distribution of the input data and encapsulates the likelihood of encountering various sets of features within a dataset;

- $P_\tau(y)$ is the prior probability distribution of the target labels and expresses the likelihood of observing particular outcomes without considering the input data.

According to which of these distributions change between τ and $\tau + w$, it is possible to categorize the concept drift:

- $P_\tau(y|X) \neq P_{\tau+w}(y|X)$: a variation in the posterior probability distribution that represents a change in the model's predictions driven by new observations. It has a direct impact on the model's performance and may result in the model no longer being able to accurately predict. In this case, two types of drift can be defined: *Real concept drift*, when the change in the posterior probability might be associated with changes in $P(X)$, and *Actual drift*, when $P(X)$ remains unaltered.

- $P_\tau(X) \neq P_{\tau+w}(X)$: a variation in the probability distribution of the input data that represents common changes in data during the model deployment. Typically, changes in $P(X)$ affect model performance, since ML models struggle to adapt to unfamiliar data [53]. This kind of drift is denoted as *Covariate shift*, but it is termed *Virtual drift* when the changes do not affect the decision boundary.

- $P_\tau(Y) \neq P_{\tau+w}(Y)$: a change in the prior probability distribution that could impact the prediction performance, especially when there is a noteworthy alteration in the distribution of classes. This kind of drift is termed *Prior-probability shift* and also comprises cases when new classes are introduced or existing classes cease to exist.

To categorize concept drift, it is also possible to consider the type of shift, which can manifest in various patterns:

- *Gradual drift* represents a progressive evolution from one concept to another one over time;

- *Abrupt/Sudden drift* represents an immediate transition from a concept to another one;

- *Incremental drift* represents a slow replacement of an old concept by a new one in a continuous manner, without clearly defining a boundary between them;
- *Recurring drift* represents a phenomenon where previously observed patterns reoccur after a time interval.

7.1.3 *An* RFD*-based approach for concept drift detection [44]*

Considering changes in data distribution represents the most popular concept drift detection technique that does not require any analysis of model predictions. These approaches try to establish if more recent data comes from a different distribution with respect to the older one, risking overlooking drifts that are not immediately evident in the overall data distribution. This can be due to the fact that concept drift does not always produce changes in the statistical properties of data, but rather shifts in its underlying relationships.

To this end, we investigate whether, by capturing the evolution of relationships through metrics based on the analysis of RFD_c evolution, it is possible to achieve any findings for determining concept drift. A first connection between dependencies and the performance of machine learning models was discussed in [115], where these types of profiling metadata have been used to characterize the existence of a function that provides an upper bound for classification accuracy. RFD_cs may evolve as generalizations, specializations, or may be invalidated (i.e., they do not hold on the updated data and are neither specialized nor generalized). These evolutions can be evaluated to quantify the divergence between two sets of RFD_cs. In what follows, we detail how RFD_cs can be exploited for concept drift detection (Section 7.1.3.1), the proposed approach (Section 7.1.3.2), and the experimental evaluation performed to evaluate it (Section 7.1.3.3).

7.1.3.1 RFDs *and concept drift*

Let us consider the sets Σ and Σ' of RFD_cs holding on a relation instance r of R in two given time instants τ and $\tau + 1$, respectively. The analysis of how RFD_cs change between the two time instants can be accomplished in two different perspectives: evaluating

φ	Σ
φ_0	$\text{Model}_{\leq 0}$, $\#\text{Owners}_{\leq 1} \rightarrow \text{Price}_{\leq 300}$
φ_1	$\text{Model}_{\leq 1}$, $\text{Year}_{\leq 0} \rightarrow \text{Price}_{\leq 300}$
φ_2	$\text{Year}_{\leq 0}$, $\text{Price}_{\leq 300} \rightarrow \text{Model}_{\leq 0}$
φ_3	$\text{Model}_{\leq 1}$, $\text{Price}_{\leq 300} \rightarrow \text{Year}_{\leq 1}$
φ_4	$\text{Model}_{\leq 0}$, $\#\text{Owners}_{\leq 1} \rightarrow \text{Year}_{\leq 1}$
φ_5	$\text{Price}_{\leq 300} \rightarrow \#\text{Owners}_{\leq 0}$
φ_6	$\text{Year}_{\leq 0} \rightarrow \#\text{Owners}_{\leq 0}$

φ	Σ'
φ'_0	$\text{Model}_{\leq 0}$, $\#\text{Owners}_{\leq 1}$, $\text{Year}_{\leq 0}$, $\rightarrow \text{Price}_{\leq 300}$
φ'_1	$\text{Year}_{\leq 0}$, $\text{Price}_{\leq 300} \rightarrow \text{Model}_{\leq 0}$
φ'_2	$\text{Model}_{\leq 1}$, $\text{Price}_{\leq 300}$, $\#\text{Owners}_{\leq 1}$, $\rightarrow \text{Year}_{\leq 1}$
φ'_3	$\text{Price}_{\leq 300} \rightarrow \#\text{Owners}_{\leq 2}$
φ'_4	$\text{Year}_{\leq 3}$, $\#\text{Owners}_{\leq 1} \rightarrow \text{Model}_{\leq 2}$

Table 7.1: An example of RFD_c sets holding at two time instants τ and $\tau + 1$.

how much Σ is changed with respect to Σ', and vice versa. In what follows, we formalize the characterization of all possible changes on the sets of holding RFD_cs according to these two scenarios.

Definition 22 (Shift from Σ to Σ'). *To quantify the degree of divergence between Σ and Σ', it is necessary to evaluate each RFD_c $\varphi \in \Sigma$ to verify if φ is somehow related to any RFD_c $\varphi' \in \Sigma'$. Specifically, $\forall \varphi \in \Sigma$:*

- *φ can also belong to Σ';*
- *φ can be specialized by at least one $\varphi' \in \Sigma'$;*
- *φ can neither belong to Σ' nor be specialized by any $\varphi' \in \Sigma'$, meaning that φ has been invalidated.*

As an example, let us consider the sets of RFD_cs Σ and Σ' provided in Table 7.1, which are discovered at time τ and $\tau + 1$, respectively. By examining the RFD_cs in Σ, we can quantify the shift as follows:

- the RFD_c φ_2 does not change, since it also belongs to Σ';
- 5 RFD_cs are specialized in Σ'. In particular, φ_0 and φ_1 are specialized by φ'_0, φ_3 and φ_4 by φ'_2, and φ_5 by φ'_3;
- the RFD_c φ_6 is invalidated, since it does not belong to Σ' and there is no RFD_c in Σ' that specializes it.

In other words, this type of analysis allows to evaluate the change of an older RFD_c set compared to a more recent one. Conversely, as mentioned, it is also possible to perform an analysis

evaluating the change of a newer RFD$_c$ set compared to an older one.

Definition 23 (Shift from Σ' to Σ). *To quantify the degree of divergence between Σ' and Σ, it is necessary to evaluate each RFD$_c$ $\varphi' \in \Sigma'$ to verify if φ' is somehow related to any RFD$_c$ $\varphi \in \Sigma$. Specifically, $\forall \varphi' \in \Sigma'$:*

- *φ' can also belong to Σ;*
- *φ' can be generalized by at least one $\varphi \in \Sigma$;*
- *φ' can neither belong to Σ nor be generalized by any $\varphi \in \Sigma$, meaning that φ' is a new RFD$_c$.*

As an example, consider the sets of RFD$_c$s Σ and Σ' in Table 7.1. By examining the RFD$_c$s in Σ', we can quantify its shift as follows:

- the RFD$_c$ φ'_1 does not change, since it also belongs to Σ;
- 3 RFD$_c$s are generalized in Σ. In particular, φ'_0 is generalized by φ_0 and φ_1; φ'_2 by φ_3 and φ_4; and φ'_3 by φ_5;
- the RFD$_c$ φ'_4 is a new RFD$_c$, since it does not belong to Σ and there is no RFD$_c$ in Σ that generalizes it.

From this characterization of possible changes between Σ and Σ', a more fine-grained analysis can be introduced by considering further properties of generalizations and specializations. We will present some of these characterizations in the following section.

In what follows, we describe the proposed concept drift detection methodology. In particular, we first discuss the reason why analyzing shifts in RFD$_c$s can provide effective support in detecting concept drift. After a general overview of the proposed approach, we describe it by detailing each involved step.

7.1.3.2 *An RFD$_c$-based concept drift detection approach*

We now introduce an approach tailored for supporting supervised ML models by detecting concept drift. This is accomplished through analyses of the shift in terms of RFD$_c$s during the model deployment, in order to warn about the necessity of retraining the model if the shift is severe. Since concept drift can occur only

for some of the target labels, our approach evaluates the shift for each class, so it can provide more detailed insights into which specific classes are affected by the drift.

Figure 7.1 shows an overview of the proposed approach. As we can see, it is based on two main steps, whose operation flows are highlighted in two different colors (i.e., black and red arrows). In particular, in the training phase, the model that has to be monitored over time is trained on the available data. At the same time, a discovery process is performed on the same data, through which it is possible to extract the set of holding RFD$_c$s for each target label. In the next step, the trained model is deployed and starts to make predictions on incoming data.

As previously discussed, the model's performance can degrade over time due to a drift occurring in new data. However, since there is no knowledge about the predictions made, our approach entails conducting periodical checks to verify whether the data at hand has significantly changed. To accomplish this, a new RFD$_c$ discovery process is performed on an updated dataset, which consists of the concatenation of the training data and the new instances predicted by the model. Then, for each class, the original set of RFD$_c$s holding on the training data is compared with the updated set of RFD$_c$s. To perform this comparison, we leverage several RFD-based metrics that are defined later. If through these metrics a significant shift is highlighted, it may be necessary to retrain the model. In what follows we will detail the operations that allow our approach for *Collecting meaningful* RFD$_c$s and *Evaluating* concept drift.

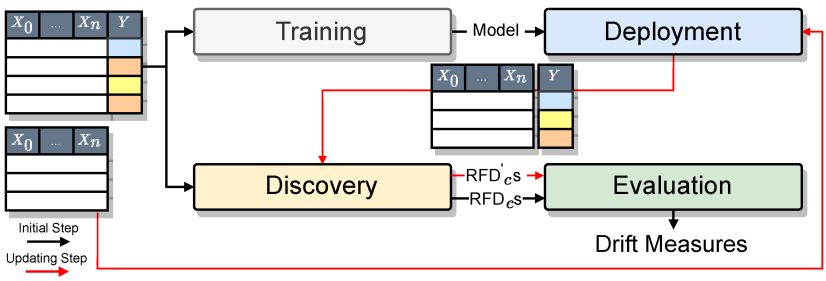


Figure 7.1: A general overview of the proposed approach.

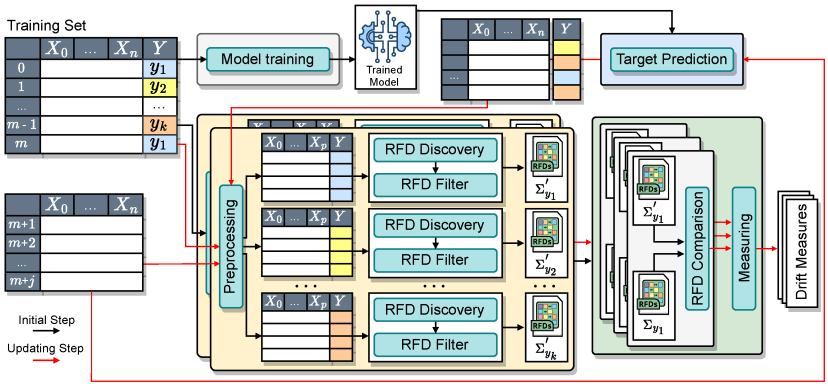


Figure 7.2: An overview of the initialization and updating steps underlying the proposed approach.

COLLECTING MEANINGFUL RFD$_c$S. Figure 7.2 shows in detail the proposed approach. By following one of the main goals of it, sets of minimal RFD$_c$S are collected in different steps of the whole process, i.e., the *initial* and the *updating* ones. Nevertheless, independently from them, the proposed approach underlies three main phases for collecting meaningful RFD$_c$S: i) Preprocessing, ii) RFD$_c$ Discovery, and iii) RFD$_c$ Filtering; as highlighted by the yellow box in Figure 7.2.

i) Preprocessing. The preprocessing phase aims to prepare the dataset in input for the RFD$_c$ discovery. The main operations performed in this phase are the selection of the most relevant features, the organization of data into equivalence classes, and the splitting of data. The first two operations aim to reduce the number of features and the variability in the data, allowing for RFD$_c$ discovery processes to quickly focus on the most informative features, by also avoiding data having a too fine-grained representation. Specifically, for the first operation, we leverage mutual information-based feature selection [109] to maintain only the most relevant attributes for the follow-up analyses. This technique was elected as a distinguished technique for its ability in capturing any type of relationship [71], and for its robustness to noise and data transformations, yielding effective selection of features [129]. The second operation is applied to attributes with a high variability of values, which are arranged into equivalence

classes, ensuring that similar values are assigned to the same class [49] and replaced with the identifier of such an equivalence class. In the third operation, the dataset is split according to the k target labels. Thus, the output of this step consists of k subsets, each containing instances belonging to a specific class.

ii) RFD$_c$ *Discovery.* After the preprocessing phase, each of the k subsets is given as input to an RFD$_c$ discovery algorithm. In particular, for this approach, we leverage the DOMINO algorithm [45], which has the peculiarity of inferring by itself the distance constraints for each attribute. This feature is useful when the application domain is not well-known and there is uncertainty about the distance constraints to apply. However, any other RFD$_c$ discovery algorithm can be chosen based on the characteristics of the application domain. The output of this phase consists of k sets of RFD$_c$s, i.e., Σ_{y_i}, $i = 1, 2, \ldots, k$, where k is the number of target labels.

ii) RFD$_c$ *Filtering.* For a given dataset, the presence of holding RFD$_c$s can scale to the order of thousands or even more. In order to perform the subsequent comparison steps, in this phase we filter discovered dependencies, aiming at retaining, for each target label, only its most representative RFD$_c$s, i.e., those that distinguish it most from the others. To perform initial filtering of the original dependencies, we remove from each set Σ_{y_i} all the RFD$_c$s that are also present in the other sets Σ_{y_j}, with $j = 1, 2, \ldots, k$ and $i \neq j$. According to this, we can define Σ_{y_i} as:

$$\Sigma_{y_i} = \Sigma_{y_i} \setminus \bigcup_{i,j=1,2,\ldots,k \,\wedge\, i \neq j} \Sigma_{y_j}$$

Thus, the resulting k RFD$_c$ sets contain, for each class, only its unique dependencies. After that, we further filter RFD$_c$s by leveraging the concept of minimality. In particular, for each set Σ_{y_i}, we maintain only RFD$_c$s that are minimal with respect to all the RFD$_c$s belonging to other sets Σ_{y_j}, with $j \neq i$. This ensures that the RFD$_c$s of each class are unique and not related to those discovered for the other classes. The output of this phase consists of the updated k RFD$_c$ sets, each encapsulating the most representative dependencies for its respective target class.

As an example, let us consider a scenario in which two target labels are involved, namely y_i and y_j, respectively. Thus, supposing that after the discovery process at given time τ, the following resulting RFD$_c$s are provided:

- $\Sigma_{y_i} = \Sigma \cup \{\varphi_7\}$, where
  Σ refers to the set of RFD$_c$s shown in Table 7.1 and
  φ_7: $\mathsf{Year}_{\leq 2}$, $\mathsf{\#Owners}_{\leq 1} \rightarrow \mathsf{Model}_{\leq 3}$; and
- $\Sigma_{y_j} = \{\varphi_7, \varphi_8, \varphi_9\}$, where
  φ_8: $\mathsf{Model}_{\leq 2}$, $\mathsf{Price}_{\leq 300} \rightarrow \mathsf{\#Owners}_{\leq 0}$ and
  φ_9: $\mathsf{Year}_{\leq 3} \rightarrow \mathsf{Model}_{\leq 2}$.

Notice that the RFD$_c$ φ_7 is shared between the two sets Σ_{y_i} and Σ_{y_j} and that $\varphi_8 \in \Sigma_{y_j}$ is not minimal with respect to $\varphi_5 \in \Sigma_{y_i}$. Consequently, the sets Σ_{y_i} and Σ_{y_j} will become $\Sigma_{y_i} = \{\varphi_0, \varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6\}$ and $\Sigma_{y_j} = \{\varphi_9\}$ after the application of the filtering strategy.

Evaluating Drift through RFD$_c$s. As mentioned in Section 7.1.3.2, our approach entails the assessment of updated data to verify whether significant changes occur also in terms of RFD$_c$s. This detection serves as a trigger to retrain the model and enhance its adaptability to changing data dynamics. In particular, during the deployment, the model trained in the previous step starts making predictions on incoming data, and these predicted instances are incrementally integrated into the original training dataset. Thus, the new data (comprised of the original training data with true labels and the newly predicted instances) is reintroduced to the discovery module, where the same sequence of operations described in the previous section (i.e., preprocessing, discovery, and RFD$_c$ filtering) is performed.

Starting from the sets of RFD$_c$s resulting from the updated sets of data, the drift evaluation can be performed. Specifically, in this step, the proposed approach considers both the Σ_{y_i} and Σ'_{y_i} sets of RFD$_c$s, with $i = 1, 2, \ldots, k$ and k the number of target labels. Then, it consists of two main phases that underlie the whole evaluation process: i) RFD$_c$ Comparison and ii) Measuring; as highlighted by the green box in Figure 7.2.

i) RFD$_c$ *Comparison.* This phase takes care of comparing, for each target label, the original and updated RFD$_c$ sets, according to the

type of shifts among sets of RFD$_c$s (described in Section 7.1.3.1). Notice that, in what follows, we describe the comparison methodology between two sets of RFD$_c$s, since it can be easily generalized to the comparison of all sets of RFD$_c$s in Σ_{y_i} and Σ'_{y_i} with $i = 1, 2, \ldots, k$.

The comparison between Σ_{y_i} and Σ'_{y_i} provides different interpretations according to the direction followed during the process. In fact, the shift of RFD$_c$s can be performed from Σ_{y_i} and Σ'_{y_i} (see Definition 22) and vice versa (see Definition 23), yielding different types of changes between the two sets of RFD$_c$s. Nevertheless, it is possible to quantify the occurrence of specific types of changes involved during the comparison process as shown in Table 7.2.

Specifically, independently of the direction, there can be a certain number of RFD$_c$s that appear equal in both sets, namely *Imm*. Instead, if the comparison is performed from Σ_{y_i} to Σ'_{y_i}, then it is possible to quantify the number of RFD$_c$s in Σ_{y_i} that are:

- generically specialized in Σ'_{y_i}, namely *Spec*;
- specialized in Σ'_{y_i} by adding attributes on the LHS, namely $Spec_{\supset}$, $Spec_{\supset_{\Rightarrow}}$, $Spec_{\supset_1}$;
- specialized in Σ'_{y_i} by varying thresholds only, namely $Spec_{eq}$, $Spec_{eq_{\Leftarrow}}$, $Spec_{eq_{\Rightarrow}}$, $Spec_{eq_{\Leftrightarrow}}$;
- invalidated, i.e., neither present nor specialized in Σ'_{y_i}, namely *Inv*.

Notice that details about the different criteria of specializations/generalizations are provided in Table 7.2.

As an example, let us consider the two sets of RFD$_c$s Σ and Σ' shown in Table 7.1, which can be denoted as Σ_{y_i} and Σ'_{y_i} since they are associated to a single label. Thus, by performing a comparison from Σ_{y_i} to Σ'_{y_i}, it is possible to say that there is just one *Imm* RFD$_c$, i.e., φ_2 and one *Inv* RFD$_c$, i.e., φ_6. Moreover, among the five *Spec* RFD$_c$s: φ_0, φ_1, φ_3 φ_4, and φ_5; only the latter, compared with φ'_3, satisfies the $Spec_{eq}$ and the $Spec_{eq_{\Rightarrow}}$ criteria since the specialization is driven by a simple variation of the RHS threshold. Instead, the other four RFD$_c$s all satisfy the $Spec_{\supset_1}$ criterion since just one attribute is added on the LHS (i.e., φ_0 and φ_1 with φ'_0, and φ_3, φ_4 with φ'_2). Moreover, φ_0 and φ_3 also satisfy $Spec_{\supset}$ since

Notation	Definition
Σ_{y_i}	RFD$_c$s holding on the training instances with target class y_i
Σ'_{y_i}	RFD$_c$s holding on the training instances and the new predicted ones with target class y_i
Imm	Number of RFD$_c$s that belongs both to Σ_{y_i} and to Σ'_{y_i}
Inv	Number of RFD$_c$s in Σy_i that are not present in any form in $\Sigma' y_i$
$Spec$	Number of RFD$_c$s in Σ_{y_i} that are specialized in Σ'_{y_i}
$Spec_{\supset}$	Number of RFD$_c$s in Σ_{y_i} that are specialized in Σ'_{y_i} by an RFD$_c$ with the same thresholds on the attributes in common
$Spec_{\supset_{\Rightarrow}}$	Number of RFD$_c$s in Σ_{y_i} that are specialized in Σ'_{y_i} by an RFD$_c$ with lower or equal thresholds on the attributes in common on the LHS, and higher threshold on the RHS
$Spec_{\supset_1}$	Number of RFD$_c$s in Σ_{y_i} that are specialized in Σ'_{y_i} by an RFD$_c$s with only an additional attribute on the LHS
$Spec_{eq}$	Number of RFD$_c$s in Σ_{y_i} that are specialized in Σ'_{y_i} by an RFD$_c$ with the same attributes
$Spec_{eq_{\Leftarrow}}$	Number of RFD$_c$s in Σ_{y_i} that are specialized in Σ'_{y_i} by an RFD$_c$ with the same attributes, with the same threshold on the RHS and lower or equal thresholds on the LHS (with at least one lower)
$Spec_{eq_{\Rightarrow}}$	Number of RFD$_c$s in Σ_{y_i} that are specialized in Σ'_{y_i} by an RFD$_c$ with the same attributes, with the same thresholds on the LHS and greater or equal threshold on the RHS (with at least one greater)
$Spec_{eq_{\Leftrightarrow}}$	Number of RFD$_c$s in Σ_{y_i} that are specialized in Σ'_{y_i} by an RFD$_c$ with the same attributes, with greater threshold on the RHS and lower or equal thresholds on the LHS (with at least one lower)
New	Number of RFD$_c$s in Σ'_{y_i} that are not derived from any RFD$_c$ in Σ_{y_i}
Gen	Number of RFD$_c$s in Σ'_{y_i} that are generalized in Σ_{y_i} in any form
$Gen_{\subset}$	Number of RFD$_c$s in Σ'_{y_i} that are generalized in Σ_{y_i} by an RFD$_c$ with the same thresholds on the attributes in common
$Gen_{\subset_{\Rightarrow}}$	Number of RFD$_c$s in Σ'_{y_i} that are generalized in Σ_{y_i} by an RFD$_c$ with the same threshold on the LHS and lower or equal thresholds on the RHS on the attributes in common (with at least one lower)
$Gen_{\subset_1}$	Number of RFD$_c$s in Σ'_{y_i} that are generalized in Σ_{y_i} by an RFD$_c$ with only one attribute less on the LHS
Gen_{eq}	Number of RFD$_c$s in Σ'_{y_i} that are generalized in Σ_{y_i} by an RFD$_c$ with the same attributes involved
$Gen_{eq_{\Leftarrow}}$	Number of RFD$_c$s in Σ'_{y_i} that are generalized in Σ_{y_i} by an RFD$_c$ with the same attributes, with the same threshold on the RHS and greater or equal thresholds on the LHS (with at least one greater)
$Gen_{eq_{\Rightarrow}}$	Number of RFD$_c$s in Σ'_{y_i} that are generalized in Σ_{y_i} by an RFD$_c$ with the same attributes, the same thresholds on the LHS, and lower or equal thresholds on the RHS (with at least one lower)
$Gen_{eq_{\Leftrightarrow}}$	Number of RFD$_c$s in Σ'_{y_i} that are generalized in Σ_{y_i}

Table 7.2: Reference table for the notations used to categorize different type of RFD evolution.

they maintain the same thresholds on common attributes when compared with φ'_0 and φ'_2, respectively.

On the other hand, if the comparison is performed from Σ'_{y_i} to Σ_{y_i}, then it is possible to quantify the RFD$_c$s in Σ'_{y_i} that are:

- generically generalized in Σ_{y_i}, namely *Gen*;
- generalized in Σ_{y_i} by removing attributes on the LHS, namely $Gen_{\subset}$, $Gen_{\subset_{\Rightarrow}}$, $Gen_{\subset_1}$;
- generalized in Σ_{y_i} by varying thresholds only, namely Gen_{eq}, $Gen_{eq_{\Leftarrow}}$, $Gen_{eq_{\Rightarrow}}$, $Gen_{eq_{\Leftrightarrow}}$;
- new, i.e., neither present nor generalized in Σ_{y_i}, namely *New*.

As an example, let us consider the same sets of RFD$_c$s involved in the previous example. By performing a comparison from Σ'_{y_i} to Σ_{y_i}, it is possible to say that there is just one *Imm* RFD$_c$, i.e., φ'_1 and one *New* RFD$_c$, i.e., φ'_4. Moreover, among the three *Gen* RFD$_c$s: φ'_0, φ'_2, φ'_3; only the latter, compared with φ_5, satisfies the Gen_{eq} and the $Gen_{eq_{\Rightarrow}}$ criteria since the generalization is driven by a simple variation of the RHS threshold. Instead, the other two RFD$_c$s all satisfy the $Gen_{\subset_1}$ criterion since just one attribute does not appear on the LHS (i.e., φ'_0 with φ_0 and φ_1, and φ'_2 with φ_3 and φ_4). Moreover, φ'_0 and φ'_2 also satisfy $Gen_{\subset}$ criterion since they maintain the same thresholds on common attributes when compared with φ_0 and φ_3, respectively.

To summarize, many criteria can be used to evaluate the shift between two sets of RFD$_c$s, according to possible invalidation of some RFD$_c$s, the generation of new RFD$_c$s, and/or different types of specializations/generalizations. In particular, the latter can allow the evaluation at different levels of granularity. Overall, the quantitative information provided by the several comparison criteria can be used to define different metrics to measure a possible drift into data, which is reflected in the variations on holding RFD$_c$s as will be described in what follows.

ii) Measuring. From the comparison phase, all the terms shown in Table 7.2 are obtained. Starting from them, the proposed approach can measure the shift in terms of RFD$_c$s according to a suite of proposed RFD-based metrics. Some of them evaluate the magnitude of the change of Σ_{y_i} with respect to Σ'_{y_i}, while others

Metric	Definition
D_1	$\frac{New + ((Gen - Gen_{eq}) \cdot 0.5) + (Gen_{eq} \cdot 0.05)}{\lvert\Sigma'_{y_i}\rvert}$
D_2	$\frac{New + ((Gen - Gen_{eq}) \cdot 0.5) + (Gen_{eq} \cdot 0.05) + Inv}{\lvert\Sigma_{y_i} \cup \Sigma'_{y_i}\rvert}$
D_3	$\frac{New + Inv}{\lvert\Sigma_{y_i} \cup \Sigma'_{y_i}\rvert}$
D_4	$\frac{New + (Gen_{\subset_1} \cdot 0.25) + ((Gen - \\ - Gen_{\subset_1} - Gen_{eq}) \cdot 0.5) + (Gen_{eq} \cdot 0.05)}{\lvert\Sigma_{y_i} \cup \Sigma'_{y_i}\rvert}$
D_5	$\frac{Inv + ((Spec - Spec_{eq}) \cdot 0.5) + (Spec_{eq} \cdot 0.05)}{\lvert\Sigma_{y_i}\rvert}$
D_6	$\frac{New + (Gen \cdot 0.5) + ((Gen_{eq_\Rightarrow} + Gen_{eq_\Leftarrow} + Gen_{eq_\Leftrightarrow}) \cdot 0.05)}{\lvert\Sigma_{y_i} \cup \Sigma'_{y_i}\rvert}$
D_7	$\frac{Inv + (Spec_1 \cdot 0.25) + ((Spec - Spec_{\supset_1} - \\ - Spec_{eq}) \cdot 0.5) + (Spec_{eq} \cdot 0.05)}{\lvert\Sigma_{y_i}\rvert}$
D_8	$\frac{New + Inv + ((Gen_{eq} - (Gen_{eq_\Rightarrow} + Gen_{eq_\Leftarrow} + \\ + Gen_{eq_\Leftrightarrow}) \cdot 0.5) + ((Gen_{eq_\Rightarrow} + Gen_{eq_\Leftarrow} + Gen_{eq_\Leftrightarrow}) \cdot 0.05)}{\lvert\Sigma_{y_i} \cup \Sigma'_{y_i}\rvert}$
D_9	$\frac{New + Inv + ((Gen - Gen_{\subset_1} - Gen_{eq}) \cdot 0.5) + (Gen_{\subset_1} \cdot 0.25) + (Gen_{eq} \cdot \\ \cdot 0.05) + ((Spec - Spec_{\supset_1} - Spec_{eq}) \cdot 0.5) + (Spec_{\supset_1} \cdot 0.25) + (Spec_{eq} \cdot 0.05)}{\lvert\Sigma_{y_i} \cup \Sigma'_{y_i}\rvert}$
D_{10}	$\frac{Inv + (Spec_{\supset_1} \cdot 0.1) + ((Spec - Spec_{\supset_1} - \\ - Spec_{eq}) \cdot 0.2) + (Spec_{eq} \cdot 0.02)}{\lvert\Sigma_{y_i}\rvert}$
D_{11}	$\frac{New + Inv + (Spec \cdot 0.1) + (Gen \cdot 0.1)}{\lvert\Sigma_{y_i} \cup \Sigma'_{y_i}\rvert}$
D_{12}	$\frac{Inv + ((Spec - Spec_{eq}) \cdot 0.3) + (Spec_{eq} \cdot 0.02)}{\lvert\Sigma_{y_i}\rvert}$

Table 7.3: RFD-based divergences.

consider the opposite direction, assessing the changes in Σ'_{y_i} with respect to Σ_{y_i}. Specifically, we defined two categories of metrics: the ones in the first category aim to quantify the *divergence* between two sets of RFD_cs, while those in the second category are inspired by machine learning, following a *confusion matrix-based* evaluation.

Table 7.3 presents the suite of *divergence* metrics, which range from simpler metrics to more refined ones. Specifically, D_5, D_7, D_{10}, and D_{12} are based on the characterizations of the shift from Σ_{y_i} to Σ'_{y_i}, while D_1 consider the shift from Σ'_{y_i} to Σ_{y_i}. Instead, the remaining metrics consider both perspectives, providing a single divergence value that summarizes the change in both sets. As shown in Table 7.3, the proposed metrics leverage coefficients to weight different types of RFD_c evolution, assigning greater importance to more substantial changes (e.g., invalidations and

new RFD$_c$s) while attributing a lower contribution to moderate ones (e.g., specializations and generalizations), aiming at accurately estimating the severity of changes among the RFD$_c$ sets. Notice that all the defined divergence metrics are normalized with respect to the total number of RFD$_c$s involved in the specific evaluation.

As an example, let us consider the two sets of RFD$_c$s Σ and Σ' shown in Table 7.1, which can also be denoted as Σ_{y_i} and Σ'_{y_i} since they are associated to a single label. Thus, according to Table 7.3, it is possible to apply the metric D_5 on the considered scenario to measure the divergence from Σ_{y_i} to Σ'_{y_i}:

$$D_5 = \frac{Inv + ((Spec - Spec_{eq}) \times 0.5) + (Spec_{eq} \times 0.05)}{|\Sigma_{y_i}|}$$

$$= \frac{1 + ((5-1) \times 0.5) + (1 \times 0.05)}{7} = \frac{1 + 2 + 0.05}{7} = 0.44$$

The second category of metrics is inspired by the confusion matrix, which is commonly employed for ML evaluation. In particular, we adapted the concepts of True\False Positives and True\False Negatives to evaluate the changes among the compared sets of RFD$_c$s. Our aim is to investigate whether metrics derived from our (re-)interpretation of the confusion matrix align with the trends observed in the actual model.

Table 7.4 shows the seven metrics we defined. Specifically, some of these interpretations (i.e., CF_1, CF_2, CF_3, CF_4) evaluate the degree of change from Σ_{y_i} to Σ'_{y_i}, while others (i.e., CF_5, CF_6, CF_7) consider the opposite perspective, describing the change from Σ'_{y_i} to Σ_{y_i}. To clarify this interpretation, let us consider the first metric (i.e., CF_1), through which we can identify *True Positives* as the number of RFD$_c$s that were in Σ_{y_i} and that are still in Σ'_{y_i}. In other words, these are the RFD$_c$s that we expected to have and that are correctly in Σ'_{y_i}. By following the same reasoning, we can consider *False Negatives* as the RFD$_c$s that were in Σ_{y_i} but not in Σ'_{y_i}. Instead, *False Positives* represent RFD$_c$s that were not in Σ_{y_i} but in Σ'_{y_i} (e.g., new RFD$_c$s). *True Negatives* are always equal to zero, since they represent RFD$_c$s not included in both Σ_{y_i} and Σ'_{y_i}.

Consequently, we considered the *F1-Measure*, *Precision*, and *Recall* metrics computed through this adapted interpretation of the confusion matrix. In general, lower values for *Precision*, *Recall*,

Metric	True Positives	False Positives	False Negatives
CF_1	Imm	New	Inv
CF_2	$Imm + \sum_{\forall G \in Gen_{eq}*} G$	$New + (Gen - \sum_{\forall G \in Gen_{eq}*} G)$	Inv
CF_3	$Imm + \sum_{\forall G \in Gen_{eq}*} G$	New	Inv
CF_4	Imm	$New + Gen$	Inv
CF_5	Imm	$Inv + Spec$	New
CF_6	Imm	$Inv + Spec$	$New + Gen$
CF_7	$Imm + Gen$	$Inv + Spec$	New

^ $Gen_{eq}* = \{Gen_{eq}, Gen_{eq_{\Rightarrow}}, Gen_{eq_{\Leftarrow}}, Gen_{eq_{\Leftrightarrow}}\}$.

Table 7.4: Metrics inspired by confusion matrix items.

and *F1-Measure* indicate a larger change between the two sets of RFD$_c$s. This is due to the fact that *False Positives* and *False Negatives* are associated with the possible evolutions of RFD$_c$s, while *True Positives* are associated with RFD$_c$s that do not change or slightly evolve with respect to the original ones.

As an example, let us consider the two sets of RFD$_c$s Σ and Σ' shown in Table 7.1, which can also be denoted as Σ_{y_i} and Σ'_{y_i} since they are associated to a single label. Thus, according to Table 7.4, it is possible to interpret confusion matrix items as defined by the metric CF_5 on the considered scenario: $TruePositives = Imm = 1$; $FalsePositives = (Inv + Spec) = (1 + 5)$; and $FalseNegatives = New = 1$. Thus, starting from these items, it is possible to compute the *Precision*, the *Recall*, and the *F1-Measure*.

7.1.3.3 *Experimental Evaluation*

In this section, we present the evaluation performed on the proposed RFD-based metrics, aiming at demonstrating their effectiveness in quantifying possible drifts within data. To this end, we investigate whether the trends of the RFD-based metrics are more correlated to the performance trend of the model with respect to other related approaches. As a matter of fact, a higher correlation would mean that the proposed metrics are better able to capture changes within a dataset, leading to reliable insights into whether a model's performance is declining or not. Different scenarios were analyzed to demonstrate that (i) RFD-based metrics provide

more reliable estimations than baseline distances and that (ii) RFD-based metrics also capture drifts within data when it does not affect the value distribution but the relationships among attributes.

BASELINE APPROACHES. To compare the proposed approach with related techniques, we leveraged FROUROS [166], a Python library that provides a wide variety of algorithms for drift detection. Specifically, since our approach falls into the category of data distribution-based approaches, we considered available *Data Drift* measures. Among these, we considered approaches that quantify the shift in the data and provide a value within a bounded range, as in the case of the proposed metrics. The first baseline method we considered is the Hellinger distance [97], recommended by [88] and leveraged by several approaches in the literature [65, 89]. Instead, a second baseline method we considered is *HiNormalizedComplement* [173]. Since these measures quantify drift for only a single attribute, for both we employed two aggregation strategies to obtain a single distance between two batches of data: (i) by using the average of all distances [65] and (ii) by using the maximum between all distances [151]. In what follows, for the *Hellinger* (*HiNormalizedComplement*, resp.) distance, we refer to the first strategy as He_{mean} (Hi_{mean}, resp.) and to the second strategy as He_{max} (Hi_{max}, resp.).

EXPERIMENTAL SETTINGS. The experimental evaluation has been performed in two different phases. In the first one, we considered datasets with *Known Drift* to compare the effectiveness of all the considered approaches and to determine the best metrics among the proposed ones. Thereafter, we employ the latter on datasets with *Unknown Drift* to simulate a real-world scenario. In what follows, we describe the datasets used in our evaluation (see Table 7.5), by also providing details about the evaluation process.

i) Datasets with known drift. The datasets considered in this scenario can be divided into two groups. The first one, namely *Statistical Drift*, contains 9 configuration of the *Followers* dataset obtained from a repository that allowed us to mix up normal data and data affected by gradual or abrupt drift over a variable num-

ber of columns. The second group, namely *Attribute-relationship Drift*, considers three classification datasets (i.e., *Recruitment*, *Age*, and *Forest CovType*) and their drifted version, which have been syntetically generated by shuffling the values of their columns independently (i.e., without mixing values between different columns). This ensures that the distribution of each attribute remains unchanged, while the underlying relationships between different attributes are altered. Specifically, for each dataset we shuffled the values of three columns (○), all columns except the target attribute (△), and half of the columns for one portion of data and all columns for the remaining one (◇). The latter generation method aims to simulate a more gradual drift.

ii) Datasets with unknown drift. For this scenario, we considered 5 datasets: *Bankrupt*, *Event Logs*, *Cleveland*, *KC*2, and *Heart-Statlog*. *Bankrupt* contains financial data collected from 1999 to 2009. The *Event Logs* [175] dataset contains data used for training a Meta-Learning system to recommend the best anomaly detection algorithm. It is composed of 365 instances, which comprises the 168 used by the authors and other synthetic ones. The *Cleveland* dataset contains data about heart diseases in patients, while the *KC*2 dataset contains software metrics extracted from source code to predict software defects. Finally, the *Heart-Statlog* dataset contains medical attributes related to heart health.

As shown in Table 7.5, for all dataset configurations, we randomly sampled a certain number of rows in order to vary the data within each configuration. Instead, we used all samples for smaller datasets (i.e., datasets with ID from 21 to 24). The number of classes reported in Table 7.5 refers to those appearing in the sampled data. Notice that, although our approach can be applied on any type of data, we faced the necessity of considering datasets with only numerical features due to the requirements of the baseline approaches.

iii) Evaluation Process. All the experimental sessions we performed involved a partitioning process to split datasets into four batches, corresponding to the 25%, 45%, 70%, and 100% of their size, respectively. Thus, the first batch contains tuples from 0% to 25%, the second one contains tuples from 25% to 45%, and so forth.

The first batch is also used for training a Random Forest model, which is deployed for making predictions over the other batches. For datasets with *Known Drift*, we sampled normal data for training and for the first test batch, whereas we sampled drifted data for the successive test batches. Thus, we performed the preprocessing step on the training dataset by applying mutual information-based feature selection to only consider the most relevant features (see *Sel. Features* column in Table 7.5). Then, for each class we performed an RFD_c discovery step and filtered the most meaningful dependencies. Thereafter, we incrementally added tuples for each subsequent test batch, whose target attributes' classes are those predicted by the model. Specifically, the updated dataset was given as input to the discovery module to obtain updated sets of RFD_cs. Finally, we compared the original and the updated sets of dependencies by computing the metrics described in Section 7.1.3.2.

To evaluate the effectiveness of the proposed metrics, we compared, for each class, the correlation of their trend with the actual performance trend of the model in terms of *F1-Measure*. Notice that, for confusion matrix-based metrics, the *F1-Measure* is computed and compared with the model performances. The overall correlations on all classes are compared with the ones achieved by the baseline approaches. For the metrics that express the divergence between Σ_{y_i} and Σ'_{y_i} (including the baseline approaches), we expect a negative correlation, as the model performance should decrease when divergence increases. Conversely, for the confusion matrix-based metrics, we expect a positive correlation, as higher scores denote less difference among the compared sets of RFD_cs.

Experimental Results. In what follows, we first discuss the experimental results observed on datasets with *Known Drift*, and then the ones recorded on datasets with *Unknown Drift*.

i) Datasets with known drift. Figure 7.3 shows the distribution of correlations obtained by each metric in the configurations with *Known Drift* (IDs 1-18). These have been computed as the average of the correlations over all classes. The metrics are ordered on the x-axis according to the median correlation value, starting

Dataset	Source	ID	Samp. Rows	Classes	Sel. Features	Drift
Followers	Kaggle	1	5000	2	11	Abrupt
		2	5000	2	11	Abrupt
		3	5000	2	11	Abrupt
		4	5000	2	11	Gradual
		5	5000	2	11	No Drift
		6	5000	2	11	Abrupt
		7	5000	2	11	Gradual
		8	5000	2	11	Abrupt
		9	5000	2	11	Gradual
Age	UCI ML	10	4500	2	9	●
		11	4500	2	9	▲
		12	4500	2	9	◆
Recruitment	Kaggle	13	2500	2	10	●
		14	2500	2	10	▲
		15	2500	2	10	◆
CovType	UCI ML	16	4000	2	12	●
		17	4000	2	12	▲
		18	4000	2	12	◆
Bankrupt	UCI ML	19	1000	2	9	Unknown
		20	1000	2	9	Unknown
Event Logs	[175]	21	365	3	20	Unknown
Cleveland	UCI ML	22	303	2	14	Unknown
KC2	Open ML	23	522	2	9	Unknown
Heart-Statlog	Open ML	24	270	2	10	Unknown

Table 7.5: Details of the datasets employed in the evaluation.

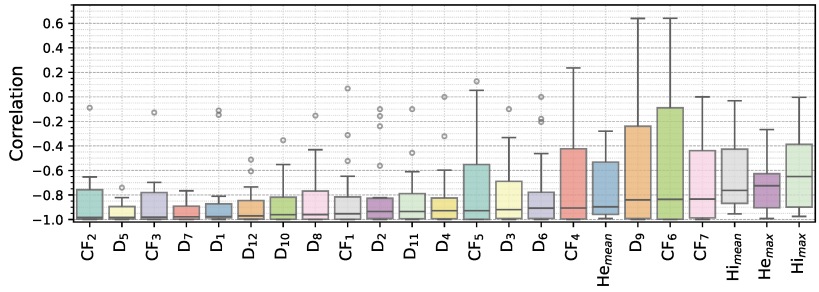


Figure 7.3: Correlations of metrics with the models' F1-Measure (the lower the better) on datasets with *Known Drift* (1-18).

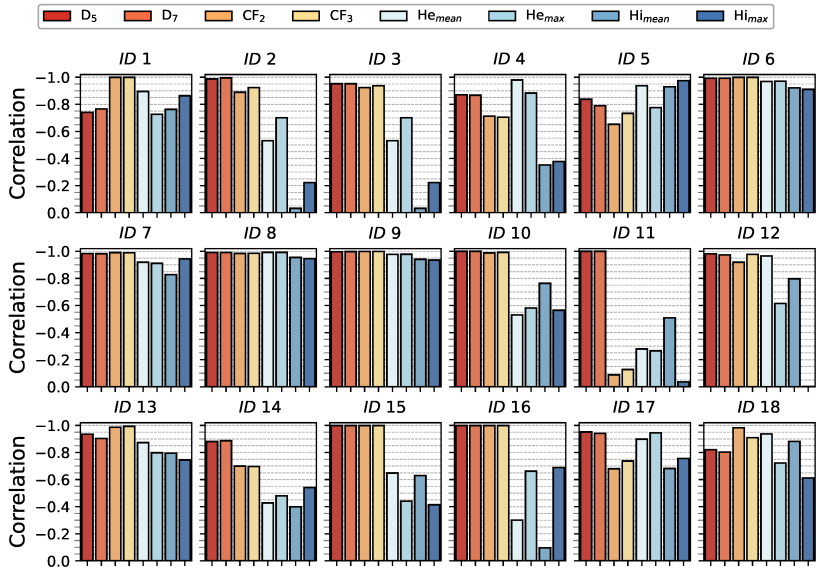


Figure 7.4: Correlations with the models' F1-Measure on datasets with *Known Drift*.

from the best towards the worst. Notice that, in order to show all metrics in a single plot, we considered the inverted correlation values for the confusion matrix-based metrics, since, as expected, they showed positive values.

In general, we can observe that most of the RFD-based metrics achieved stronger correlations than the baseline approaches. In fact, Hi_{mean}, He_{max}, and Hi_{max} present weaker correlations, while He_{mean} recorded slightly better results, outperforming three of the proposed metrics (i.e., D_9, CF_6, and CF_7). All other RFD-based

divergences and confusion matrix-based metrics showed stronger correlations, with median values better than -0.9, confirming the validity of both approaches in quantifying drift in the data.

We found that the best metrics are two divergences (i.e., D_5 and D_7) and two confusion matrix-based metrics (i.e., CF_2 and CF_3), even though the latter exhibit a larger interquartile range, indicating more variability in the results. On the other hand, the best divergences show a narrower interquartile range, suggesting that they may be more consistent and reliable. This is reflected by analyzing the average correlation obtained by these metrics: D_5 and D_7 have an average correlation of -0.94 and -0.93, respectively, while CF_3 and CF_2 have an average correlation of 0.87 and 0.86, respectively. Concerning the baseline approaches, the best metric was He_{mean}, with an average correlation of -0.75, while He_{max} achieved a slightly lower correlation (i.e., -0.73). Hi_{mean} and Hi_{max} performed significantly worse, with average correlations of -0.62 and -0.59, respectively. Thus, among the two aggregation strategies, averaging performed better than selecting the maximum distance (i.e., He_{mean} vs He_{max} and Hi_{mean} vs Hi_{max}). This could be due to the fact that the maximum aggregation strategy is likely more sensitive to outliers, overemphasizing single large shifts that may not impact the overall behavior of the model's performance.

Figure 7.4 shows in detail the results of the individual experiments for the top 4 RFD-based metrics with respect to the baseline approaches. As described before, we divided this first phase of experiments into two parts to evaluate two different drift scenarios. The first row of results (IDs 1 to 9) relates to the experiments performed on the *Followers* dataset. In this scenario, we expected the baseline approaches to perform well, since the dataset contains changes in the statistical properties of individual attributes, which should represent the type of drift the baseline approaches better detect. This expectation was confirmed by the experimental results, in which the *Hellinger* distance performed reasonably well: He_{mean} achieved an average correlation of -0.86 and He_{max} a correlation of -0.85, with both metrics showing lower correlations only for experiments with IDs 2 and 3. Despite these good results, RFD-based metrics outperformed them. In fact, D_5 was the best metric, with an average correlation of -0.927,

followed by D_7 (i.e., -0.926), CF_3 (i.e., 0.91), and CF_2 (i.e., 0.90). Instead, Hi_{mean} and Hi_{max}, recorded the worst results, with correlations of -0.71 and -0.64, respectively.

The second group of experiments (IDs 10-18) is shown in the second row of Figure 7.4. As discussed, we artificially introduced drift by shuffling the column values to alter multi-column relationships. This type of drift significantly affected the performance of baseline approaches, which were often unable to provide a correct drift estimation. He_{mean} achieved an average correlation of -0.65, while He_{max} and Hi_{mean} recorded a correlation of -0.61. Finally, Hi_{max} obtained the worst result, with an average correlation of -0.48. Thus, the trend of these distances was not aligned to the *F1-Measure* of the model. Instead, the RFD-based metrics showed the best results. In particular, D_5 and D_7 achieved an average correlation of -0.95 and -0.94, respectively; whereas CF_3 and CF_2 performed slightly worse, with correlations of 0.82 and 0.81, respectively. This decrease is mainly due to the negative results obtained in the experiment with ID 11, caused by a low correlation on one of the classes. Overall, we can conclude that although confusion matrix-based metrics are capable of obtaining significant results, they are less reliable than RFD-based divergences. Among the latter, D_5 and D_7 proved to be the most effective, consistently maintaining strong correlations across all experiments. To evaluate the generalizability of the results with respect to other ML models, we analyzed the correlations obtained using Random Forest (RF), Support Vector Classifier (SVC), Gradient Boosting Classifier (GB), and Multilayer Perceptron (MLP) on a sample of experiments. As shown in Figure 7.5, all metrics exhibit minimal variations. In fact, although the F1-Measure differs among models, the behavior of the latter is similar, i.e., they remain stable with normal data and degrade with drift-affected data.

ii) Datasets with unknown drift. Figure 7.6 shows the correlations achieved by the top 4 RFD-based metrics on datasets with *Unknown Drift* (IDs 19-24), with respect to the ones obtained by the baseline approaches. As can be seen, the RFD-based divergences D_5 and D_7 achieved the strongest correlations in almost

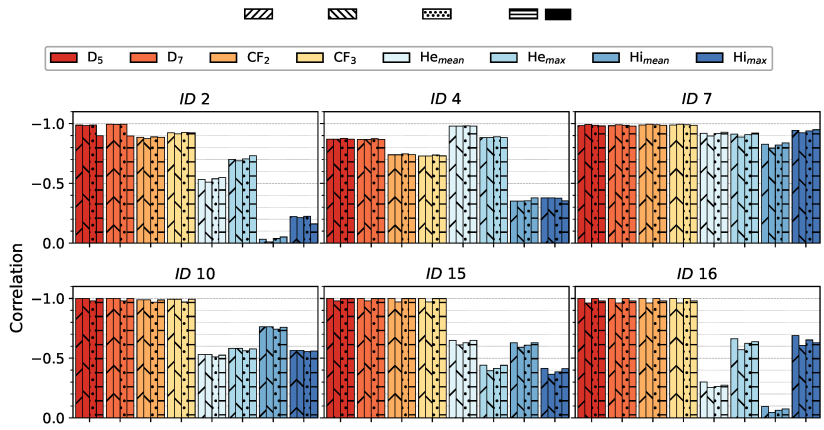


Figure 7.5: Correlations with different ML models.

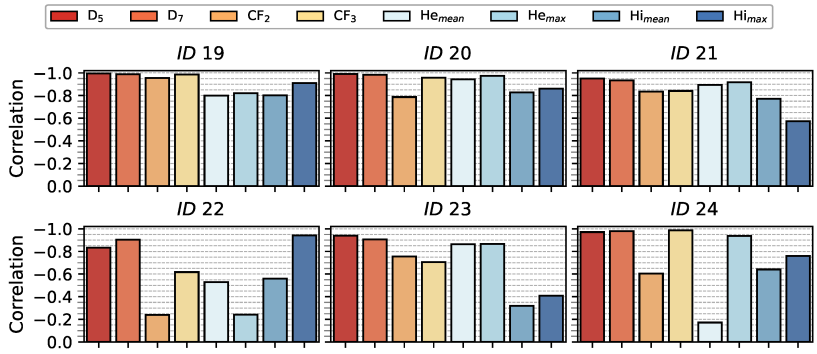


Figure 7.6: Correlations with the models' F1-Measure on datasets with *Unknown Drift*.

all experiments, confirming themselves as the best metrics we proposed, achieving an average correlation of -0.946 and -0.948, respectively. Concerning confusion matrix-based metrics, we observed a similar behavior with respect to the previous dataset configurations. In fact, although they achieved good correlations in some configurations, their performances were less consistent, leading to a lower average outcome (i.e., 0.85 for CF_3 and 0.70 for CF_2). In particular, CF_2 has been affected by poor outcomes in experiments with IDs 22 and 24.

As for the baseline approaches, the results in this scenario diverged from those observed previously. In particular, although Hellinger was still the most effective baseline metric, in this sce-

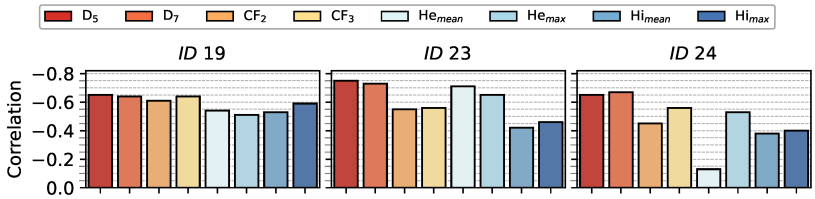


Figure 7.7: Correlations obtained by using smaller test batches.

nario, the better aggregation strategy was selecting the highest value. In fact, He_{max} and Hi_{max} achieved an average correlation of -0.79 and -0.74, respectively, while He_{mean} and Hi_{mean} recorded a correlation of -0.70 and -0.65, respectively. The better results of He_{max} and Hi_{max} might be explained by a significant distribution change of an attribute that most influenced the model behavior in some of the tested configurations. However, also in this case, D_5, D_7, and CF_3 were more accurate in quantifying drifts with respect to all baseline approaches.

As a further contribution, we investigated how correlations change when using smaller test batches. For a subset of experiments, we reorganized the test batches on the same data. Specifically, after using 25% of the data for training, we defined the first test batch as 5% of the dataset (i.e., from 25% to 30%), and the subsequent test batches as 10% each (e.g., from 30% to 40%, from 40% to 50%, and so forth). As shown in Figure 7.7, there has been a general decrease in correlations, due to the models exhibiting a more unstable trend. However, by comparing the correlation with the ones in Figure 7.6, the overall behavior of the metrics remains consistent, with D_5 and D_7 standing out as the most reliable metrics even in this scenario.

Discussion. The analyzed results showed that studying the evolution of RFDs over time can provide useful insights into whether the monitored model is providing accurate predictions or not. As for the divergence metrics, most of them showed a strong negative correlation with the performance of the model, indicating that their upward trend is very similar to the downward trend of the model. Among the proposed divergences, D_5 and D_7 were found to be the most robust and effective in quantifying drifts in the

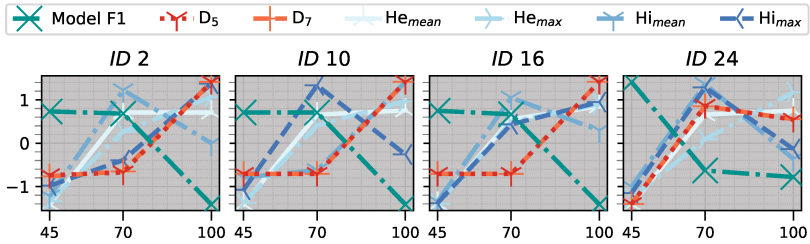


Figure 7.8: Trend analysis.

data on all the analyzed scenarios. As for confusion matrix-based metrics, most of them also exhibited a rather high correlation. This indicates that our reinterpretation of the confusion matrix to represent changes in RFDs is effective, and the trends of the *F*1-*Measure* computed on it are comparable to the ones of the monitored model. Among the proposed confusion matrix-based metrics, CF_3 and CF_2 achieved the best correlations with the models' performance in the analyzed scenarios. However, these metrics showed more instability than divergences, indicating that they need further refinements. One possible reason for this instability is that, unlike divergence metrics, confusion matrix-based metrics do not use weighting coefficients for the different ways in which RFDs can evolve. To summarize, we consider D_5 and D_7 to be the best metrics to use for quantifying concept drift, as they consistently demonstrated strong correlations with the performance of ML models. Although D_5 and D_7 differ slightly in how they evaluate specializations, their similar behavior suggests that they can be used interchangeably.

Experimental results also showed that most of the proposed metrics can provide more accurate estimates of drifts within data than other data distribution-based measures proposed in the literature. This is due to the fact that, although the latter demonstrated their capability in capturing changes in the single-column statistics of data, they did not account for changes between multi-attribute relationships. For this reason, we argue that RFD-based metrics better reflect model performance, minimizing the risk of false positives and false negatives when employed in a concept drift detection system.

To support this claim, Figure 7.8 presents illustrative examples comparing the model's behavior with the divergences D_5 and D_7, as well as with the baseline metrics. The trends shown are averaged across all classes and have been z-normalized to enable a visual comparison. As it can be seen, even though in some experiments the performance of the model sharply dropped, the baseline metrics reported only slight changes or no changes; whereas in some other cases, they even tended to follow the model performance, leading to unexpected behaviors that make such distances unable to describe the model degradation in such scenarios. Examples of this behavior that can be recognized as a false negative are observed in the third test batch of experiments with IDs 2, 10, 16, and 24. On the other hand, in some experiments baseline metrics reported distributional changes that had not affected model performances since the underlying relationships among the attributes possibly remained unchanged. Examples of this behavior that can be recognized as a false positive are observed in the second test batch of experiments with IDs 2, 10, 16. In contrast to baseline metrics, RFD-based divergences were better able to describe the trend of the model, rising in case of performance degradation and remaining stable otherwise. Thus, we can conclude that in a real-world scenario where a model is deployed and it is not possible to obtain feedback on its predictions, RFD-based metrics can better provide insights into its performance trend over time.

7.1.3.4 *Limitations*

This paper primarily focused on proposing novel RFD-based drift metrics and on empirically evaluating their behavior with respect to the actual performance trend of the model. Although exhaustive, the evaluation of the proposed approach has been only performed in a controlled environment. To this end, the proposed approach should be integrated within a standardized process for monitoring a deployed model. This will also require an evaluation based on offline metrics.

To summarize, we investigated the potential of profiling metadata to assess the evolution of data over time. Specifically, we formalized the theoretical relationship between changes in terms of RFDs and performance trends of predictive models over time. Then, we introduced two categories of RFD-based metrics to quantify drift. In particular, they are based on sets of RFD_cs discovered from data collected in different time instants. Moreover, we introduced a suite of RFD-based divergences and a set of RFD confusion matrix-based metrics inspired by well-known ML measures. To evaluate the proposed metrics, we considered several datasets with both known and unknown drift, by also comparing them with other distribution-based measures. Results showed that the trend of RFD-based metrics is strongly correlated with the *F1-Measure* of the model, and that they provide more reliable insights than the compared baseline metrics, especially in more complex cases in which drift affects relationships between attributes. Drawing from empirical evidence, we established D_5 and D_7 as the most effective RFD-based metrics for accurately assessing drift.

7.2 QUERY RELAXATION

Retrieving the desired results is not a trivial task. More often than could be imagined, the query retrieves just a few results or, in some cases, nothing. This problem can have a significant impact on data analysis, as the lack of results or a limited number of them can restrict the ability to draw meaningful conclusions or obtain comprehensive and representative insights from the available data. In these cases, the user should manually modify the query parameters until he/she gets the desired result. Clearly, this process is time-consuming and can lead to an unpleasant user experience. For this reason, query relaxation techniques aim to automatically relax the query parameters in order to enlarge the query result. In what follows, we provide an overview of existing approaches (Section 7.2.1) and we formalize the Query Relaxation problem (Section 7.2.2). Then, we present a novel RFD-based query relaxation tool (Section 7.2.3)

7.2.1 *Literature Review*

In recent years, many proposals have explored the problem of query relaxation from a methodological point of view. Some of these works are based on the generation of special statistical summaries to support the rewriting of the user's query. For example, Poosala & Ganti [149] presented an approach called AQUA (Approximate Query Answering) that uses small amounts of precomputed summary statistics of the data to answer queries quickly, albeit approximately. The authors hypothesize that many OLAP applications can tolerate approximations in query results in exchange for significant reductions in response time. In AQUA, the data cube, which represents a hierarchy of multi-dimensional cubes, is approximated using multi-dimensional histograms. Any aggregate query that can be answered from the original data cube is then answered approximately using the approximate data cube. The approach also provides error estimates in the resulting answer. While this approach leverages a simple type of metadata, other proposals rely on dependencies discovered from the dataset. For instance, in [47] the authors propose an approach for rewriting failing queries that are in the disjunctive normal form and involve both discrete and continuous attributes. The approach relies on the discovery of RFDs from the dataset, which are then ranked based on a utility strategy that allows for predicting which dependency is most suited for the current query. To rewrite a failing query, the approach considers the attributes that are involved both in the query and in the selected RFD, and then it leverages the RFD thresholds to relax the query constraint. This methodology will be described in detail in section 7.2.2, as it represents the basis of the work presented in Section 7.2.3. Another dependency-based approach is the one proposed in [137]. The authors highlight how randomly relaxing query constraints could lead to undesirable results. Their method is based on the insight that the results closest to those sought by the user in the query are those that differ only on the least important attributes. For this reason, the first attribute that should be relaxed must be the least relevant. To identify these attributes, the authors rely on AFDs. In particular, these dependencies are used to compute influence weights, which describe the influence of an attribute

over the other attributes. Therefore, the attributes with the lowest influence should be relaxed first in order to reduce the impact of the relaxation process on the similarity of the results.

A different category of query relaxation methods comprises machine-learning-based approaches. In [135] the authors introduced an algorithm called LOQR (Learning for Online Query Relaxation) which, like the approach in [47] considers queries in disjunctive form that involve both discrete and continuous attributes. LOQR works by learning decision rules from a small subset of the target database and uses this knowledge to relax the constraints in the failing query. The algorithm consists of three steps: learning the decision rules, finding the most similar rule to the failing query, and using this rule to relax the constraints. The experiments conducted on six application domains showed that LOQR successfully relaxes more than 95% of the failing queries and is fast, processing queries with up to 20 attributes in under a second, making the approach feasible for real-time scenarios. This method has been subsequently revisited in [136], as the previous version tended to generate queries that were too simple when the algorithm was trained on small datasets. In contrast to the previous method, which uses learned decision rules that are then converted into queries, this new approach starts with an empty query and tries to add as many as possible of the user's query's original constraints (or a relaxation of them). The order in which the constraints are considered is derived from the domain's causal structure, which is learned from a small sample of the dataset.

More recent works explored the relaxation of queries on spatiotemporal data and studied the application of query relaxation for similarity search in RFD nodes. In [15] the authors calculate attribute weights for the query attributes based on their importance in the query. The weights are calculated using a fuzzy entropy-based method that considers the degree of uncertainty and importance of each attribute. Subsequently, a relaxation range is chosen based on the desired level of relaxation. This relaxation range is utilized to ease the constraints on the initial query graph, creating the relaxation query model. This model is then executed on the data graph, and the query results are ranked according to their relevance to the initial query graph. A

relevance score is employed, which factors in the degree of relaxation, attribute weights, and the similarity between the query results and the initial query graph. In [76] the authors propose a query relaxation approach for similarity search based on RDF nodes. They suggest starting the relaxation considering a query node that is overly specific and gradually relaxing it to find similar nodes as approximate answers. The relaxation of the query is done by applying transformations to the query elements, such as removing a query element or replacing a class with a superclass or a node with a variable. The authors introduce two algorithms, called "answers partitioning" and "lazy join", to improve the efficiency of query relaxation. The answers partitioning algorithm partitions the set of possible answers into clusters, where each cluster is defined by a relaxed query and the subset of answers that are not answers of other clusters. This algorithm allows for a more fine-grained relaxation of the query and helps to prune the space of relaxed queries. The lazy join algorithm optimizes the computation of the match-set, which is the set of all matchings of the query pattern on the RDF graph. Instead of computing one large global join, the match-set is represented by a structure called a match-tree, which contains several small local joins. This optimization significantly improves the efficiency of the query relaxation process.

Finally, Li et al. [119] addressed the problem of query relaxation in the context of semantic association (SA) search within a large entity-relation graph. SA search involves finding direct and indirect relationships between a set of entities specified in a query, and the goal is to return a compact connected subgraph containing all the query entities. However, existing efficient algorithms for SA search may fail to return results when some query entities are distant in the graph. The proposed approach focuses on removing the smallest number of entities from the original failing query to create a maximum successful sub-query. The objective is to minimize the loss of result quality caused by relaxation while ensuring that the largest proportion of query entities can still be connected in the relaxed SA. To achieve this goal, the proposed approach introduces the concept of a "certificate" for the success of a sub-query. A certificate is an entity that satisfies a distance-based condition concerning the query entities.

The core idea is to efficiently find such a certificate to verify the success of a maximum sub-query. The approach employs a best-first search algorithm that leverages distance-based estimation to effectively prune the search space, improving search efficiency. Additionally, the proposed approach enhances its performance by incorporating two fine-grained heuristics. One heuristic is based on the degree of entities in the graph, while the other is based on distance considerations. These heuristics help guide the search algorithm to find an optimal solution more quickly.

While the previous works focus only on the introduction of new methods to relax queries, other proposals focused in proposals that involve the user in an interactive relaxation process, leading to the definition of semi-automatic tools. Among these, Heer et al. [95] proposed an innovative query relaxation approach relying on the design of a proper graphical user interface that allows the user to perform selections. Their approach allows users to specify selections based on attributes, regions, or specific criteria. Furthermore, it introduces a query relaxation engine that enables users to interactively generalize their selections. This means users can expand their initial selections based on certain criteria. The query relaxation engine automates the process by analyzing attributes and network topology, making generalized selections over static and dynamic data feasible. More recently, Albarrak et al. [6] introduced ORange, a system prototype designed for objective-aware range query refinement. The primary objective of ORange is to fine-tune a range query to align with a predefined cardinality constraint while considering the (dis)similarity between the original query and its refined counterpart. The central concept driving this approach is the novel scheme known as SAQR, which stands for Similarity-Aware Query Refinement, which allows for efficient query refinement by focusing on both cardinality and similarity constraints. The authors' proposal also implements a relaxation functionality by allowing users to specify a higher cardinality constraint for the refined query. ORange ensures that the refined range query aligns closely with the user's desired cardinality while maximizing its similarity to the initial query. This combination of objectives enhances the precision and relevance of the refined query. Another interactive

approach is the one proposed in [133], in which the authors present the IQR (Interactive Query Relaxation) system. In situations where a query yields no results, IQR suggests relaxing one of the original query conditions at a time to the user. The user has the option to either accept or reject each relaxation suggestion. This process continues until either a non-empty answer is found or it is determined that obtaining a non-empty answer is not possible. The relaxation suggestions provided by IQR are guided by specific optimization objectives, which may include minimizing user interactions or maximizing profit. In particular, IQR employs a probabilistic framework to assess the likelihood of the user accepting a relaxation suggestion and the effectiveness of the relaxation in achieving the chosen optimization objective. These objectives can be user-centered, such as maximizing result relevance or minimizing the number of interactions, as well as provider-centric, such as maximizing profit.

7.2.2 *Problem Overview*

A query relaxation process requires the modification of some of the constraints of a query with the likelihood that the database will contain more tuples satisfying them than those resulting from the original query. More formally, a query relaxation process can be defined as follows:

Definition 24 (Query relaxation process). *Given a relation instance r of a relation R, and a query Q whose answer set $Q(r)$, $|Q(r)| \leq |r|$ is derived through a set of query conditions $\Gamma = \{\gamma_{B_1}, \ldots, \gamma_{B_k}\}$ such that γ_{B_i} has the form $(B_i = b_i)$, a query relaxation process aims to find a query Q' whose answer set $Q'(r)$, $|Q(r)| \leq |Q'(r)| \leq |r|$ is derived through a novel set of query conditions Γ', such that each tuple $t_j \in Q'(r)$ either satisfies conditions in Γ (i.e., $t_j \in Q(r)$), or satisfies conditions related to those in Γ. To this end, we propose a query relaxation methodology that exploits the concept of RFD$_c$ in order to derive the set of related conditions Γ'.*

RFD$_c$s can be leveraged for relaxing queries, according to the methodology defined in [47]. According to it, it is possible to leverage any RFD$_c$ φ involving at least one attribute included in the conditions defined in the original query Q to automatically

define Q'. More specifically, RFD$_c$s can be exploited to define two different query relaxation strategies, which permit to (*i*) extend query conditions by including similar values, and (*ii*) change query conditions by considering some others defined on alternative attributes, but related to those involved in Q. These two relaxation strategies are formalized in the following definitions.

Definition 25 (Extending query conditions through RFD$_c$s.). *Given a query Q and an* RFD$_c$ $\varphi : X_{\Phi_1} \xrightarrow{\Psi_{err}(0)} A_{\phi_2}$*, for each $B \in X$ if $\gamma_B = (B = b) \in \Gamma$ (i.e., γ_B is one of the conditions in Q), then we might rewrite Q in Q' by replacing γ_B with $\gamma'_B = (b'$ like $b)$, where like means that either $b - \varepsilon \leq b' \leq b + \varepsilon$ (if B is a numerical attribute), with ε derived from the* RFD$_c$ *threshold on B, or b' IN $\{b_1, \ldots, b_l\}$ (if B is a textual attribute), where $\{b_1, \ldots, b_l\}$ represent a collection of values similar to b according to the similarity/distance constraint associated to B in φ.*

Definition 26 (Changing query conditions through RFD$_c$s.). *Given an* RFD$_c$ $\varphi : X_{\Phi_1} \xrightarrow{\Psi_{err}(0)} A_{\phi_2}$ *and the answer set $Q'(r)$ obtained from the previous query extension strategy, then we might rewrite Q' in Q'' by replacing the condition γ'_B from Γ' according to the following formula:*

$$\gamma''_B = \bigvee_{t \in Q'(r)} \left(\bigwedge_{C \in X \cup A \,\wedge\, \gamma_C \notin \Gamma} c'' \text{ like } t[C] \right)$$

where ε is derived from the RFD$_c$ *threshold on C, and $\{c_1, \ldots, c_l\}$ represents a collection of values similar to $t[C]$ according to the similarity/distance constraint associated to C in φ.*

In other words, given the selected RFD$_c$ φ, γ''_B rewrites γ'_B by considering each tuple t in $Q'(r)$ (i.e., t satisfies γ'_B), and by considering each other attribute C involved in φ from which it is possible to define a new condition according to the similarity/distance constraint associated to C and the value $t[C]$. Notice that, no conditions are included for the attributes C involved in the original query (i.e., $\gamma_C \in \Gamma$), since these attributes are already involved in the query relaxation process started from γ_C. Thus, γ''_B represents a disjunction of new conditions (i.e., one for each tuple in $Q'(r)$), each rewriting γ'_B with a conjunction of new conditions over alternative attributes included in φ but not in Q. (i.e.,

one for each attribute C in $X \cup A$ not included in any condition of the original query Q).

The number of RFD$_c$s holding on a given dataset can be high, even considering only the set of minimal RFD$_c$s. For this reason, it is important to have a strategy for identifying the most useful ones holding on a relation r. According to the methodology in [47], it is possible to rank the collection of RFD$_c$s according to the following criteria that maximize the original semantics of the query: (*i*) minimizing the LHS cardinality, (*ii*) maximizing thresholds for the LHS attributes, (*iii*) minimizing the threshold for the RHS attribute [45], and (*iv*) maximizing the number of attributes involved in the original query conditions. Moreover, a filtering strategy imposes the removal of all the RFD$_c$s that do not include at least one attribute involved in the conditions defined over the original query Q.

7.2.3 *Required [39]*

One of the current research gaps in the context of Query Relaxation lies in the need for interactive tools that foster user engagement and empowerment, actively involving users in relaxing their queries and bridging the semantic gap between their intent and the data they seek. These tools should have, at the same time, a high level of automation in order to provide a satisfying user experience. To address this challenge, we introduced REQUIRED, and RFD$_c$-based query relaxation tool that implements the methodology defined in [47] and formalized in Section 7.2.2 and integrates it in an interactive system. The following sections will detail the architecture of REQUIRED and its functionalities.

7.2.3.1 *System Architecture*

REQUIRED relies on a microservice architecture in which data is exchanged through REST APIs by means of the JSON standard format in order to guarantee high portability, scalability, and maintainability of components. Although the tool is developed as a WebApp, the Back-end component of REQUIRED can be integrated with a plugin into other systems, such as DBMS clients.

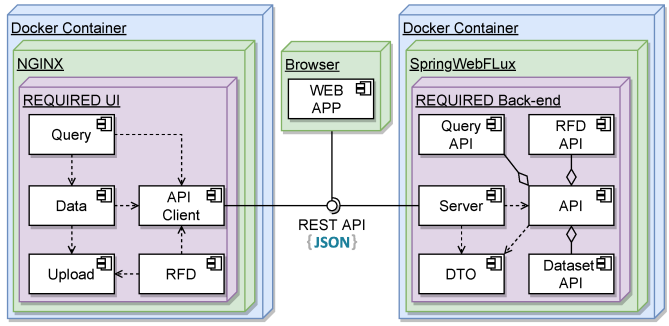


Figure 7.9: Overview of REQUIRED's architecture.

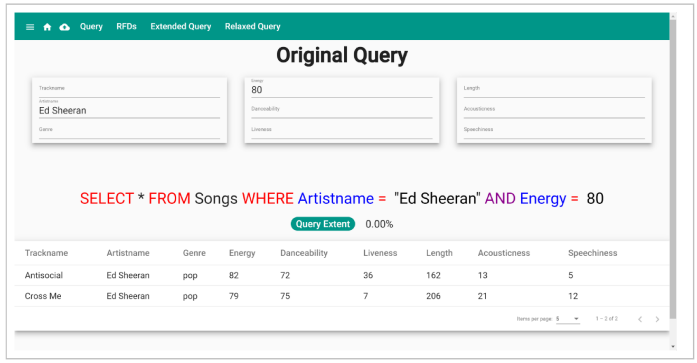


Figure 7.10: Interface for performing a query on a dataset.

Figure 7.9 shows an overview of the system architecture. As we can see, REQUIRED is separated into two containers: one including the user interfaces (REQUIRED *UI*) and the other containing the services required for executing the system functionalities (REQUIRED Back-end). The components of the user interfaces access the REQUIRED services in the back-end through the *API Client* component. The latter interacts with the REQUIRED services in the other container by invoking the *Server* component, which is responsible for communicating with the other services in the container and for executing their functionalities.

The user interacts with REQUIRED through a web browser through which s/he can upload a dataset and submit queries. Then, the user can upload the RFD_cs holding on the dataset, so that s/he can select the one best suited for relaxing the query.

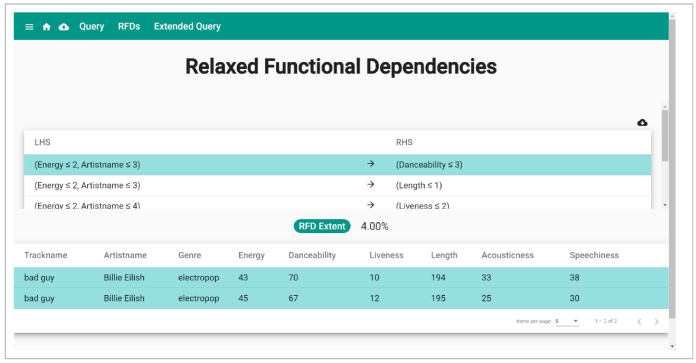


Figure 7.11: Interface for the selection of an RFD.

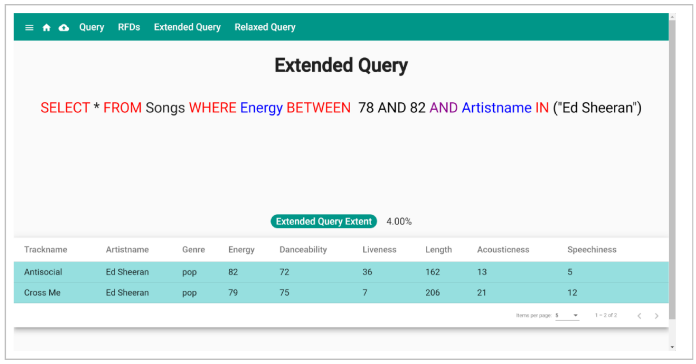


Figure 7.12: Query extension interface.

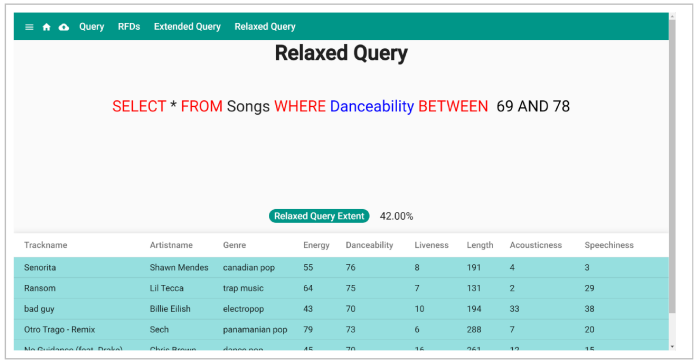


Figure 7.13: Query relaxation interface.

7.2.3.2 *Implementation details*

The tool has been developed as an application divided into multiple Docker containers. The Client components have been designed as a monorepo architecture using Angular and NX frameworks, which permit defining reactive interfaces and facilitate future updating of the functionalities behind REQUIRED. The Server is a reactive RESTFul web service that has been built as a Maven multi-module project using Java and SpringWebFlux. Through Docker, both the SpringWebFLux microservice and the Angular UI application have been containerized in two Docker Images freely accessible on Docker Hub[1].

7.2.3.3 *Functionalities*

REQUIRED is able to provide users with several functionalities, requiring a minimum number of configurations for their execution. After uploading a dataset, REQUIRED automatically identifies its attributes and their types, i.e., numerical or textual, and it generates input fields with which the user can interact (Figure 7.10). Starting from this, REQUIRED provides three main functionalities:

- **Executing a query**: We can execute a query by instantiating the attributes recalled in its conditions. REQUIRED automatically composes and enquires the dataset, and shows the answer set within the table at the bottom of the interface. Moreover, the percentage value at the top of the table shows the rate by which occurrences contained in the dataset match the executed query. Figure 7.10 shows an example of the query performed on the *Songs*[2] dataset. As we can see, we have set the values "Ed Sheeran" and "80" for the attributes Artistname and Energy, respectively. Consequently, REQUIRED enquires the dataset by executing the following query: "SELECT * FROM Songs WHERE Artistname = 'Ed Sheeran' AND Energy = 80".

- **Extending a query**: After executing a query, we can upload a set of RFD_cs holding on the considered dataset. Then,

1 https://hub.docker.com/u/rfdtoolbox/

2 Spotify's Top 50 Songs in 2019 available on Kaggle

REQUIRED shows in a table, those automatically ranks and, and shows the RFD_cs in a table, after applying the ranking and filtering strategy (Figure 7.11). After that, we can select an RFD_c, and REQUIRED automatically extends the query results according to the methodology described in Section 7.2.2 [47]. Figure 7.12 shows an example of the extended query after selecting the RFD_c:

$$\varphi : \mathsf{Energy}_{(\leq 2)}, \mathsf{Artistname}_{(\leq 3)} \xrightarrow{\Psi_{err}(0)} \mathsf{Danceability}_{(\leq 3)}$$

leading to the considerations of songs with similar Energy i.e., in the range $[78, 82]$, and with similar Artistname, i.e., the ones that have a Levenshtein distance ≤ 3 with respect to 'Ed Sheeran'. As we can see, REQUIRED updates the previous query by extending its conditions and rewrites it in the following query: "SELECT * FROM Songs WHERE Energy BETWEEN 78 AND 82 AND Artistname IN ('Ed Sheeran')". The extended query led to an increase in the matching occurrences, reaching a rate of 4% with respect to the whole dataset.

- **Relaxing a query**: After extending a query, we can relax some constraints according to the methodology described in Section 7.2.2 [47]. In particular, after selecting the "Relaxed Query" button, REQUIRED automatically re-defines some constraints of the extended query by considering the RFD_c selected during the previous step. Figure 7.13 shows an example of the relaxed query considering the selected RFD_c φ, which has led to the definition of the new relaxed query: "SELECT * FROM Songs WHERE Danceability BETWEEN 69 AND 78". As we can see, the resulting occurrences have significantly increased with respect to those achieved with the original query, reaching a matching rate of 42% with respect to the whole dataset. Notice that, if the query in the previous step already provides more than 10 tuples in the results, REQUIRED triggers a prompt warning to notify the user with a possible over-relaxation of the query's results in the application of this relaxation step.

7.2.3.4 *Limitations*

In its current version, REQUIRED supports the relaxation of queries that include basic SQL operators and a single table. Its functionalities might be extended by introducing support for range queries, join queries, and by tailoring the relaxation process for additional data types.

To summarize, we proposed REQUIRED, a query relaxation tool that leverages Relaxed Functional Dependencies. Through its user interface, REQUIRED offers the possibility to upload a dataset and execute queries. To enlarge the result set, the user can load a set of RFD$_c$s, which are automatically ranked by the tool according to a utility strategy. After selecting an RFD$_c$, REQUIRED can extend the query conditions or even completely rewrite some conditions of the user's query.

SUMMARY OF PART III

In this part of the thesis, we investigated further application of Data Profiling. In particular, we first focused on the problem of Concept Drift Detection, providing a formalization of drift and a review of existing detection approaches. Then, we introduced a novel concept drift detection approach based on the analysis of the evolution of RFDs over time. Through an experimental evaluation, we showed that the proposed approach outperforms existing data distribution-based detection methods, which analyze single-attribute distribution shifts rather than evaluating multi-attribute relationships. Then, we focused on Query Relaxation, providing a formalization of the problem and explaining how RFDs can be leveraged in this scenario. We provided a review of existing methods, and we proposed REQUIRED, an interactive Query Relaxation tool based on RFDs. Overall, the contributions presented in this part of the thesis show the potential of Data Profiling in supporting different tasks, and pave the way for future, metadata-driven, research directions.

8

CONCLUSION AND FUTURE DIRECTIONS

This Chapter provides closing remarks and discusses future research directions that can be undertaken starting from the proposed works.

In this thesis, we have explored the potential of metadata-driven approaches in supporting different tasks that are increasingly gaining interest in many sectors, including the one related to public administration. More specifically, we focused on the application of data dependencies to support *Data Imputation*, *Concept Drift* detection, and *Query Relaxation*. Moreover, we also introduced advancements of *Data Profiling*, aiming to improve the overall applicability and interpretability of metadata.

In the **first part of the thesis**, we provided an overview of Data Profiling tasks and applications, highlighting the importance of these techniques in extracting compact and meaningful knowledge from data. We focused on data dependencies, which have been increasingly relied upon for several tasks in recent years. Then, we presented some of the challenges related to this field, highlighting the lack of semantic-aware dependency discovery algorithms and tools capable of visualizing and effectively explaining large sets of evolving dependencies. In relation to the former challenge, we presented S-DiMε [42], an RFD discovery algorithm that enables the use of semantic-aware distance functions to properly assess similarities when dealing with long textual attributes. Then, we presented RYAN [41], a tool that enables the exploration and visualization of RFDs discovered at different time instants, allowing for the analysis of their evolution. Moreover, RYAN integrates an LLM interaction module that enables providing natural language explanations of the reasons why a particular RFD has evolved. Table 8.1 summarizes the main findings and contributions of the proposed works in this part.

Contribution	Ref.	Main Findings/Contributions
S-DIMε	[42]	S-DIMε is the first RFD discovery algorithm that leverages semantic distances in the discovery process. Compared with a traditional discovery algorithm, i.e., DIMε, S-DIMε enabled the discovery of novel meaningful RFDs (7.30%-39.53%). Moreover, S-DIMε outperforms DIMε in terms of execution times (21.16% – 98.42%) and memory consumption (5.35% – 7.29%).
RYAN	[41]	RYAN is the first RFD visualization tool that offers the possibility to analyze metadata evolution and to provide natural language explanations on why and how RFDs evolve by leveraging LLMs. Through an user study, we show that RYAN is able to provide effective visualizations of RFDs and to deliver detailed and correct explanations.

Table 8.1: Summary of the papers presented in this thesis to introduce advancements in Data Profiling, along with the key findings/contributions of each work.

In the **second part of the thesis**, we focused on the problem of Data Imputation, which stands out as one of the most critical aspects in any data-driven task. We first provided an overview of existing techniques and highlighted some of the open challenges related to the application of metadata-driven approaches in the scenario of time series data and in dynamic contexts. For the former challenge, we presented RIFT [172], an RFD-based imputation framework that introduces several optimizations tailored for multivariate time series, bringing significant improvements in the efficiency of RFD-based imputation approaches and enabling their use in this scenario. Then, we presented an empirical study [26] that investigated the cooperation between Data Profiling and metadata-driven Data Imputation tasks in incremental scenarios, highlighting the mutual benefits that these tasks can provide to each other. Table 8.2 summarizes the main findings and contributions of the proposed works in this part.

Finally, in the **third part of the thesis**, we focused on other use cases of Data Profiling. In particular, we presented a novel RFD-based approach for detecting concept drift in ML pipelines [44], highlighting how the evolution of metadata is strictly correlated to the performance of ML models and showing that studying the evolution of RFDs provides more accurate drift estimations

Contribution	Ref.	Main Findings/Contributions
RIFT	[172]	RIFT is the first RFD-based imputation framework tailored for multivariate time series data. It introduces several optimizations in how RFDs are discovered and employed during the discovery process. Experimental results showed that RIFT improves the execution time of traditional RFD-based approaches (up to 98.92% for RFD discovery and up to 65.64% for imputation) and that it represents a useful preprocessing step to inject high quality values to support further imputation steps, significantly improving *Precision*, *Recall*, and *RMSE* (up to 29.38%).
Incremental Imputation Study	[26]	This empirical study investigated the mutual benefits between incremental RFD discovery and RFD-based imputation. Through the definition of two pipelines, we evaluated to what extent the continuous update of RFDs can enhance the imputation quality, and to what extent imputation can mitigate the effects of missing values in discovery processes. Results showed that updating RFDs significantly reduce *RMSE* (33.15% on average) and that re-evaluating previous imputations based on outdated metadata further reduces errors (16.4% on average). Moreover, reducing missing values through imputation enables to obtain a set of RFDs closer to the ideal set (27.16%on average).

Table 8.2: Summary of the papers presented in this thesis to introduce advancements in Data Imputation, along with the key findings/contributions of each work.

with respect to studying simpler statistics. Then, we presented an interactive RFD-based Query Relaxation tool [39] that enables users to interactively relax query constraints and consequently enlarge the queries' result set with elements correlated to the original user intent. Table 8.3 summarizes the main findings and contributions of the proposed works in this part.

Together, the proposed approaches contribute to bridging the gap between theoretical advancements in data profiling and their operational use in real-world scenarios, tackling problems that are becoming increasingly relevant in the digitalized public sector scenario. Beyond their technical contributions, the approaches presented in this thesis also open promising directions in the broader context of data governance and responsible Artificial Intelligence (AI). The ability to extract, visualize, monitor, and explain metadata provides insights into how data evolve and models behave over time. This can support governance frameworks by

Contribution	Ref.	Main Findings/Contributions
RFD-based Concept Drift Detection	[44]	This work proposed the first RFD-based approach to quantify concept drift in ML Settings. It proposes an approach to identify a set of representative RFDs for each class of the training set, and it introduces two suites of RFD-based metrics to quantify metadata evolution. Experimental results showed that the trend of the proposed metrics is strictly correlated to the actual performance trend of the monitored model. Moreover, the proposed approach outperformed existing data distribution-based approaches when considering different drift types, especially in scenarios in which drifts affect multi-attribute relationships.
REQUIRED	[42]	REQUIRED is the first interactive query relaxation tool that involve the user in the relaxation process. It enables users to perform queries and visualize the results, and leverage RFDs to implement two different ways of relaxing queries, either by modifying the existing query conditions or totally rewriting them by considering different attributes.

Table 8.3: Summary of the papers presented in this thesis to introduce novel applications of RFDs, along with the key findings/contributions of each work.

offering interpretable evidence of data quality, model shifts, and potential biases, which is increasingly relevant for organizations that must ensure transparency, traceability, and accountability in their data-driven processes. Moreover, the proposed imputation approaches directly support data quality, which is one of the core principles of preprocessing steps and data governance processes in ML settings. Finally, the query relaxation approach contributes to data accessibility and controlled information retrieval, both of which are increasingly relevant in responsible AI practices.

Concerning the future perspectives, the works proposed in this thesis pave the way for several research directions that can be pursued. Starting from the S-DIMε algorithm, possible improvements include the implementation of a methodology to automatically determine which distance measure should be applied according to the nature of each attribute. Moreover, the definition of the comparison thresholds still requires further investigations, since defining the latter when dealing with attributes whose values are compared by considering their embeddings is not a straightforward task. As concerns the RYAN explainability tool, future work includes the extension of the platform to support different types

of metadata and to include additional visualization methods, as well as new interactive components that dynamically update with data changes. We also plan to enhance explainability by aggregating insights from multiple LLMs and developing tailored prompts for other metadata. Moreover, we aim to evaluate RYAN with a broader range of users, and to measure task-related metrics such as the success rate in the assigned tasks.

Considering the RIFT framework, while in our work we showed that, even when providing a reduced number of imputations, we were able to significantly improve the performances of subsequent imputation approaches, we aim to develop a stand-alone RFD-based approach. This will require either modifying the logic of RFD-based imputation algorithms or considering entire subsequences as imputation candidates. We also aim to further improve the scalability of RFD-based imputation by introducing new optimizations and distributed architectures. Concerning the empirical study on the cooperation between Data Profiling and Data Imputation in incremental scenarios, we would like to explore the design of an iterative pipeline enabling mutual refinement between imputation and RFD_c discovery, and to leverage adaptive feedback to improve the quality of both tasks. Additionally, we aim to address the challenges posed by concept drift and evolving data distributions by developing methods for automated change detection and selective updates of RFD_cs and imputed values. Finally, while in the presented study we simulated the incremental scenario by re-executing the imputation process across different batches, its outcomes emphasized the importance of introducing incremental metadata-driven imputation approaches, as well as the introduction of more refined approaches to reconsider imputations made on the basis of outdated metadata.

Considering the RFD-based concept drift detection approaches, in the future, we would like to investigate other types of profiling metadata to define a complete framework of drift metrics, which holistically consider statistics, patterns, and properties among data to monitor model performances. Furthermore, we plan to refine the proposed metrics, with a particular focus on confusion matrix-based ones, which showed promising results but were not as consistent as the divergence metrics. Moreover, while in this work we leveraged a static discovery algorithm, future works

should employ incremental discovery strategies [28, 35, 40] to update RFD$_c$s over time without reconsidering already processed data to keep RFD$_c$s updated while reducing discovery times. This requires defining incremental RFD discovery algorithms capable of also inferring similarity/distance thresholds. Finally, while we showed that the evolution of RFDs is strictly correlated to the performance of ML models, it is necessary to implement a proper concept drift alerting system, which will require the definition of appropriate thresholds for triggering warnings, as done in other data-distribution-based approaches. Finally, as concerns the REQUIRED tool, we aim to extend the support of RFD-based query relaxation to other SQL operators and also admit the consideration of multiple tables, join queries, and the integrity constraint compliance.

To summarize, the results observed in the proposed contributions pave the way for future research directions, with the aim of developing effective metadata-centric paradigms that underlie data-driven tasks.

BIBLIOGRAPHY

[1] Ziawasch Abedjan, Lukasz Golab, and Felix Naumann. "Data Profiling: A Tutorial." In: *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017*. ACM, 2017, pp. 1747–1751.

[2] Ziawasch Abedjan, Lukasz Golab, Felix Naumann, and Thorsten Papenbrock. *Data Profiling*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2018. ISBN: 978-3-031-00737-8.

[3] Ziawasch Abedjan, Patrick Schulze, and Felix Naumann. "DFD: Efficient Functional Dependency Discovery." In: *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*. ACM, 2014, pp. 949–958.

[4] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of databases*. Vol. 8. Addison-Wesley Reading, 1995.

[5] Mahmoud Abo Khamis, Hung Q Ngo, and Dan Suciu. "Computing join queries with functional dependencies." In: *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 2016, pp. 327–342.

[6] Abdullah M. Albarrak, Tatiana Noboa, Hina A. Khan, Mohamed A. Sharaf, Xiaofang Zhou, and Shazia Sadiq. "ORange: Objective-Aware Range Query Refinement." In: *IEEE 15th International Conference on Mobile Data Management, MDM 2014, Brisbane, Australia, July 14-18, 2014 - Volume 1*. IEEE Computer Society, 2014, pp. 333–336.

[7] Waqar Ali, Wenhong Tian, Salah Ud Din, Desire Iradukunda, and Abdullah Aman Khan. "Classical and modern face recognition approaches: a complete review." In: *Multimedia Tools and Applications* 80.3 (2021), pp. 4825–4880.

[8] Robert Anderson, Yun Sing Koh, and Gillian Dobbie. "Predicting concept drift in data streams using metadata clustering." In: *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE. Rio de Janeiro, Brazil: IEEE, 2018, pp. 1–8.

[9] Marcelo Arenas and Leonid Libkin. "A normal form for XML documents." In: *Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. PODS '02. New York, NY, USA: Association for Computing Machinery, 2002, 85–96. ISBN: 1581135076.

[10] Olawale F Ayilara, Lisa Zhang, Tolulope T Sajobi, Richard Sawatzky, Eric Bohm, and Lisa M Lix. "Impact of missing data on bias and precision when estimating change in patient-reported outcomes from a clinical registry." In: *Health and quality of life outcomes* 17 (2019), pp. 1–9.

[11] Irham Azizan, Samsul Ariffin Bin Abdul Karim, and S Suresh Kumar Raju. "Fitting rainfall data by using cubic spline interpolation." In: *MATEC Web of Conferences*. Vol. 225. 2018.

[12] Fabio Azzalini, Chiara Criscuolo, and Letizia Tanca. "FAIR-DB: FunctionAl DependencIes to discoveR Data Bias." In: *EDBT/ICDT Workshops*. 2021.

[13] Manuel Baena-Garcıa, José del Campo-Ávila, Raul Fidalgo, Albert Bifet, Ricard Gavalda, and Rafael Morales-Bueno. "Early drift detection method." In: *Fourth international workshop on knowledge discovery from data streams*. Vol. 6. Citeseer. 2006, pp. 77–86.

[14] Alexei Baevski, Wei-Ning Hsu, Alexis Conneau, and Michael Auli. "Unsupervised Speech Recognition." In: *Proceedings of the Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems NeurIPS 2021*. virtual: 2021, pp. 27826–27839.

[15] Luyi Bai, Xiaofeng Di, and Lin Zhu. "Query relaxation of fuzzy spatiotemporal RDF data." In: *Appl. Intell.* 52.11 (2022), pp. 13195–13213.

[16] Abdelghani Bakhtouchi, Ladjel Bellatreche, and Yamine Ait-Ameur. "Ontologies and functional dependencies for data integration and reconciliation." In: *International Conference on Conceptual Modeling*. Springer. 2011, pp. 98–107.

[17] Parikshit Bansal, Prathamesh Deshpande, and Sunita Sarawagi. "Missing value imputation on multidimensional time series." In: *arXiv preprint* (2021).

[18] Roberto Souto Maior de Barros, Danilo Rafael de Lima Cabral, Paulo Mauricio Gonçalves Jr., and Silas Garrido Teixeira de Carvalho Santos. "RDDM: Reactive drift detection method." In: *Expert Syst. Appl.* 90 (2017), pp. 344–355.

[19] Gustavo Batista and Maria Carolina Monard. "An analysis of four missing data treatment methods for supervised learning." In: *Applied Artificial Intelligence* 17.5-6 (2003), pp. 519–533.

[20] Firas Bayram, Bestoun S. Ahmed, and Andreas Kassler. "From concept drift to model degradation: An overview on performance-aware drift detectors." In: *Knowl. Based Syst.* 245 (2022), p. 108632.

[21] Khalid Belhajjame. "DynASt: Efficient maintenance of agree-sets against dynamic datasets." In: *Proc. 26th Int. Conf. Extending Database Technol.* 2023, pp. 14–26.

[22] Donald J Berndt and James Clifford. "Using dynamic time warping to find patterns in time series." In: *Proceedings of the 3rd international conference on knowledge discovery and data mining*. 1994, pp. 359–370.

[23] Laure Berti-Équille, Hazar Harmouch, Felix Naumann, Noël Novelli, and Saravanan Thirumuruganathan. "Discovery of genuine functional dependencies from relational data with missing values." In: *Proceedings of the VLDB Endowment* 11.8 (2018), pp. 880–892.

[24] Leopoldo Bertossi, Solmaz Kolahi, and Laks VS Lakshmanan. "Data cleaning and query answering with matching dependencies and matching functions." In: *Proceedings of the 14th International Conference on Database Theory*. 2011, pp. 268–279.

[25] Paul Boniol, John Paparrizos, and Themis Palpanas. "An Interactive Dive into Time-Series Anomaly Detection." In: *IEEE ICDE*. 2024.

[26] Bernardo Breve, Loredana Caruccio, Stefano Cirillo, Vincenzo Deufemia, Gianpaolo Iuliano, Giuseppe Polese, and Roberto Stanzione. "Data Dependency Discovery meets Missing Value Imputation: Highlighting Dynamic Synergies." In: *29th International Conference on Extending Database Technology, EDBT 2026 (Submitted)*.

[27] Bernardo Breve, Loredana Caruccio, Stefano Cirillo, Vincenzo Deufemia, and Giuseppe Polese. "Dependency Visualization in Data Stream Profiling." In: *Big Data Res.* 25 (2021), p. 100240.

[28] Bernardo Breve, Loredana Caruccio, Stefano Cirillo, Vincenzo Deufemia, and Giuseppe Polese. "IndiBits: Incremental Discovery of Relaxed Functional Dependencies using Bitwise Similarity." In: *39th IEEE International Conference on Data Engineering, ICDE 2023, Anaheim, CA, USA, April 3-7, 2023*. IEEE, 2023, pp. 1393–1405.

[29] Bernardo Breve, Loredana Caruccio, Stefano Cirillo, Vincenzo Deufemia, and Giuseppe Polese. "Decentralized and Incremental Discovery of Relaxed Functional Dependencies Using Bitwise Similarity." In: *IEEE Transactions on Knowledge and Data Engineering* 36.12 (2024), pp. 7380–7398.

[30] Bernardo Breve, Loredana Caruccio, Vincenzo Deufemia, and Giuseppe Polese. "RENUVER: A Missing Value Imputation Algorithm based on Relaxed Functional Dependencies." In: *Proceedings of the 25th International Conference on Extending Database Technology, EDBT 2022, Edinburgh, UK, March 29 - April 1, 2022*. OpenProceedings.org, 2022, 1:52–1:64.

[31] Giulia Bruno, Paolo Garza, Elisa Quintarelli, and Rosalba Rossato. "Anomaly detection through quasi-functional dependency analysis." In: *Journal of Digital Information Management* 5.4 (2007).

[32] Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. "Brits: Bidirectional recurrent imputation for time series." In: *NeurIPS* 31 (2018).

[33] Riccardo Cappuzzo, Saravanan Thirumuruganathan, and Paolo Papotti. "Relational data imputation with graph neural networks." In: *EDBT/ICDT 2024, 27th International Conference on Extending Database Technology*. 2024.

[34] Loredana Caruccio, Gaetano Cimino, Stefano Cirillo, Domenico Desiato, Giuseppe Polese, and Genoveffa Tortora. "Malicious account identification in social network platforms." In: *ACM Transactions on Internet Technology* 23.4 (2023), pp. 1–25.

[35] Loredana Caruccio and Stefano Cirillo. "Incremental Discovery of Imprecise Functional Dependencies." In: *ACM J. Data Inf. Qual.* 12.4 (2020), 19:1–19:25.

[36] Loredana Caruccio, Stefano Cirillo, Vincenzo Deufemia, and Giuseppe Polese. "Incremental Discovery of Functional Dependencies with a Bit-vector Algorithm." In: *Proceedings of the 27th Italian Symposium on Advanced Database Systems, Castiglione della Pescaia (Grosseto), Italy, June 16-19, 2019*. Vol. 2400. CEUR Workshop Proceedings. CEUR-WS.org, 2019.

[37] Loredana Caruccio, Stefano Cirillo, Vincenzo Deufemia, and Giuseppe Polese. "Efficient Discovery of Functional Dependencies from Incremental Databases." In: *iiWAS2021: The 23rd International Conference on Information Integration and Web Intelligence, Linz, Austria, 29 November 2021 - 1 December 2021*. ACM, 2021, pp. 400–409.

[38] Loredana Caruccio, Stefano Cirillo, Vincenzo Deufemia, Giuseppe Polese, and Roberto Stanzione. "Data Analytics on Twitter for Evaluating Women Inclusion and Safety in Modern Society." In: *Proceedings of the 1st Italian Conference on Big Data and Data Science (ITADATA) 2022*. Milan, Italy: CEUR-WS.org, 2022, pp. 75–86.

[39] Loredana Caruccio, Stefano Cirillo, Vincenzo Deufemia, Giuseppe Polese, Roberto Stanzione, et al. "REQUIRED:

A Tool to Relax Queries through Relaxed Functional Dependencies." In: *EDBT*. 2023, pp. 823–826.

[40] Loredana Caruccio, Stefano Cirillo, Vincenzo Deufemia, Giuseppe Polese, et al. "Incremental Discovery of Functional Dependencies with a Bit-vector Algorithm." In: *Proceedings of the 27th Italian Symposium on Advanced Database Systems*. Castiglione della Pescaia (Grosseto), Italy: CEUR-WS.org, 2019.

[41] Loredana Caruccio, Stefano Cirillo, Gianpaolo Iuliano, Giuseppe Polese, and Roberto Stanzione. "RYAN: A tool for explaining and visually analyzing the evolution of Relaxed Functional Dependencies." In: *2024 IEEE International Conference on Big Data (BigData)*. IEEE. 2024, pp. 1249–1254.

[42] Loredana Caruccio, Stefano Cirillo, Gianpaolo Iuliano, Giuseppe Polese, and Roberto Stanzione. "Specializing an RFD discovery algorithm for sentence-level semantic similarity." In: *Data & Knowledge Engineering (Submitted)* (2025).

[43] Loredana Caruccio, Stefano Cirillo, Giuseppe Polese, Giandomenico Solimando, Shanmugam Sundaramurthy, and Genoveffa Tortora. "Claude 2.0 Large Language Model: tackling a real-world classification problem with a new Iterative Prompt Engineering approach." In: *Intelligent Systems with Applications* (2024), p. 200336.

[44] Loredana Caruccio, Stefano Cirillo, Giuseppe Polese, and Roberto Stanzione. "An RFD-based approach for concept drift detection in Machine Learning Systems." In: *Proceedings 28th International Conference on Extending Database Technology, EDBT 2025, Barcelona, Spain, March 25-28, 2025*. OpenProceedings.org, 2025, pp. 816–828.

[45] Loredana Caruccio, Vincenzo Deufemia, Felix Naumann, and Giuseppe Polese. "Discovering Relaxed Functional Dependencies Based on Multi-Attribute Dominance." In: *IEEE Trans. Knowl. Data Eng.* 33.9 (2021), pp. 3212–3228.

[46] Loredana Caruccio, Vincenzo Deufemia, and Giuseppe Polese. "Relaxed functional dependencies—a survey of approaches." In: *IEEE Transactions on knowledge and data engineering* 28.1 (2015), pp. 147–165.

[47] Loredana Caruccio, Vincenzo Deufemia, and Giuseppe Polese. "Learning Effective Query Management Strategies from Big Data." In: *16th IEEE International Conference on Machine Learning and Applications, ICMLA 2017, Cancun, Mexico, December 18-21, 2017*. IEEE, 2017, pp. 643–648.

[48] Loredana Caruccio, Vincenzo Deufemia, and Giuseppe Polese. "Visualization of (multimedia) dependencies from big data." In: *Multim. Tools Appl.* 78.23 (2019), pp. 33151–33167.

[49] Loredana Caruccio, Vincenzo Deufemia, and Giuseppe Polese. "Mining relaxed functional dependencies from data." In: *Data Min. Knowl. Discov.* 34.2 (2020), pp. 443–477.

[50] Surajit Chaudhuri and Kyuseok Shim. "Including group-by in query optimization." In: *VLDB*. Vol. 94. 1994, pp. 12–15.

[51] Haipeng Chen, Sushil Jajodia, Jing Liu, Noseong Park, Vadim Sokolov, and VS Subrahmanian. "FakeTables: Using GANs to generate functional dependency preserving tables with bounded real data." In: *IJCAI*. 2019, pp. 2074–2080.

[52] Debo Cheng, Shichao Zhang, Zhenyun Deng, Yonghua Zhu, and Ming Zong. "kNN Algorithm with Data-Driven k Value." In: *Advanced Data Mining and Applications - 10th International Conference, ADMA 2014, Guilin, China, December 19-21, 2014. Proceedings*. Vol. 8933. Lecture Notes in Computer Science. Springer, 2014, pp. 499–512.

[53] Chun-Wei Chiang and Ming Yin. "You'd better stop! Understanding human reliance on machine learning models under covariate shift." In: *Proceedings of the 13th ACM Web Science Conference 2021*. United Kingdom: ACM, 2021, pp. 120–129.

[54] Fei Chiang and Renée J. Miller. "Discovering data quality rules." In: *Proc. VLDB Endow.* 1.1 (2008), pp. 1166–1177.

[55] Xu Chu, Ihab F Ilyas, and Paolo Papotti. "Holistic data cleaning: Putting violations into context." In: *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. IEEE. 2013, pp. 458–469.

[56] Cristiano Codagnone, Giovanni Liva, Laura Gunderson, Gianluca Misuraca, and Emanuele Rebesco. "Europe's digital decade and autonomy." In: *Study Requested by the ITRE Committee. Policy Department for Economic, Scientific and Quality of Life Policies, European Parliament, Luxembourg* (2021).

[57] Edgar F Codd. "A relational model of data for large shared data banks." In: *Communications of the ACM* 13.6 (1970), pp. 377–387.

[58] Carlo Combi, Matteo Mantovani, Alberto Sabaini, Pietro Sala, Francesco Amaddeo, Ugo Moretti, and Giuseppe Pozzi. "Mining approximate temporal functional dependencies with pure temporal grouping in clinical databases." In: *Comput. Biol. Medicine* 62 (2015), pp. 306–324.

[59] Qin Cui, Wenkui Zheng, Wei Hou, Ming Sheng, Peng Ren, Wang Chang, and XiangYang Li. "Holocleanx: A multi-source heterogeneous data cleaning solution based on lakehouse." In: *International Conference on Health Information Science*. Springer. 2022, pp. 165–176.

[60] Christopher John Date. *Relational Database: Writings, 1989-1991*. Addison-Wesley Longman Publishing Co., Inc., 1992.

[61] Pratim Datta, Laurie Walker, and Fabrizio Amarilli. "Digital transformation: Learning from Italy's public administration." In: *Journal of Information Technology Teaching Cases* 10.2 (2020), pp. 54–71.

[62] Massimiliano De Benedetti, Fabio Leonardi, Fabrizio Messina, Corrado Santoro, and Athanasios Vasilakos. "Anomaly detection and predictive maintenance for photovoltaic systems." In: *Neurocomputing* 310 (2018), pp. 59–68.

[63] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." In: *arXiv preprint arXiv:1810.04805* (2018).

[64] Yujuan Ding, Wenqi Fan, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. "A Survey on RAG Meets LLMs: Towards Retrieval-Augmented Large Language Models." In: *arXiv preprint* (2024).

[65] Gregory Ditzler and Robi Polikar. "Hellinger distance based drift detection for nonstationary environments." In: *2011 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments, CIDUE 2011, Paris, France, April 13, 2011*. IEEE, 2011, pp. 41–48.

[66] Hong-Hai Do and Erhard Rahm. "COMA—a system for flexible combination of schema matching approaches." In: *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*. Elsevier. 2002, pp. 610–621.

[67] Wenjie Du. "PyPOTS: a Python toolbox for data mining on Partially-Observed Time Series." In: *arXiv preprint* (2023).

[68] Wenjie Du, David Côté, and Yan Liu. "Saits: Self-attention-based imputation for time series." In: *Expert Systems with Applications* 219 (2023), p. 119619.

[69] Marius Eich, Pit Fender, and Guido Moerkotte. "Faster plan generation through consideration of functional dependencies and keys." In: *Proceedings of the VLDB Endowment* 9.10 (2016), pp. 756–767.

[70] Tlamelo Emmanuel, Thabiso M. Maupong, Dimane Mpoeleng, Thabo Semong, Banyatsang Mphago, and Oteng Tabona. "A survey on missing data in machine learning." In: *J. Big Data* 8.1 (2021), p. 140.

[71] Pablo A. Estevez, Michel Tesmer, Claudio A. Perez, and Jacek M. Zurada. "Normalized Mutual Information Feature Selection." In: *IEEE Transactions on Neural Networks* 20.2 (2009), pp. 189–201.

[72] Ronald Fagin. "Multivalued dependencies and a new normal form for relational databases." In: *ACM Transactions on Database Systems (TODS)* 2.3 (1977), pp. 262–278.

[73] Seyed Mostafa Fakhrahmad, M. H. Sadreddini, and Mansoor Zolghadri Jahromi. "AD-Miner: A new incremental method for discovery of minimal approximate dependencies using logical operations." In: *Intell. Data Anal.* 12.6 (2008), pp. 607–619.

[74] Wenfei Fan, Hong Gao, Xibei Jia, Jianzhong Li, and Shuai Ma. "Dynamic constraints for record matching." In: *VLDB J.* 20.4 (2011), pp. 495–520.

[75] Wenfei Fan, Floris Geerts, Laks V. S. Lakshmanan, and Ming Xiong. "Discovering Conditional Functional Dependencies." In: *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China*. IEEE Computer Society, 2009, pp. 1231–1234.

[76] Sébastien Ferré. "Answers Partitioning and Lazy Joins for Efficient Query Relaxation and Application to Similarity Search." In: *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*. Vol. 10843. Lecture Notes in Computer Science. Springer, 2018, pp. 209–224.

[77] Marie Fischer, Paul Roessler, Paul Sieben, Janina Adamcic, Christoph Kirchherr, Tobias Straeubig, Youri Kaminsky, and Felix Naumann. "BCNF*-From Normalized-to Star-Schemas and Back Again." In: *Companion of the 2023 International Conference on Management of Data*. 2023, pp. 103–106.

[78] Peter A. Flach and Iztok Savnik. "Database Dependency Discovery: A Machine Learning Approach." In: *AI Commun.* 12.3 (1999), pp. 139–160.

[79] Vincent Fortuin, Dmitry Baranchuk, Gunnar Rätsch, and Stephan Mandt. "Gp-vae: Deep probabilistic time series imputation." In: *AISTATS*. 2020.

[80] Satish Gajawada and Durga Toshniwal. "Missing value imputation method based on clustering and nearest neighbours." In: *International Journal of Future Computer and Communication* 1.2 (2012), pp. 206–208.

[81] João Gama, Pedro Medas, Gladys Castillo, and Pedro Pereira Rodrigues. "Learning with Drift Detection." In: *Advances in Artificial Intelligence - SBIA 2004, 17th Brazilian Symposium on Artificial Intelligence, São Luis, Maranhão, Brazil, September 29 - October 1, 2004, Proceedings*. Vol. 3171. Lecture Notes in Computer Science. Springer, 2004, pp. 286–295.

[82] Cristiano Garcia, Daniel F. Leite, and Igor Skrjanc. "Incremental Missing-Data Imputation for Evolving Fuzzy Granular Prediction." In: *IEEE Trans. Fuzzy Syst.* 28.10 (2020), pp. 2348–2362.

[83] Mouzhi Ge and Markus Helfert. "A Review of Information Quality Research - Develop a Research Agenda." In: Jan. 2007, pp. 76–91.

[84] Floris Geerts, Giansalvatore Mecca, Paolo Papotti, and Donatello Santoro. "The LLUNATIC data-cleaning framework." In: *Proceedings of the VLDB Endowment* 6.9 (2013), pp. 625–636.

[85] Floris Geerts, Giansalvatore Mecca, Paolo Papotti, and Donatello Santoro. "Mapping and cleaning." In: *2014 IEEE 30th International Conference on Data Engineering*. IEEE. 2014, pp. 232–243.

[86] Albrecht Gnauck. "Interpolation and approximation of water quality time series and process identification." In: *Analytical and bioanalytical chemistry* 380 (2004).

[87] Lukasz Golab, Howard J. Karloff, Flip Korn, Divesh Srivastava, and Bei Yu. "On generating near-optimal tableaux for conditional functional dependencies." In: *Proceeding of the VLDB Endow.* 1.1 (2008), pp. 376–390.

[88] Igor Goldenberg and Geoffrey I. Webb. "Survey of distance measures for quantifying concept drift and shift in numeric data." In: *Knowl. Inf. Syst.* 60.2 (2019), pp. 591–615.

[89] Igor Goldenberg and Geoffrey I. Webb. "PCA-based drift and shift quantification framework for multidimensional data." In: *Knowl. Inf. Syst.* 62.7 (2020), pp. 2835–2854.

[90] Lovedeep Gondara and Ke Wang. "MIDA: Multiple Imputation Using Denoising Autoencoders." In: *Advances in Knowledge Discovery and Data Mining - 22nd Pacific-Asia Conference, PAKDD 2018, Melbourne, VIC, Australia, June 3-6, 2018, Proceedings, Part III*. 2018.

[91] Georg Gottlob, Stephanie Tien Lee, Gregory Valiant, and Paul Valiant. "Size and treewidth bounds for conjunctive queries." In: *Journal of the ACM (JACM)* 59.3 (2012), pp. 1–35.

[92] Ömer Gözüaçik, Alican Büyükçakir, Hamed R. Bonab, and Fazli Can. "Unsupervised Concept Drift Detection with a Discriminative Classifier." In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*. Beijing, China: ACM, 2019, pp. 2365–2368.

[93] Ege Berkay Gulcan and Fazli Can. "Unsupervised concept drift detection for multi-label data streams." In: *Artif. Intell. Rev.* 56.3 (2023), pp. 2401–2434.

[94] Francesco Gullo, Giovanni Ponti, Andrea Tagarelli, Giuseppe Tradigo, and Pierangelo Veltri. "A time series approach for clustering mass spectrometry data." In: *Journal of Computational Science* 3.5 (2012).

[95] Jeffrey Heer, Maneesh Agrawala, and Wesley Willett. "Generalized selection via interactive query relaxation." In: *Proceedings of the 2008 Conference on Human Factors in Computing Systems, CHI 2008, 2008, Florence, Italy, April 5-10, 2008*. ACM, 2008, pp. 959–968.

[96] Arvid Heise, Jorge-Arnulfo Quiané-Ruiz, Ziawasch Abedjan, Anja Jentzsch, and Felix Naumann. "Scalable Discovery of Unique Column Combinations." In: *Proc. VLDB Endow.* 7.4 (2013), pp. 301–312.

[97] Ernst Hellinger. "Neue begründung der theorie quadratischer formen von unendlichvielen veränderlichen." In: *Journal für die reine und angewandte Mathematik* 1909.136 (1909), pp. 210–271.

[98] Mohammad Kazim Hooshmand and Doreswamy Hosahalli. "Network anomaly detection using deep learning techniques." In: *CAAI Transactions on Intelligence Technology* 7.2 (2022), pp. 228–243.

[99] David Tse Jung Huang, Yun Sing Koh, Gillian Dobbie, and Russel Pears. "Detecting Volatility Shift in Data Streams." In: *2014 IEEE International Conference on Data Mining, ICDM 2014, Shenzhen, China, December 14-17, 2014*. Shenzhen, China: IEEE Computer Society, 2014, pp. 863–868.

[100] Dongxu Huang, Dejun Mu, Libin Yang, and Xiaoyan Cai. "CoDetect: Financial fraud detection with anomaly feature detection." In: *IEEE Access* 6 (2018), pp. 19161–19174.

[101] Ykä Huhtala, Juha Kärkkäinen, Pasi Porkka, and Hannu Toivonen. "TANE: An Efficient Algorithm for Discovering Functional and Approximate Dependencies." In: *Comput. J.* 42.2 (1999), pp. 100–111.

[102] David S Johnson and Michael R Garey. *Computers and intractability: A guide to the theory of NP-completeness*. W. H. Freeman & Co., 1979.

[103] SUNGJIN KIM, KWANGGUK HAN, JINSONG KIM, and KWANGSONG RI. "Multivalued Dependency and Normal Form for XML Documents." In: ().

[104] Md Alamgir Kabir, Jacky W Keung, Kwabena E Bennin, and Miao Zhang. "Assessing the significant impact of concept drift in software defect prediction." In: *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*. Vol. 1. IEEE. 2019, pp. 53–58.

[105] Youri Kaminsky, Eduardo HM Pena, and Felix Naumann. "Discovering similarity inclusion dependencies." In: *Proceedings of the ACM on Management of Data* 1.1 (2023), pp. 1–24.

[106] Maurice G Kendall. "A new measure of rank correlation." In: *Biometrika* 30.1-2 (1938), pp. 81–93.

[107] Ronald S. King and James J. Legendre. "Discovery of functional and approximate functional dependencies in relational databases." In: *Adv. Decis. Sci.* 7.1 (2003), pp. 49–59.

[108] Jan Kossmann, Thorsten Papenbrock, and Felix Naumann. "Data dependencies for query optimization: a survey." In: *The VLDB Journal* 31.1 (2022), pp. 1–22.

[109] Lyudmyla F Kozachenko and Nikolai N Leonenko. "Sample estimate of the entropy of a random vector." In: *Problemy Peredachi Informatsii* 23.2 (1987), pp. 9–16.

[110] Georg Krempl, Indre Žliobaite, Dariusz Brzeziński, Eyke Hüllermeier, Mark Last, Vincent Lemaire, Tino Noack, Ammar Shaker, Sonja Sievi, Myra Spiliopoulou, et al. "Open challenges for data stream mining research." In: *ACM SIGKDD explorations newsletter* 16.1 (2014), pp. 1–10.

[111] Shulei Kuang, Honghui Yang, Zijing Tan, and Shuai Ma. "Efficient differential dependency discovery." In: *Proceedings of the VLDB Endowment* 17.7 (2024), pp. 1552–1564.

[112] Selasi Kwashie, Jixue Liu, Jiuyong Li, and Feiyue Ye. "Mining Differential Dependencies: A Subspace Clustering Approach." In: *Databases Theory and Applications - 25th Australasian Database Conference, ADC 2014, Brisbane, QLD, Australia, July 14-16, 2014. Proceedings*. Vol. 8506. Lecture Notes in Computer Science. Springer, 2014, pp. 50–61.

[113] Alexandros Labrinidis and Hosagrahar V Jagadish. "Challenges and opportunities with big data." In: *Proceedings of the VLDB Endowment* 5.12 (2012), pp. 2032–2033.

[114] Marie Le Guilly, Jean-Marc Petit, and Marian Scuturici. "A First Experimental Study on Functional Dependencies for Imbalanced Datasets Classification." In: *International Workshop on Information Search, Integration, and Personalization*. Springer. 2018, pp. 116–133.

[115] Marie Le Guilly, Jean-Marc Petit, and Vasile-Marian Scuturici. "Evaluating classification feasibility using functional dependencies." In: *Transactions on Large-Scale Data and Knowledge Centered Systems XLIV: Special Issue on Data Management–Principles, Technologies, and Applications* (2020), pp. 132–159.

[116] Mathieu Lepot, Jean-Baptiste Aubin, and François HLR Clemens. "Interpolation in time series: An introductive overview of existing methods, their performance criteria and uncertainty assessment." In: *Water* 9.10 (2017).

[117] V Levenshtein. "Binary codes capable of correcting deletions, insertions, and reversals." In: *Proceedings of the Soviet physics doklady* 1 (1966), pp. 1–15.

[118] Chen Li, Jinha Park, Hahyeon Kim, and Dimitrios Chrysostomou. "How can I help you? An Intelligent Virtual Assistant for Industrial Robots." In: *Companion of the 2021 ACM/IEEE International Conference on Human-Robot Interaction, HRI 2021*. Boulder, CO, USA: ACM, 2021, pp. 220–224.

[119] Shuxin Li, Gong Cheng, and Chengkai Li. "Relaxing relationship queries on graph data." In: *J. Web Semant.* 61-62 (2020), p. 100557.

[120] Wen-Syan Li and Chris Clifton. "SEMINT: A tool for identifying attribute correspondences in heterogeneous databases using neural networks." In: *Data & Knowledge Engineering* 33.1 (2000), pp. 49–84.

[121] Daniel Lindner, Daniel Ritter, and Felix Naumann. "Enabling data dependency-based query optimization." In: *arXiv preprint arXiv:2406.06886* (2024).

[122] Jixue Liu, Jiuyong Li, Chengfei Liu, and Yongfeng Chen. "Discover Dependencies from Data - A Review." In: *IEEE Trans. Knowl. Data Eng.* 24.2 (2012), pp. 251–264.

[123] Peng Liu, Lei Lei, and Naijun Wu. "A Quantitative Study of the Effect of Missing Data in Classifiers." In: *Fifth International Conference on Computer and Information Technology (CIT 2005), 21-23 September 2005, Shanghai, China*. IEEE Computer Society, 2005, pp. 28–33.

[124] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing." In: *ACM Computing Surveys* (2023).

[125] Shangdong Liu, Lili Lu, Yongpan Zhang, Tong Xin, Yimu Ji, and Ruchuan Wang. "Research on concept drift detection for decision tree algorithm in the stream of big data." In: *Proceedings of the 8th International Symposium on Parallel Architecture, Algorithm and Programming PAAP 2017, Proceedings 8*. Haikou, China: Springer, 2017, pp. 237–246.

[126] Stéphane Lopes, Jean-Marc Petit, and Lotfi Lakhal. "Efficient Discovery of Functional Dependencies and Armstrong Relations." In: *Advances in Database Technology - EDBT 2000, 7th International Conference on Extending Database Technology, Konstanz, Germany, March 27-31, 2000, Proceedings*. Vol. 1777. Lecture Notes in Computer Science. Springer, 2000, pp. 350–364.

[127] David Loshin. "Evaluating the business impacts of poor data quality." In: *Information Quality Journal* (2011).

[128] Sabine Massmann, Salvatore Raunich, David Aumüller, Patrick Arnold, Erhard Rahm, et al. "Evolution of the COMA match system." In: *Ontology Matching* 49 (2011), pp. 49–60.

[129] Robert J. May, Holger R. Maier, Graeme C. Dandy, and T.M.K. Gayani Fernando. "Non-linear variable selection for artificial neural networks using partial mutual information." In: *Environmental Modelling & Software* 23.10 (2008), pp. 1312–1326. ISSN: 1364-8152.

[130] John T. McCoy, Steve Kroon, and Lidia Auret. "Variational Autoencoders for Missing Data Imputation with Application to a Simulated Milling Circuit." In: *IFAC-PapersOnLine* (2018).

[131] Giansalvatore Mecca, Paolo Papotti, Donatello Santoro, and Enzo Veltri. "BUNNI: Learning repair actions in rule-driven data cleaning." In: *ACM Journal of Data and Information Quality* 16.2 (2024), pp. 1–31.

[132] Xiaoye Miao, Yangyang Wu, Lu Chen, Yunjun Gao, and Jianwei Yin. "An experimental survey of missing data imputation algorithms." In: *IEEE Transactions on Knowledge and Data Engineering* 35.7 (2022), pp. 6630–6650.

[133] Davide Mottin, Alice Marascu, Senjuti Basu Roy, Gautam Das, Themis Palpanas, and Yannis Velegrakis. "IQR: an interactive query relaxation system for the empty-answer problem." In: *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*. ACM, 2014, pp. 1095–1098.

[134] Jan Pawel Musial, Michael M Verstraete, and Nadine Gobron. "Comparing the effectiveness of recent algorithms to fill and smooth incomplete and noisy time series." In: *Atmospheric chemistry and physics* 11.15 (2011), pp. 7905–7923.

[135] Ion Muslea. "Machine learning for online query relaxation." In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004*. ACM, 2004, pp. 246–255.

[136] Ion Muslea and Thomas J. Lee. "Online Query Relaxation via Bayesian Causal Structures Discovery." In: *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*. AAAI Press / The MIT Press, 2005, pp. 831–836.

[137] Ullas Nambiar and Subbarao Kambhampati. "Mining Approximate Functional Dependencies and Concept Similarities to Answer Imprecise Queries." In: *Proceedings of the Seventh International Workshop on the Web and Databases, WebDB 2004, June 17-18, 2004, Maison de la Chimie, Paris, France, Colocated with ACM SIGMOD/PODS 2004*. ACM, 2004, pp. 73–78.

[138] Alfredo Nazábal, Pablo M. Olmos, Zoubin Ghahramani, and Isabel Valera. "Handling incomplete heterogeneous data using VAEs." In: *Pattern Recognit.* (2020).

[139] Kyosukef Nishida and Koichiro Yamauchi. "Detecting concept drift using statistical testing." In: *Proceedings of the 10th International Conference on Discovery Science DS 2007*. Sendai, Japan: Springer, 2007, pp. 264–269.

[140] Noël Novelli and Rosine Cicchetti. "FUN: An Efficient Algorithm for Mining Functional and Embedded Dependencies." In: *Database Theory - ICDT 2001, 8th International Conference, London, UK, January 4-6, 2001, Proceedings*. Vol. 1973. Lecture Notes in Computer Science. Springer, 2001, pp. 189–203.

[141] Muhammad S Osman, Adnan M Abu-Mahfouz, and Philip R Page. "A survey on data imputation techniques: Water distribution system as a use case." In: *IEEE Access* 6 (2018), pp. 63279–63291.

[142] John Paparrizos and Luis Gravano. "Fast and Accurate Time-Series Clustering." In: 42.2 (June 2017). ISSN: 0362-5915.

[143] Thorsten Papenbrock, Tanja Bergmann, Moritz Finke, Jakob Zwiener, and Felix Naumann. "Data Profiling with Metanome." In: *Proc. VLDB Endow.* 8.12 (2015), pp. 1860–1863.

[144] Thorsten Papenbrock, Jens Ehrlich, Jannik Marten, Tommy Neubert, Jan-Peer Rudolph, Martin Schönberg, Jakob Zwiener, and Felix Naumann. "Functional Dependency Discovery: An Experimental Evaluation of Seven Algorithms." In: *Proc. VLDB Endow.* 8.10 (2015), pp. 1082–1093.

[145] Thorsten Papenbrock and Felix Naumann. "Data-driven Schema Normalization." In: *EDBT*. Vol. 17. 2017, pp. 342–353.

[146] Hae-Sang Park and Chi-Hyuck Jun. "A simple and fast algorithm for K-medoids clustering." In: *Expert systems with applications* 36.2 (2009).

[147] Ricardo Cardoso Pereira, Pedro Henriques Abreu, Pedro Pereira Rodrigues, and Mário AT Figueiredo. "Imputation of data missing not at random: Artificial generation and benchmark analysis." In: *Expert Systems with Applications* 249 (2024), p. 123654.

[148] Leo L Pipino, Yang W Lee, and Richard Y Wang. "Data quality assessment." In: *Communications of the ACM* 45.4 (2002), pp. 211–218.

[149] Viswanath Poosala and Venkatesh Ganti. "Fast Approximate Query Answering Using Precomputed Statistics." In: *Proceedings of the 15th International Conference on Data Engineering, Sydney, Australia, March 23-26, 1999*. IEEE Computer Society, 1999, p. 252.

[150] Ivens Portugal, Paulo S. C. Alencar, and Donald D. Cowan. "The use of machine learning algorithms in recommender systems: A systematic review." In: *Expert Systems with Applications* 97 (2018), pp. 205–227.

[151] Abdulhakim Ali Qahtan, Basma Alharbi, Suojin Wang, and Xiangliang Zhang. "A PCA-Based Change Detection Framework for Multidimensional Data Streams: Change Detection in Multidimensional Data Streams." In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*. Sydney, Australia: ACM, 2015, pp. 935–944.

[152] Trivellore E Raghunathan, James M Lepkowski, John Van Hoewyk, Peter Solenberger, et al. "A multivariate technique for multiply imputing missing values using a sequence of regression models." In: *Survey methodology* 27.1 (2001), pp. 85–96.

[153] Md. Geaur Rahman and Md Zahidul Islam. "Missing value imputation using decision trees and decision forests by splitting and merging records: Two novel techniques." In: *Knowl. Based Syst.* 53 (2013), pp. 51–65.

[154] Md. Geaur Rahman and Md Zahidul Islam. "Missing value imputation using a fuzzy clustering-based EM approach." In: *Knowl. Inf. Syst.* 46.2 (2016), pp. 389–422.

[155] Nils Reimers and Iryna Gurevych. "Sentence-bert: Sentence embeddings using siamese bert-networks." In: *arXiv preprint arXiv:1908.10084* (2019).

[156] Nils Reimers and Iryna Gurevych. "Making monolingual sentence embeddings multilingual using knowledge distillation." In: *arXiv preprint arXiv:2004.09813* (2020).

[157] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. "HoloClean: Holistic Data Repairs with Probabilistic Inference." In: *Proc. VLDB Endow.* 10.11 (2017), pp. 1190–1201.

[158] El Kindi Rezig, Mourad Ouzzani, Walid G Aref, Ahmed K Elmagarmid, Ahmed R Mahmood, and Michael Stonebraker. "Horizon: Scalable dependency-driven data cleaning." In: *Proceedings of the VLDB Endowment* 14.11 (2021), pp. 2546–2554.

[159] Donald B Rubin. "Inference and missing data." In: *Biometrika* 63.3 (1976), pp. 581–592.

[160] Edin Šabić, David Keeley, Bailey Henderson, and Sara Nannemann. "Healthcare and anomaly detection: using machine learning to predict anomalies in heart rate data." In: *AI & SOCIETY* 36.1 (2021), pp. 149–158.

[161] Hiroaki Sakoe and Seibi Chiba. "Dynamic programming algorithm optimization for spoken word recognition." In: *IEEE transactions on acoustics, speech, and signal processing* 26.1 (2003), pp. 43–49.

[162] Riyanarto Sarno, Fernandes Sinaga, and Kelly Rossa Sungkono. "Anomaly detection in business processes using process mining and fuzzy association rule learning." In: *Journal of Big Data* 7.1 (2020), p. 5.

[163] Philipp Schirmer, Thorsten Papenbrock, Sebastian Kruse, Felix Naumann, Dennis Hempfing, Torben Mayer, and Daniel Neuschäfer-Rube. "DynFD: Functional Dependency Discovery in Dynamic Datasets." In: *Advances in Database Technology - 22nd International Conference on Extending Database Technology, EDBT 2019, Lisbon, Portugal, March 26-29, 2019*. OpenProceedings.org, 2019, pp. 253–264.

[164] Amir Masoud Sefidian and Negin Daneshpour. "Missing value imputation using a novel grey based fuzzy c-means, mutual information based feature selection, and regression model." In: *Expert Syst. Appl.* 115 (2019), pp. 68–94.

[165] David Simmen, Eugene Shekita, and Timothy Malkemus. "Fundamental techniques for order optimization." In: *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*. 1996, pp. 57–67.

[166] Jaime Céspedes Sisniega and Álvaro López García. "Frouros: An open-source Python library for drift detection in machine learning systems." In: *SoftwareX* 26 (2024), p. 101733.

[167] Shaoxu Song. "Data Dependencies in the Presence of Difference." PhD thesis. The Hong Kong University, 2010.

[168] Shaoxu Song and Lei Chen. "Efficient discovery of similarity constraints for matching dependencies." In: *Data & Knowledge Engineering* 87 (2013), pp. 146–166.

[169] Shaoxu Song, Fei Gao, Ruihong Huang, and Chaokun Wang. "Data Dependencies over Big Data: A Family Tree." In: *IEEE Transactions on Knowledge and Data Engineering* (2020).

[170] Shaoxu Song, Yu Sun, Aoqian Zhang, Lei Chen, and Jianmin Wang. "Enriching Data Imputation under Similarity Rule Constraints." In: *IEEE Trans. Knowl. Data Eng.* 32.2 (2020), pp. 275–287.

[171] Indro Spinelli, Simone Scardapane, and Aurelio Uncini. "Missing data imputation with adversarially-trained graph convolutional networks." In: *Neural Networks* (2020).

[172] Roberto Stanzione, Paul Boniol, Loredana Caruccio, Themis Palpanas, and Giuseppe Polese. "RIFT: Scaling Relaxed Functional Dependencies for Imputing Missing Values in Time series." In: *IEEE International Conference on Data Engineering (ICDE 2026) (Submitted)*. IEEE.

[173] Michael J Swain and Dana H Ballard. "Color indexing." In: *International journal of computer vision* 7.1 (1991), pp. 11–32.

[174] Jaroslaw Szlichta, Parke Godfrey, and Jarek Gryz. "Fundamentals of order dependencies." In: *arXiv preprint arXiv:1208.0084* (2012).

[175] Gabriel Marques Tavares and Sylvio Barbon Junior. "Process mining encoding via meta-learning for an enhanced anomaly detection." In: *Proceedings of the European Conference on Advances in Databases and Information Systems ADBIS 2021*. Tartu, Estonia: Springer, 2021, pp. 157–168.

[176] Fabian Tschirschnitz, Thorsten Papenbrock, and Felix Naumann. "Detecting inclusion dependencies on very many tables." In: *ACM Transactions on Database Systems (TODS)* 42.3 (2017), pp. 1–29.

[177] Stef Van Buuren and Catharina GM Oudshoorn. *Multivariate imputation by chained equations*. 2000.

[178] Chunzhi Wang, Nataliya Shakhovska, Anatoliy Sachenko, and Myroslav Komar. "A New Approach for Missing Data Imputation in Big Data Interface." In: *Inf. Technol. Control.* 49.4 (2020), pp. 541–555.

[179] Daisy Zhe Wang, Xin Luna Dong, Anish Das Sarma, Michael J Franklin, and Alon Y Halevy. "Functional Dependency Generation and Applications in Pay-As-You-Go Data Integration Systems." In: *WebDB*. 2009.

[180] Shyue-Liang Wang, Ju-Wen Shen, and Tzung-Pei Hong. "Incremental discovery of functional dependencies using partitions." In: *Proceedings Joint 9th IFSA World Congress and 20th NAFIPS International Conference (Cat. No. 01TH8569)*. Vol. 3. IEEE. 2001, pp. 1322–1326.

[181] Joe H Ward Jr. "Hierarchical grouping to optimize an objective function." In: *Journal of the American statistical association* 58.301 (1963), pp. 236–244.

[182] Geoffrey I Webb, Roy Hyde, Hong Cao, Hai Long Nguyen, and Francois Petitjean. "Characterizing concept drift." In: *Data Mining and Knowledge Discovery* 30.4 (2016), pp. 964–994.

[183] Grant E Weddell. "Reasoning about functional dependencies generalized for semantic data models." In: *ACM Transactions on Database Systems (TODS)* 17.1 (1992), pp. 32–64.

[184] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. "Timesnet: Temporal 2d-variation modeling for general time series analysis." In: *arXiv preprint* (2022).

[185] Richard Wu, Aoqian Zhang, Ihab F. Ilyas, and Theodoros Rekatsinas. "Attention-based Learning for Missing Data Imputation in HoloClean." In: *Proceedings of the Third Conference on Machine Learning and Systems, MLSys 2020, Austin, TX, USA, March 2-4, 2020*. 2020.

[186] Catharine M. Wyss, Chris Giannella, and Edward L. Robertson. "FastFDs: A Heuristic-Driven, Depth-First Algorithm for Mining Functional Dependencies from Relation Instances - Extended Abstract." In: *Data Warehousing and Knowledge Discovery, Third International Conference, DaWaK 2001, Munich, Germany, September 5-7, 2001, Proceedings*. Vol. 2114. Lecture Notes in Computer Science. Munich, Germany: Springer, 2001, pp. 101–110.

[187] Guanhui Yang, Xiaoting Chen, Tengsen Zhang, Shuo Wang, and Yun Yang. "An impact study of concept drift in federated learning." In: *2023 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2023, pp. 1457–1462.

[188] Hong Yao, Howard J. Hamilton, and Cory J. Butz. "FD_Mine: Discovering Functional Dependencies in a Database Using Equivalences." In: *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9-12 December 2002, Maebashi City, Japan*. Maebashi City, Japan: IEEE Computer Society, 2002, pp. 729–732.

[189] Shichao Zhang, Jilian Zhang, Xiaofeng Zhu, Yongsong Qin, and Chengqi Zhang. "Missing Value Imputation Based on Data Clustering." In: vol. 1. 2008, pp. 128–138.

[190] Zheng Zheng, Longtao Zheng, Morteza Alipourlangouri, Fei Chiang, Lukasz Golab, Jaroslaw Szlichta, and Sridevi Baskaran. "Contextual data cleaning with ontology functional dependencies." In: *ACM Journal of Data and Information Quality (JDIQ)* 14.3 (2022), pp. 1–26.

[191] Yifan Zhuang, Ruimin Ke, and Yinhai Wang. "Innovative method for traffic data imputation based on convolutional neural network." In: *IET Intelligent Transport Systems* 13.4 (2019), pp. 605–613.

La borsa di dottorato è stata finanziata dal D.M. n. 351 del 9.4.2022, a valere sul PNRR - finanziato dall'Unione europea -NextGenerationEU - Missione 4 "Istruzione e ricerca", Componente 1 "Potenziamento dell'offerta dei servizi di istruzione: dagli asili nido all'Università" - Investimento 4.1 "Estensione del numero di dottorati di ricerca e dottorati innovativi per la pubblica amministrazione e il patrimonio culturale"

