MiDaS: extract golden results from Knowledge Discovery even over incomplete Databases

Lucas S. Rodrigues, Thiago G. Vespa, Igor A. R. Eleutério,
Willian D. Oliveira, Agma J. M. Traina and Caetano Traina Jr.

Institute of Mathematics and Computer Sciences
University of São Paulo (USP), São Carlos, Brazil
{lucas_rodrigues}@usp.br, {caetano}@icmc.usp.br

Abstract. The continuous growth in data collection requires effective and efficient capabilities to support Knowledge Discovery in Databases (KDD) over large amounts of complex data. However, as activities such as data acquisition, cleaning, preparation, and recording may lead to incompleteness, impairing the KDD processes, specially because most analysis methods do not adequately handle missing data. To analyze complex data, such as performing similarity search or classification tasks, KDD processes require similarity assessment. However, incompleteness can disrupt the assessment evaluation, making the system unable to compare incomplete tuples. Therefore, incompleteness can render databases useless for knowledge extraction or, at best, dramatically reducing their usefulness. In this paper, we propose MiDaS, a framework based on a RDBMS system that offers tools to deal with missing data employing several strategies, making it possible to assess similarity over complex data, even in the presence of missing data at KDD scenarios. We show experimental results of analyses using MiDaS for similarity retrieval, classification, and clustering tasks over publicly available complex datasets, evaluating the quality and performance of several missing data treatments. The results highlight that MiDaS is well-suited for dealing with incompleteness enhancing data analysis in several KDD scenarios.

Keywords: Knowledge Discovery in Databases · Missing data · RDBMS Framework · Similarity Queries.

1 Introduction

The growing advances in data collection and organization demand effective and efficient capabilities to support Knowledge Discovery in Databases (KDD) processes over large amounts of complex data, such as data preprocessing, mining, and result evaluation. As more and more data are available for analysis, missingness increases as well. However, dealing with missing data is a challenging problem for KDD, as data preparation and analysis tools do not properly handle missingness.

Similarity queries are well-suited to retrieve complex data since comparisons based either on identity or ordering are mostly inappropriate for complex data.

Recently, new techniques have been widely explored for information retrieval, such as performing similarity comparisons over image collections to analyze and visualize Content-Based Image Retrieval (CBIR) results [18]. They are also being employed for classification, such as in Instance-Based Learning (IBL), to perform predictions based on data, mostly of them stored in Relational Database Management Systems (RDBMSs) [16,5]. Moreover, being able to express similarity queries in RDBMSs is becoming even more relevant for diverse applications that take into account the results of data mining in the KDD process.

Incompleteness problems can occur at any KDD step, including during data acquisition, data integration from different sources and as results of failures in phenomena observation and measurement [12]. Moreover, missingness can hamper data mining since similarity queries and other information retrieval techniques do not handle attributes with missing values, reducing the data available and thus the effectiveness of the KDD process.

Similarity queries over incomplete databases usually ignore attributes with missing values because distance functions cannot measure dissimilarity among incomplete objects. Figure 1 illustrates this problem with a dataset composed of medical exams of the brain, heart, and stomach. In this example, the heart and stomach exams with `NULL` for patient P_3 are missing data. The similarity between Patients P_1 and P_3 cannot be evaluated because most distance functions, such as Euclidean, cannot assess the similarity among those patients' records.

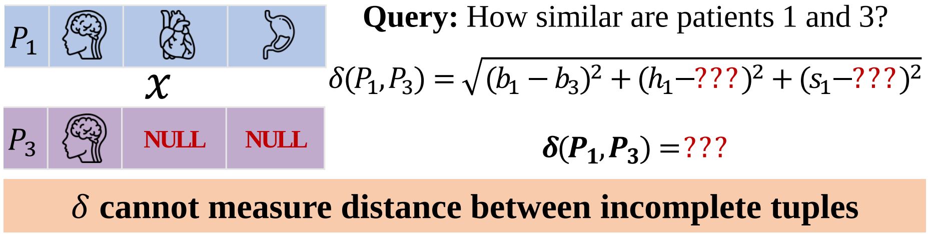

P_1
χ
P_3
NULL
NULL
Query: How similar are patients 1 and 3?
$\delta(P_1,P_3) = \sqrt{(b_1 - b_3)^2 + (h_1 - ???)^2 + (s_1 - ???)^2}$
$\boldsymbol{\delta(P_1,P_3)} = ???$
δ cannot measure distance between incomplete tuples

Fig. 1: Similarity searches over data with missing values.

Existing techniques to handle missing data include deleting incomplete tuples or imputing probable values for each missing attribute. Over the years, imputation methods have explored diverse heuristics to infer values, including the global mean, an average of the neighborhood, and regression models [13,17,4]. Recently, an approach dealing with this problem took advantage of the intrinsic information embedded in the data, using correlations among the similarity of near neighbors to weight similarity queries over incomplete databases, well-fitting the data distribution, and improving the results [18].

In recent years, several works have explored functionalities to support KDD tasks for Content Based Retrieval Systems (CBRSs) [14,1,16]. However, they do not consider the missing data problem in the RDBMS environment, thus rendering data mining impracticable over incomplete data collections. Here we aim at including additional functionalities into an RDBMSs to handle missing

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08760-8_54

data for KDD scenarios, to support information retrieval scenarios, especially for executing similarity queries and to help classification or clustering tasks.

We introduce `MiDaS`: the ***M**issing **D**ata in **S**imilarity* retrieval framework. It is an RDBMS-centered framework that provides tools based on well-suited strategies to deal with missing data for similarity comparisons and KDD tasks. `MiDaS` comprises two layers of functionalities to handle missing data and support similarity search. The missing data treatment layer exports functionalities to handle incompleteness based on six reliable, long-used approaches proposed in the literature. The query engine layer provides similarity retrieval operators suitable for queries over complete and incomplete databases. Therefore, now that information is the "new gold" for companies, `MiDaS` incorporates tools for data preparation and analysis into the RDBMS environment, allowing the use of all available data to mine the "gold" right there where the data is stored. Besides, we intend to contribute to KDD issues, providing an efficient way to deal with incompleteness in a single environment and support information retrieval tasks with quality and performance. The main contributions of this work are as follows:

- Provide a "missing data engine", including strategies for tuple deletion, several relevant imputation heuristics, and a recent approach based on data correlation that neither discard nor infer values.
- Support k-Nearest Neighbors (k-NN) and Range similarity retrieval operators even over incomplete data scenarios.
- Conduct a performance analysis of the `MiDaS` Framework, to reveal the efficiency when treating several missing data rates in complex databases.
- Present a thorough experimental analysis of the `MiDaS` strategies to show the advantages and applicabilities when the framework handles missing data for KDD scenarios, including similarity retrieval, classification based on neighborhood, and clustering analysis.

The remainder of this paper is organized as follows. Section 2 presents the relevant background and works that explore similarity searches and missing data. Section 3 presents `MiDaS` and Section 4 shows its experimental analyses. Finally, Section 5 concludes the paper.

2 Background and Related Work

Comparing by similarity is the usual way to query and retrieve complex data. Here we briefly discuss related concepts concerning this work.

We call "complex" an attribute that may be compared under varied "aspects", so it must be previously defined, for example, using a distance function over certain features of the objects. Similarity queries perform comparisons between complex objects using low-level representations called feature vectors, obtained by Feature Extraction Methods (FEMs), and stored in RDBMSs. For example, there are several suitable FEMs to process images, based on features such as color, texture, shape, or on learning approaches, such as Histogram Oriented Gradients (HOG), Haralick, and VGG16 [7,19]. We assume that $\mathbb{S}$ is a domain

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08760-8_54

of features already extracted and stored in attribute S of a RDBMS relation and S as the active domain of that attribute. Thus, S is the set of values stored, which are compared by distance functions in the similarity queries. A distance functions (δ) assess the dissimilarity between two complex objects (s_i, s_j) and return a real value from $\mathbb{R}^+$. There are several distance functions, such as Euclidean, Manhattan, Chebyshev, and others [9], and some are called a "metric" when it meets the Metric Spaces properties.

Each similarity query, either a Range or a k-Nearest Neighbors query (k-NN), is posed over a complex dataset $S \subseteq \mathbb{S}$ and requires a central object $s_q \in \mathbb{S}$ called the query center, a distance function δ, and a similarity limit. A Range Query (R_q) retrieves every tuple where $s_i \in S$ whose distance to s_q is less or equal than the limit given by a similarity radius (ξ), measured as $\delta(s_q, s_i) \leqslant \xi$. A k-NN Query (Knn_q) retrieves the tuples where s_i is one of the k nearest to s_q, where the limit is the amount k of tuples retrieved.

2.1 Missing Data and Treatments

The missing data problem in RDBMS occurs when a complex value or some of its components is NULL (e.g., when s_i is an array). Incompleteness negatively affects the quality of the data and, consequently, the experts' analyses.

Several works explore viable solutions to the problem. **Deletion methods** drop tuples or attributes with missing values. However, these methods are widely questionable, as they reduce the data available for analyses and may lose relevant information [18]. **Imputation methods** explore statistical heuristics, such as imputation based on the mean value of an attribute or exploring regression models to predict probable values [13]. Machine Learning heuristics perform imputation based on the element neighborhood, searching for the k nearest neighbors to infer missing values [4]. Imputation methods based on decision trees identify natural partitions on the data or use supervised classification algorithms to infer probable values [17].

A recent alternative for both deletion and imputation, called **SOLID** [18], exploits correlations among attributes to handle incompleteness. It uses existing values to identify pairwise attribute correlations to weigh their contribution to the similarity assessment. Hence, **SOLID** takes advantage of correlations to execute weighted queries over an incomplete dataset, employing all the data available, thus avoiding either discarding tuples or imputing values.

2.2 Content-Based Retrieval Systems

Several Content-Based Retrieval Systems (CBRSs) based on RDBMS were proposed to perform information retrieval at diverse scenarios and applications [14,16,5,11,3]. Table 1 summarizes the most relevant related works, regarding the following aspects:

- **Open-Source RDBMS**: whether it is based on Open-Source RDBMS, which brings flexibility and eases incorporating new techniques and algorithms;

- **Similarity-Retrieval Tasks**: whether it provides mechanisms for information retrieval over complex datasets (range and k-NN comparisons);
- **Handle Missing Data**: whether it supports or applies strategies to deal with incompleteness, including data cleaning and preparation aiming at KDD;
- **Self-Contained Functions**: whether it seamlessly integrates with the RDBMS architecture without requiring external interactions or add-ons tools, easing user customization.

Table 1: Related works and our proposed `MiDaS` according to relevant aspects.

Work	Year	Open-Source RDBMS	Similarity Retrieval	Missing Data	Self-Contained
SIREN [3]	2006	✗	✓	✗	✗
FMI-SiR [11]	2010	✗	✓	✗	✗
SimbA [5]	2014	✓	✓	✗	✗
Kiara [16]	2016	✓	✓	✗	✗
SimAOP [1]	2016	✓	✓	✗	✗
MSQL [14]	2017	✓	✓	✗	✗
`MiDaS`	2022	✓	✓	✓	✓

Open-Source RDBMS. Most of the related works are based on open-source architectures, mainly on PostgreSQL [14,1,16,5]. Open-Source RDBMS increases availability and makes it easier for the user to tune its functionalities or to include new algorithms and procedures. However, [3] and [11] were developed using Oracle and its specific tools. Both suggest it can be extended to PostgreSQL but do not describe how. We choose PostgreSQL because it enhances transparency and applicability for missing data management and similarity query execution.

Similarity-Retrieval. Many of the works support mechanisms to execute similarity queries in an RDBMS, highlighting the relevancy and suitability of similarity retrieval over complex data. They aim at extending existing retrieval operators with new abilities, either modifying the RDBMS core and other modules [11] or creating other relational operators using plain SQL [3,5,16,14], to allow posing both range and k-NN queries. We followed the Kiara concept [16] as the underpinning for the similarity operations in our framework, extending it to handle missing data. Our solution is adaptable to other similarity retrieval frameworks, as it does not depend on a specific RDBMS architecture.

Handle Missing Data. None of the previous works that explore similarity retrieval on RDBMSs provide mechanisms to handle missing data. In fact, they just discard incomplete tuples, losing potentially helpful information and knowledge from the databases. `MiDaS` provides suitable strategies to infer missing data to enable similarity queries over incomplete data, mainly using the **SOLID** approach, which improves flexibility and the quality of the analysis.

Self-Contained Functions. The previous works only provide functionality for the RDBMS environment to execute similarity queries, but none of them

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08760-8_54

deals with incompleteness. `MiDaS` can help with both missing-aware data preparation and similarity retrieval, providing a novel environment that the user can customize using strategies to deal with missing data in KDD processes.

3 The `MiDaS` Framework

The `MiDaS` **Framework** incorporates tools to handle incompleteness in complex data databases and support knowledge retrieval based on similarity queries. It aids KDD processes, particularly data preparation and data analysis combined with the functionalities of an RDBMS. Currently, `MiDaS` is implemented in PostgreSQL[1]. As shown in Figure 2, the `MiDaS` architecture is composed of two layers:

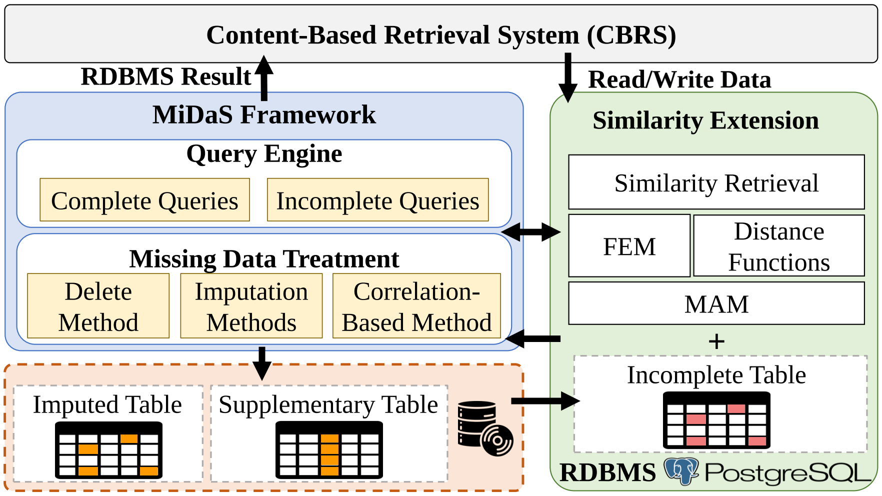


Fig. 2: Overview of `MiDaS` Framework.

- **Missing Data Treatment Layer**: provides six reliable approaches for data preparation, with emphasis on complex datasets and applications for data mining scenarios.
- **Query Engine Layer**: implements the k-NN and Range query operators, enabling similarity queries over complete and incomplete data, providing tools for information retrieval at varying applications.

3.1 The Missing Data Treatment Layer

This layer is able to handle missing data through Missing Data Treatment (MDT) modules, which can be helpful for a variety of applications. `MiDaS` includes six MDT, covering the main approaches available from the literature: one

[1] PostgreSQL: Available at: `https://www.postgresql.org/`. Access date: 09 Fev. 2022.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08760-8_54

based on tuple deletion, four on imputation following varying heuristics, and one on attribute correlation, described as follows.

The **(i) Deletion** MDT just ignores the tuples with missing values without changing the original table. The imputation MDTs assign a value for each missing value. There are four options, as follows: **(ii) Mean Imputation (MI)**: fills the missing values using the mean of the existing values of the attribute in the entire table. **(iii)** ***k*****-NN Imputation (kNNI)**: fills the missing values using the average value of the k-Nearest Neighbors of each tuple, where k is defined by the user. **(iv) Regression Imputation (RI)**: predicts missing values based on linear regression using other attributes and the observed values of the attribute. **(v) Decision-Tree Imputation (DTI)**: identifies data partitions based on the similarity between elements and applies a classification algorithm to infer a probable value for each missing case. Each imputation MDT receives an incomplete table and the required user-defined parameters and returns the imputed table. The user chooses how to process the output further, either updating the original table or storing the output as a temporary table.

Finally, **(vi) SOLID MDT** evaluates pairwise correlations among selected attributes. SOLID receives the incomplete table and the *correlation threshold* as a real-valued parameter and discards the attributes with low correlations just for each pairwise comparison. Hence, SOLID generates a supplementary table (**"weights_tableName"**) with two attributes: the compatible attribute pair and the corresponding weights, as explained in Section 2. As with the other MDTs, SOLID does not update the input table. Every MDT can be activated using either PL/Python or SQL statements with storing functions/procedures into the RDBMS.

`MiDaS` executes the MDTs in either `Pre-Computation` or `Query Run-time` mode, bringing flexibility to the user when choosing how to treat and prepare the incomplete datasets. In `Pre-Computation` mode, an MDT is processed, and the result is saved as a new table in the RDBMS as the imputed or supplementary table. When similarity queries are posed, this table is used instead of the original. On the other hand, execution in `Query Run-time` mode re-executes the MDT method at each query execution. `MiDaS` covers most of the missing data handling needs in various scenarios, producing data more suitable for KDD processes improving the quality of data analysis. Moreover, as the `MiDaS` is Open-Source and based on an open RDBMS, other MDTs can be included.

3.2 The Query Engine Layer

Similarity queries over either complete or incomplete datasets require appropriate retrieval operators, and this topic is especially relevant for RDBMSs. Current RDBMSs do not support those operators, although similarity queries may be expressed in SQL combining existing relational operators and functions. However, considering all the variations involved, especially when missing data must be handled, expressing similarity queries can be troublesome. Thus, the `MiDaS` was also developed to provide resources to express similarity queries, implementing the Query Engine layer to provide similarity retrieval operators. It executes both

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08760-8_54

Range and k-NN queries over complete and incomplete data using any of the provided MDTs.

We provide the similarity retrieval operators based on the concept that the result of a (sub-)query over a database relation is also a relation. Hence, we implement each operator as an SQL function that returns a table, callable at the FROM clause of a SELECT command. Statement 1.1 presents the general syntax of the command to pose a similarity query over complete or incomplete tables. It can be integrated with the elements of the SQL language, such as referring to other tables and clauses of the select statement and other RDBMS commands. As the result, the `MiDaS` is a robust and practical similarity engine for complex data retrieval, able to handle complete and incomplete data.

```sql
SELECT queryResult.* FROM simQueryWithMissing(
    table_name    anyelement,    -- input table
    mdt_method    VARCHAR,       -- MDT method
    sim_operator  VARCHAR,       -- knn or range
    sim_criterion NUMERIC,       -- k or radius value
    obj_query     COMPLEX_DATA,  -- query center
    dist_func     VARCHAR        -- distance function
    ) as queryResult;
```

Statement 1.1: Similarity Query with MiDaS Framework.

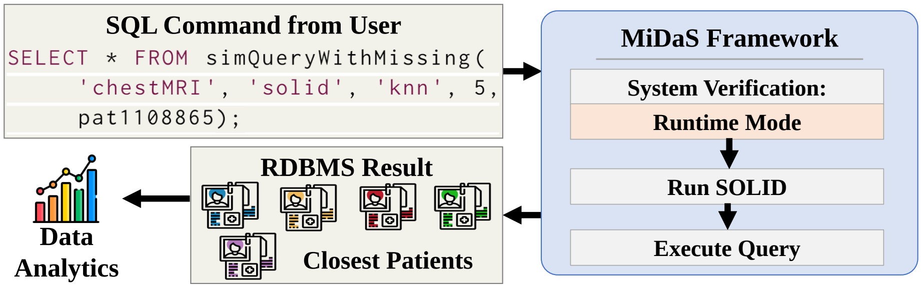
The `simQueryWithMissing` function requires the following parameters:

- `table_name`: the table where the similarity search must be performed;
- `mdt_method`: the MDT method, choosing `deletion`, `mean`, `knn`, `regression`, `dt` or `solid` to handle missing data, or `NULL` for complete data;
- `sim_operator`: the similarity comparison operator: specify `knn` or `range`;
- `sim_criterion`: the similarity limit: the k for `knn` queries or radius for `range`;
- `obj_query`: the query center;
- `dist_function` the distance function. `MiDaS` Framework provides `L1` to Manhattan, `L2` to Euclidean, or `Linf` to Chebyshev. Also, the user can define other distance functions as stored functions.

Execution in `Pre-Computation` or `Query Run-time` mode is transparent for the query layer, as when the table for a Pre-Computation MDT exists, the query is executed using the pre-processed table to speed up execution. Figure 3 illustrates the query execution that retrieves the five patients closest to patient `pat110886`, using the SOLID approach as the MDT method over the table of MRI chest exams stored in table `chestMRI`, using the `L2` distance function. When the weights table for SOLID is not already prepared, `MiDaS` automatically chooses the `Query Run-time` mode, calls the MDT to generate the table with the default parameters, and stores it in the RDBMS, to speed up further queries. When the table is already computed, then the query is executed in `Pre-Computation` mode.

4 Experimental Analysis

This section describes the datasets used in the experiments, how the experiments were evaluated, the analysis of their performances, and how `MiDaS` can be



```sql
SELECT * FROM simQueryWithMissing(
     'chestMRI', 'solid', 'knn', 5,
     pat1108865);
```

Fig. 3: Query execution example with `MiDaS` Framework.

applied for the KDD process, including information retrieval, classification, and clustering tasks, highlighting the framework applicability.

4.1 Datasets and Implementation Specifications

The experiments were performed using four datasets of images, including medical images, objects, and a font types collection. Table 2 shows their number of tuples (n), attributes (d), and a brief description.

Table 2: Image datasets used in this work.

Dataset	n	d	**Description**
ds-Spine [6]	54	10	Lumbar muscles and vertebral bodies MRI
ds-Coil [15]	100	72	Objects posing over 360 rotation, with 5 interval degrees
ds-Letters [10]	152	52	Font types of alphabetic in lower and upper cases letters
ds-Brats [2]	1251	5	Glioma MRI scans with pathologically confirmed diagnosis

Data collected and processed in January 25, 2022

Every tuple in a relation may have missing values. For example, each attribute of the dataset *ds-Letters* is the image of a letter in a font, each font having d =52 letters, but some letters may be missing for a specific font. We extracted features from the four image datasets using five FEMs: Local Binary Patterns (LBP), Zernike, Haralick, Histogram of Oriented Gradients (HOG), and VGG16 (a CNN model) [7,19]

`MiDaS` was implemented using PostgreSQL 13.2, incorporating PL/Python scripts and the well-known open libraries Pandas, Numpy, Scipy, Sklearn, and others, for missing data treatment. We used the C++ language for operations of bulk-loading features from complex collections and SQL scripts for distances functions and similarity retrieval operations. We create the scripts as `MiDaS` RDBMS, developing the imputations heuristics based on models from Sklearn and using SOLID scripts from the repository of [18]. Scripts and more details about data management are available in a Git repository[2], which also provides the extracted features.

4.2 MiDaS Performance

This experiment evaluates MiDaS performance running each MDT for k-NN and Range queries in both runtime modes. Each query was executed 50 times, randomly changing the query center at each execution over the datasets with up to 50% of randomly missing values. We setup both queries with no preprocessing and MDT=SOLID, changing parameters sim_operator and defining sim_criterion with k = 21 and range = 0.55. Changing each query for each MDT just requires modifying the mdt_method parameter. Tables 3 and 4 present the average wall-clock time (in seconds), showing the Query Run Time (QT columns) and the MDT execution time, concerning the Pre-Computation (PCM columns).

Table 3: k-NN queries elapsed time (in seconds) of each MDT in 50% missingness.

Dataset	**Deletion**		**SOLID**		**MI**		**kNNI**		**RI**		**DTI**	
	QT	PCM	QT	PCM	QT	PCM	QT	PCM	QT	PCM	QT	PCM
ds-Spine	0.04	–	0.11	0.70	0.24	2.10	0.22	4.56	0.22	9.06	0.20	42.54
ds-Coil	0.18	–	0.43	18.66	0.76	0.58	0.84	0.74	0.66	6.32	0.74	15.74
ds-Brats	0.16	–	0.14	6.14	0.64	0.44	0.62	0.58	0.60	1.10	0.62	2.46
ds-Letters	0.12	–	0.42	14.72	0.76	0.52	0.78	0.84	0.78	5.46	0.76	12.84

Table 4: Range queries elapsed time (in seconds) of each MDT in 50% missingness.

Dataset	**Deletion**		**SOLID**		**MI**		**kNNI**		**RI**		**DTI**	
	QT	PCM	QT	PCM	QT	PCM	QT	PCM	QT	PCM	QT	PCM
ds-Spine	0.04	–	0.11	0.70	0.18	2.10	0.24	4.56	0.20	9.06	0.26	42.48
ds-Coil	0.18	–	0.43	18.66	0.72	0.58	0.74	0.84	0.68	6.32	0.43	15.70
ds-Brats	0.08	–	0.15	6.14	0.60	0.44	0.64	0.58	0.60	1.10	0.68	2.46
ds-Letters	0.18	–	0.72	14.72	0.74	0.52	0.76	0.84	0.78	5.46	0.72	12.84

Notice that only the QT time is spent in Pre-Computation mode (PCM), whereas both (QT+PCM values) times are spent in Query Runtime mode (QT). Overall, the query time of the Pre-Computation mode varies from 0.04 to 0.78 seconds for k-NN and Range queries. The Query runtime mode adds the time to execute SOLID or an imputation MDT, leading to a total time-varying from 0.44 up to 42.54 seconds (PCM columns). For other missing rates, the queries give similar times.

These experiments show that MiDaS achieves high performance in both modes, allowing a quick comparison between the various MDT approaches. We highlight that Pre-Computation mode is better suited for scenarios where frequent queries are posed over datasets that undergo few or no updates because they only require

[2] Git repository of MiDaS: https://github.com/lsrusp/MiDaS.

the MDT method to be executed once. The `Query Runtime` mode is better suited for fast-changing datasets because it allows the MDT to track data evolution.

4.3 `MiDaS` for Knowledge Discovery on Databases

We show the applicability of `MiDaS` MDT approaches for distinguished scenarios where missingness can cause hardness or even make tasks impracticable for KDD. Therefore, we present MDTs analysis, highlighting the quality and advantages for similarity retrieval, classification, and clustering scenarios.

Similarity Retrieval. We analyzed SOLID MDT results to deal with incompleteness using dataset *ds-Spine*, exploiting the advantages of weighted queries based on correlation data. Figure 4 shows its tuples distribution using the MDS Projection [8], making it possible to visualize the multidimensional space distribution of elements using the distance matrix. The distance matrix is based on the Euclidean distances between tuples over (a) complete data, (b) 10% missingness, (c) posing SOLID for 10%, (d) 50% missing rate, and (e) SOLID execution for 50% missing, respectively for each case. Blue points are complete tuples, red are incomplete tuples, and black points are the SOLID application. We highlight that the SOLID takes advantage of the correlation between complex attributes to any missingness scenarios, well-fitting the data distribution to allow posing queries that achieve high-quality results even at large amounts of missingness.

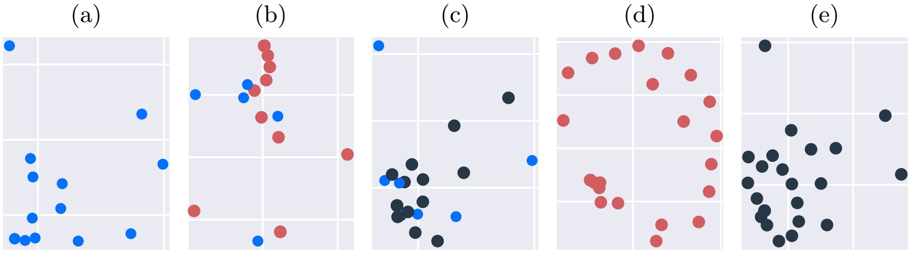


Fig. 4: Similarity retrieval over *ds-Spine* in scenarios of (a) complete data, (b) 10% missingness, (c) SOLID for 10%, (d) 50% missing rate and (e) SOLID execution for 50% missing. Blue points are tuples complete, red are incomplete ones, and black points are evaluation results from SOLID.

Classification Scenarios. We analyzed the instance classification based on a k-NN Classifier model using `MiDaS` imputations based on Mean, k-NN, and Regression for cases of 50% missing rate. The k-NN Classifier employs Euclidean distance to measure dissimilarity. This experiment shows how a treated dataset can be better suited to employing our framework for supervised tasks since untreated missingness can bias classification. In this task, we classified the elements from the *ds-Letters* dataset using the font type as the “class” of the tuple, such as “Times New Roman”, “Verdana”, etc. We split the train/test sets at 50% each, changing k between 3 to 21, and executed the evaluation using F1-Measure, Precision, and Recall, as shown in Figure 5. The results show that the employed imputation methods were well-suited for classification scenarios, resulting in high values up to 0.89, 0.89, and 0.90 for F1-Measure, Precision, and Recall, respectively. Therefore, the `MiDaS` can be applied and is fast enough in practice to support missingness treatment for labeled datasets, such as *ds-Letters* (see Section 4.2).

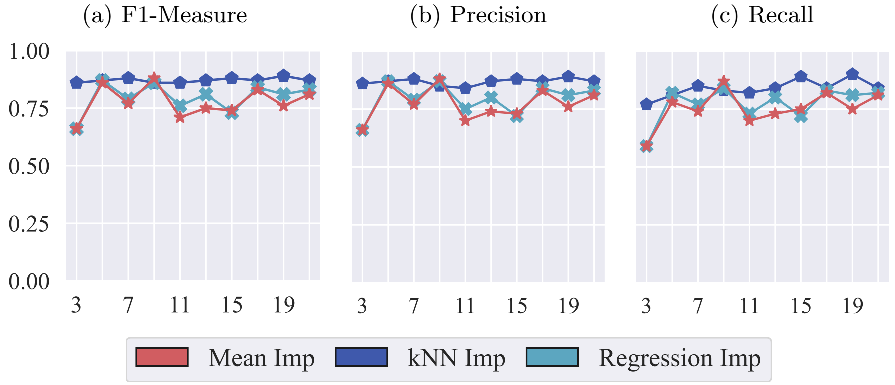


Fig. 5: Classification evaluation for *ds-Letters* dataset using F1-Measure, Precision, and Recall of k-NN Classifier.

Clustering Patient Analysis. We analyzed the missing data distortion in the clustering scenario using *ds-Brats*, a medical data collection of patients with several exam images. We intend to discover patients based on the similarity of the exams distribution but having 50% of incomplete exams. In Figure 6, we show the application of the KMeans technique over (a) only complete patients, (b) deletion of incomplete patients, and (c) using the Decision Tree Imputation (DTI) to impute values for the tuples that were deleted in (b) case. As seen in (b) with only complete tuples, missingness can be highly harmful to clustering because the clusters' distortions blur the patient group's discovery. (c) DTI takes advantage of the data partitions based on highly similar objects and aid in reducing the clusters distortions, spending just a few seconds to deal with missingness (see Tables 3 and 4). Each distinguished color in Figure 6 represents a patients' cluster. We evaluate the clustering distribution in (a), (b), and (c) using the silhouette metric (the closer to one, the better the clusters), resulting in 0.473, 0.204, and 0.571, respectively, highlighting that missingness treatments are advantageous and improve clustering tasks. Hence, the `MiDaS` makes it possible to discover patient groups and enables, for example, the behavior analyses of each patient group to support the physician's decisions using the most helpful information.

As shown in the varied scenarios evaluated in this section, we highlight that the `MiDaS` provides many methods well-suited to handle data incompleteness and that it is very useful for several KDD scenarios. Therefore, this Section supports our assertion that the `MiDaS` provides a simple but effective approach to dealing

DOI: 10.1007/978-3-031-08760-8_54

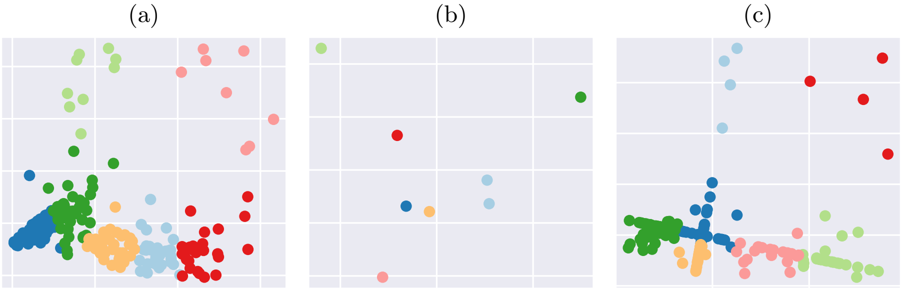


Fig. 6: Clustering analysis for KMeans application over *ds-Brats* in scenarios over (a) only complete patient data, (b) deletion of 50% incomplete patient records, and (c) use Decision Tree Imputation (DTI), focusing on the missing impact and the DTI advantages to impute values for tuples deleted case in (b).

with incompleteness, leading to obtain high-quality results improving analytics by extracting golden insights from the data in a timely manner.

5 Conclusions

In this work, we presented `MiDaS`: the ***Mi**ssing **Da**ta in **S**imilarity* retrieval framework, implemented as an extension of the PostgreSQL RDBMS that is able to provide well-suited treatments for missing data, enabling analysis tasks for Knowledge Discovery in Databases. `MiDaS` aims to meet the requirements of KDD tasks by embedding proper tools into an RDBMS and bringing data processing directly to where the data is stored. It provides six MDT methods well-suited for dealing with missing data and provides features for similarity retrieval that do not discard nor impute values and for performing classification and clustering tasks. For similarity retrieval, it provides query operators that are able to handle both complete and incomplete datasets, seamlessly integrated with the other features of the RDBMS. Therefore, we claim that `MiDaS` is a novel and practical environment for users to handle missingness using the provided MDTs, which allows performing KDD analyses integrated to execute similarity queries over complete or incomplete data. The experimental evaluation shows that `MiDaS` provides strong resources to deal with incompleteness in large databases, increasing the quality of results from various KDD scenarios and the efficiency of similarity query execution.

Acknowledgements

This research was supported in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil - Finance Code 001, by the São Paulo Research Foundation (FAPESP, grants No. 2016/17078-0, 2020/07200-9, 2020/10902-5), and the National Council for Scientific and Technological Development (CNPq).

References

1. Al Marri, W.J., et al.: The similarity-aware relational database set operators. Information Systems **59**, 79–93 (2016). https://doi.org/10.1016/j.is.2015.10.008
2. Baid, U., et al.: The rsna-asnr-miccai brats 2021 benchmark on brain tumor segmentation and radiogenomic classification. arXiv preprint arXiv:2107.02314 (2021)
3. Barioni, M.C.N., et al.: SIREN: A similarity retrieval engine for complex data. In: The 32nd International Conference on VLDB. pp. 1155–1158. ACM (2006)
4. Batista, G., et al.: A study of k-nearest neighbour as an imputation method. In: Soft Computing Systems - Design, Management and Applications. Frontiers in Artificial Intelligence and Applications, vol. 87, pp. 251–260. IOS Press (2002)
5. Bedo, M.V.N., Traina, A.J.M., Traina-Jr., C.: Seamless integration of distance functions and feature vectors for similarity-queries processing. JIDM **5**(3), 308–320 (2014), https://periodicos.ufmg.br/index.php/jidm/article/view/276
6. Burian, E., et al.: Lumbar muscle and vertebral bodies segmentation of chemical shift encoding-based water-fat mri. BMC Musculoskeletal Disorders **20**(1), 1–7 (2019), 10.1186/s12891-019-2528-x
7. Coelho, L.P.: Mahotas: Open source software for scriptable computer vision. arXiv preprint arXiv:1211.4907 (2012)
8. Cox, M.A., Cox, T.F.: Multidimensional scaling. In: Handbook of data visualization, pp. 315–347. Springer (2008)
9. Deza, M.M., Deza, E.: Encyclopedia of distances. In: Encyclopedia of distances, pp. 1–583. Springer (2009). https://doi.org/10.1007/978-3-642-30958-8
10. Hajder, S.: Letters organized by typefaces - standard windows fonts with each letters organized in classes by typeface (2020), https://www.kaggle.com/killen/bw-font-typefaces
11. Kaster, D.S., et al.: Fmi-sir : A flexible and efficient module for similarity searching on oracle database. Journal of Information and Data Management **1**(2), 229–244 (2010), https://periodicos.ufmg.br/index.php/jidm/article/view/36
12. Lei, C., et al.: Expanding query answers on medical knowledge bases. In: The 23rd International Conference on Extending Database Technology. pp. 567–578. OpenProceedings.org (2020). https://doi.org/10.5441/002/edbt.2020.67
13. Little, R.J., Rubin, D.B.: Statistical analysis with missing data, vol. 793. John Wiley & Sons (2019). https://doi.org/10.1002/9781119482260
14. Lu, W., et al.: Msql: Efficient similarity search in metric spaces using sql. The VLDB Journal **26**(6) (2017). https://doi.org/10.1007/s00778-017-0481-6
15. Nene, S.A., et al.: Columbia object image library (coil-100). Tech. rep., Department of Computer Science, Columbia University, New York, USA (1996)
16. Oliveira, P.H., et al.: On the support of a similarity-enabled relational database management system in civilian crisis situations. In: ICEIS 2016 - Proceedings of the 18th International Conference on Enterprise Information Systems. pp. 119–126. SciTePress (2016). https://doi.org/10.5220/0005816701190126
17. Rahman, M.G., Islam, M.Z.: Missing value imputation using decision trees and decision forests by splitting and merging records: Two novel techniques. Knowledge-Based Systems **53** (2013). https://doi.org/10.1016/j.knosys.2013.08.023
18. Rodrigues, L.S., et al.: Taking advantage of highly-correlated attributes in similarity queries with missing values. In: 13th International Conference on Similarity Search and Applications, SISAP 2020. LNCS, vol. 12440, pp. 168–176. Springer (2020). https://doi.org/10.1007/978-3-030-60936-8_13
19. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08760-8_54