POLITECNICO MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE

Executive Summary of the Thesis

A knowledge-driven approach for supporting data preparation

Laurea Magistrale in Computer Science and Engineering - Ingegneria Informatica

Author: Enrico Staiano

Advisor: Prof. Cinzia Cappiello

Co-advisor: Camilla Sancricca

Academic year: 2022-2023

1. Introduction

Nowadays, organizations increasingly rely on collecting and analyzing large volumes of data to support their business decisions: data-driven management has become central. However, the success of the analyses and decisions based on data depends greatly on the quality of the data itself. Working with poor quality data will lead to flawed or unreliable results [1].

Therefore, an effective data preparation phase aimed to improve data quality is fundamental for the success of any analysis to be performed on the data. Data preparation is a long and complex process, involving a large variety of techniques and issues, and it has been demonstrated that can take up to 80% of the work of a data scientist [4]. Moreover, depending on the available data and on the desired analyses to perform, it has been shown that different data preparation pipelines may be suitable [2]. For these reasons, it may be hard for a user, especially for a non-expert, to navigate through the heterogeneous, long path of data preparation [3].

This work addresses this issue by proposing an approach that supports users, guiding them through the preparation process and offering suggestions tailored for their needs.

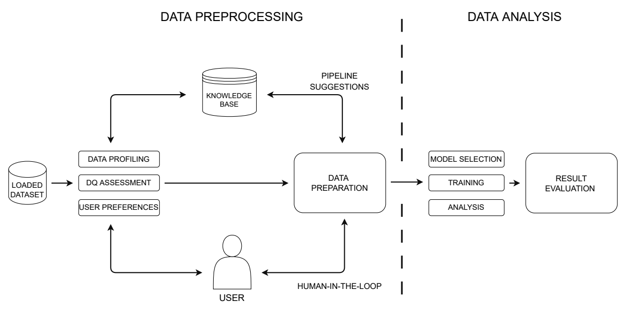

Figure 1: Architecture

2. Methodology

The methodology presented in this work aims to guide users through data preparation and help them select the best actions to satisfy their specific needs. This methodology considers the data on which the users are working and the kind of analysis they want to perform, and suggests a data preparation pipeline suitable for conducting a reliable analysis in that context. The high-level representation of the architecture used in this methodology is shown in Figure 1 (this work focuses on the Data Preprocessing section of the schema).

The architecture relies on a Knowledge Base (KB), which contains all the information needed to support the data preparation process. A detailed description of the KB can be found in Section 3.

At the beginning of the process, the users load the Dataset they want to analyze and select a Machine Learning (ML) Application they want to execute as goal of their analysis. Once the dataset is loaded, its Data Profile is computed. The Data Profile contains a set of characteristics of the dataset (e.g., number of tuples, number of attributes, percentage of missing values, etc.) called Data Profile Features, which can be useful during the preparation phase.

At this point, the actual data preparation phase begins. The data preparation phase can be divided in several segments, each of them focusing on a different set of actions. For each Data Quality Dimension considered, there is a segment that focuses on improving that dimension; moreover, there is an additional segment (called "ML Application-oriented" segment) that contains actions specifically aimed to improve the execution of the ML Application. In each segment, a list of possible actions are proposed, called Data Preparation Techniques. Often, a technique can be implemented in practice in multiple ways, called Data Preparation Methods. Some techniques (or specific methods) can be applied only if some conditions on the Data Profile Features of the dataset are verified. Before proposing actions to the users, these conditions are checked, and only the techniques/methods that pass these checks are actually suggested. Considering the ML Application-oriented segment, the techniques proposed in this segment are specifically selected for the ML Application chosen at the beginning by the users.

Furthermore, every time a preparation technique is selected, the architecture can give the user an indication about the best method to implement that technique with, considering the user's specific context (dataset + ML application selected). To achieve this last objective, the approach adopted in this work relies on the utilization of classifiers. The details of this approach will be explained in Section 5. Because of the described architecture functionalities, the suggestions made to the users are customized to their needs and goals.

This architecture uses a Human-In-The-Loop approach: users are constantly involved in the entire data preparation phase at various levels. Complete freedom is always guaranteed for the users: they are free to reject the received suggestions and can also choose independently the data preparation pipeline to perform. Moreover, the users are involved also at execution time: during the execution of some techniques, the architecture can ask the users feedback and input, if needed. The users' choices, which may be different from the actions suggested, are stored in the KB. In this way, it is possible to create a history of users' preferences, with the aim of understanding the most common choices and their differences from the suggested actions. This data can also be leveraged, in the future, to adjust the suggestions and align them with users' past choices in similar contexts.

Note that the Knowledge Base plays a fundamental role in the entire process, from data profiling until the end of data preparation. All the concepts discussed and the relationships between them are stored in the KB and retrieved when needed. During the data preparation phase, the KB is continuously interrogated to propose the appropriate techniques and methods for the specific context of the users.

3. Knowledge Base Conceptual Design

The conceptual schema of the KB designed in this work is shown in Figure 2. In this section, the contents of the KB are listed and briefly described.

A ***Data Object*** (*do*) represents a set of data loaded by the users. A *do* can represent a Dataset or a Dataset Column. A ***Data Profile Feature*** (*dpf*) indicates a characteristic that a data object can have. As evident in the conceptual schema, the value that a feature takes in the specific case of a certain data object is not a property of the feature alone, but it is a property of the relationship between the object and the feature. A ***Data Profiling Technique*** (*dpft*) is a technique that performs the profiling of data objects and returns in output the values of some profile features. A ***Data Preparation Technique*** (*dpt*) is a possible technique that the users can apply to their data. Each *dpt* has a granularity of application, that indicates whether this technique must be applied on a whole dataset, on a single column, or can be applied in both cases. A ***Data Quality Di-***

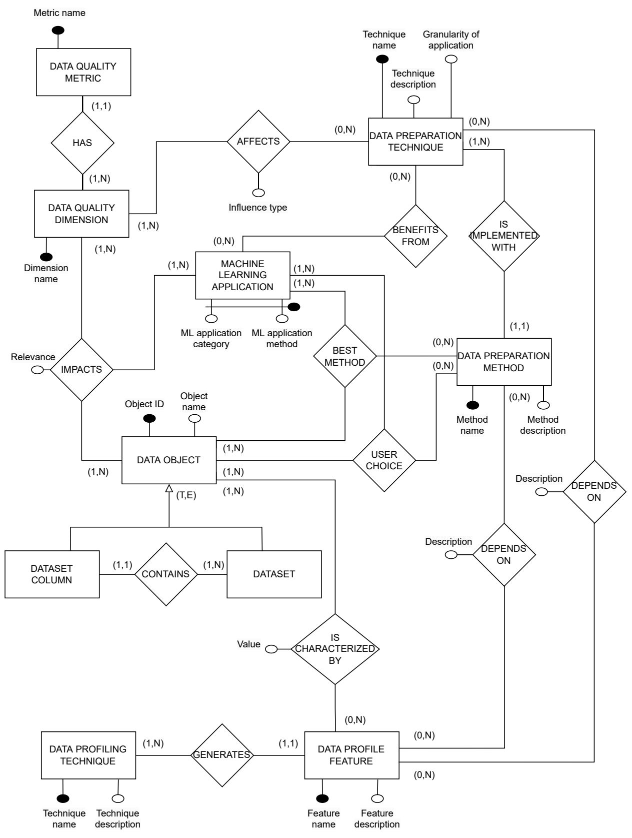

Figure 2: Knowledge Base Conceptual Schema

mension (*dqd*) is a concept that captures a specific data quality aspect. A *dpt* can affect one or more *dqd* (either in positive or in negative ways). A ***Data Quality Metric*** (*dqm*) expresses how a data quality dimension can be assessed. Every *dqd* has one or more *dqm*. A ***Machine Learning Application*** (*mla*) is an application that the users can choose as their goal analysis. Every *mla* is characterized by a macrocategory, which indicates the kind of ML application considered (e.g., clustering, classification or regression), and by a specific method that actually implements that application. A *mla* can benefit from zero, one or more *dpt*. A ***Data Preparation Method*** (*dpm*) indicates a possible way in which a preparation technique can be performed in practice. Each *dpt* is implemented with one or more *dpm*.

A data preparation technique or method can depend on one or more data profile features: this means that the execution of that technique or method can only take place if the considered feature meets a certain condition (which is stored in the "Description" attribute of the "depends on" relationships).

The ternary relationship Impacts is needed to store knowledge about the impact that each data quality dimension has with a certain ML application and Data Object. The "Relevance" property indicates the importance of that dimension in that context. The ternary relationship Best Method, instead, is used to store knowledge about which preparation method is best for implementing a certain technique in a given context. In particular, in this work, experiments were conducted to generate knowledge regarding the best Imputation methods to use in different contexts (details in Section 5). Lastly, the relationship User Choice allows to record what methods were actually selected by the users. This knowledge is useful to build the history of the choices made by past users in various contexts.

4. Knowledge Base Implementation

This section describes how the KB presented in Section 3 was implemented. To implement the KB it was decided to use a graph database, in particular, Neo4j. To store data, Neo4j uses nodes and edges (called relationships). Nodes can be tagged with labels, the "types" of the nodes. Relationships are connections between two nodes and have a direction and a type. Both nodes and relationships can have properties (attributes).

To follow, a few examples of how the knowledge is stored in the database are reported. In Figure 3, some of the main concepts of the KB are displayed. It is possible to see a data preparation technique (in blue), a method that implements it (in red), and the dimensions affected (in purple). Note that each dimension has a metric (in orange). Both techniques and methods can depend on data profile features (in yellow). The nodes and relationships have all the attributes, not shown here for brevity, present in the conceptual schema.

The ternary relationships are stored in the graph database using intermediate nodes. An example of how knowledge about the best methods to use in different contexts is stored can be found in Figure 4. This example indicates that the method "Mean Imputation" is the best imputation method to use in the context of dataset

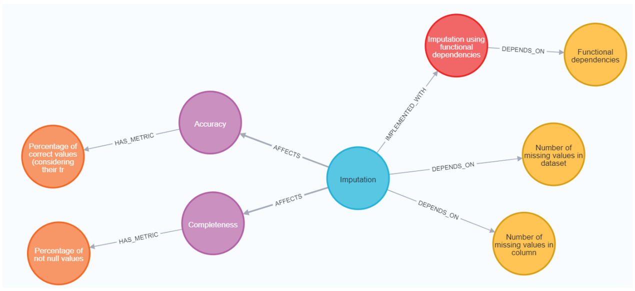
Imputation using functional dependencies
DEPENDS_ON
Functional dependencies
Accuracy
HAS_METRIC
Percentage of correct values (considering their tr
AFFECTS
IMPLEMENTED_WITH
Imputation
DEPENDS_ON
Number of missing values in dataset
AFFECTS
Completeness
HAS_METRIC
Percentage of not null values
DEPENDS_ON
Number of missing values in column

Figure 3: Imputation technique, dimensions and profile features

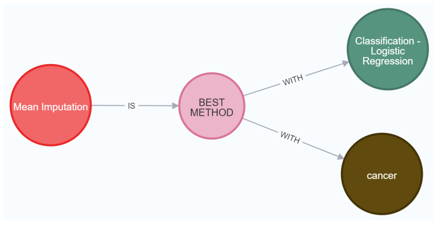

Figure 4: Best data preparation method

"cancer" and application "Classification - Logistic Regression". Similar implementations were used for the relationships "Impacts" and "User Choice".

5. Classifiers Design and Implementation

One of the goals of this work was to exploit the KB to suggest to the users the best methods for their specific analysis context. Considering a single preparation technique, the problem can be more precisely specified in this way: the user selects a data object and a ML application, and the objective is to return the best method with which to implement the technique in that context. This problem was moved to multi-class classification, where the target variable is the best method to use in the given situation.
In particular, the classifiers implemented in this work are focused on the data quality dimension of Completeness, specifically on the missing values imputation technique. The classifiers are given as input the data profile features of the data object and the selected ML application and return the best imputation method to fill in the missing values of that data object. Since a data object can either be an entire dataset or a single column, two separate classifiers have been developed: one for the entire dataset case, predicting the best method to impute all the missing values in the dataset, and one for the single column case, predicting the best method to impute the missing values of that specific column.
For building the classifiers, the first part of the work focused on generating the knowledge needed for training and testing. The following is a brief description of the knowledge generation process for the entire dataset case. First, a clean, complete dataset is considered, and its profile features are extracted and stored. Then there is an error injection phase, during which the dataset is injected with missing values in random positions. This injection is repeated many times, each time with a different percentage of missing values, to create multiple incomplete datasets. At this point, the missing values of these datasets are imputed several times, each time with a different imputation method. After an imputation method has been applied, the imputed dataset obtained is given in input to a ML algorithm, which is trained and tested (using cross-validation) on that dataset. The ML algorithm's performance serves as an indicator of the effectiveness of the imputation method: in this way it is possible to establish what is the best imputation method to use with a certain dataset and ML algorithm. The initial dataset's profile features, the percentage of missing values injected, the ML algorithm, and the obtained best imputation method are stored as a unit of knowledge. This procedure is repeated with different ML algorithms and clean datasets. A similar process is used to generate the knowledge for the single-column case, but in this case the missing values are injected only in the considered column, and at the beginning a feature selection algorithm is applied on the clean dataset to keep only its most significant columns. To make the knowledge generation processes more robust, each experiment was repeated in parallel 8 times, injecting the missing values in different positions. The generated knowledge was first stored in the KB (in the "Best Method" section) and then gathered in two datasets, one for each classifier to be trained and tested.
At this point, many kinds of classification algorithms were tried for both the entire dataset and the single-column case. To check their performance, k-fold cross-validation was applied. However, the results obtained were not satisfactory with any classification algorithm tried.

After analyzing the situation, it was observed that in the vast majority of cases, the scores of the three best imputation methods for a certain context are very close to each other: the difference in performance among the top three methods is often minimal. Therefore, predicting any of the top three methods would still give a very good suggestion for the users. For these reasons, it was decided to extend the generated knowledge to contain all three best imputation methods. After enriching the knowledge, further experiments with the classifiers were conducted, considering a prediction correct if the imputation method predicted belonged to the top three methods. As expected, a relevant boost in performance was obtained.
Since in the knowledge generation process varying percentages of missing values were injected in the same clean data object, in the obtained knowledge there were similar rows referred to the same data object and ML algorithm, but with different percentages of missing values. This raised concerns that presenting nearly identical rows to the classifier might confuse it. Therefore, it was decided to aggregate each group of these similar rows into a single row. In the case of the original knowledge (containing only the best method, not the top three), the aggregated best method was calculated as the mode of the best methods of the initial rows. For the enriched knowledge the aggregation process was more complex: each row had its ranking of the top three methods, the goal was to aggregate those rows in a single row with a unique ranking. To solve this rank aggregation problem, it was used an approach inspired by Borda's method. This aggregation led to a further improvement in the classifiers' performance.
After testing several different classifiers, coupled with the corresponding hyperparameter tuning, the highest performance achieved was an accuracy of 77.78 for the entire dataset case and 67.96 for the single-column case. For the entire dataset case, the classifier reporting the highest performance is a KNN model with a number of neighbors $k = 4$. For the single-column case, the best classifier is a Logistic Regression model, with Lasso regularization.

6. Tool Implementation

To demonstrate a practical application of the concepts discussed in the previous sections, during this work an already existing data preparation tool was extended and enriched with new functionalities. The tool is implemented with Flask: a framework in Python designed for web application development. The following is a brief general overview of the tool, with particular emphasis on the functionalities added in this work.
Firstly, the users can load the dataset to be prepared and select a ML application as their objective analysis. The dataset undergoes a data profiling phase, during which it is explored and its data profile is extracted. The tool shows the users the dataset's characteristics and the quality problems it may have. Subsequently, in the data preparation phase, the users can build their preparation pipeline by choosing the actions to perform from a list of possible preparation techniques and methods. The contributions to this tool added during this work are focused on the preparation phase. Specifically, the objective was to integrate the previously described KB and classifier into the tool. The first step was to connect the tool's Flask environment with the Neo4j database in which the KB is stored, to directly query the KB from the Python code.
In the preparation phase, the list of available data preparation actions is divided into segments, similarly to what was described before in Section 2: there is a segment for each of the quality dimension considered (completeness, accuracy, uniqueness), and a last segment dedicated to the selected ML application. Within each segment, there are several actions available for users to select. Each action is characterized by a preparation technique and a method that implements that technique.
All the actions available are loaded from the KB. For the segments dedicated at improving a quality dimension, the actions are obtained through the following querying process: for the considered dimension, all the techniques that affect that dimension positively (i.e., the influence type is "Improvement") are extracted from the KB along with the methods implementing each technique. Regarding instead the segment dedicated at improving the execution of the ML application, the procedure is slightly different. The

actions in this segment are customized for the ML application selected; in this way, the suggestions are personalized for that particular scenario. The querying process is the following: for the selected ML application, all the techniques that benefit that specific application are retrieved, along with the methods implementing the techniques. Retrieving the preparation actions from the KB, rather than hard-coding them in the system, provides flexibility and scalability: the KB operates as a dynamic repository of actions that can be constantly expanded and updated.
Another contribution made in this work is a functionality that retrieves and presents to the user information about a selected preparation action. Firstly, the user requests information about an action (composed as mentioned by a technique and a method). When the request is received, the tool executes a series of queries to the KB, retrieving all the information about the chosen action. The results of the queries are then assembled together and presented in a user-friendly textual response. The extracted information is the following: (i) the dimensions affected by the technique (specifying the type of its influence), (ii) which ML applications benefit from the technique, (iii) the data profile features on which the technique and method depend on, along with the corresponding description of the dependencies. The objective of this functionality is to make users more aware of the actions they can choose and to clarify when and why it is appropriate to use them.
Besides the KB, the developed classifier (entire dataset case) was also integrated into the tool. The classifier was trained on all the knowledge available, was saved (already trained) as a file, and then was imported into the tool. After the users load a new dataset, its data profile features are extracted and saved together with the selected ML Application. This data is then provided as input to the trained classifier, which predicts the best imputation method to be applied to the dataset. Once the best method is predicted, it is displayed to the users during the preparation phase, and it is automatically included in the data preparation pipeline. The integration of the classifier provides users with a suggestion tailored to their specific data and selected application.

7. Conclusions

This work presented a methodology aimed to guide users through the data preparation process, suggesting actions appropriate for their data and analysis objectives. The foundation of this methodology is a Knowledge Base, initially conceptually designed and then implemented and populated using a graph database. Through experiments, valuable knowledge about the best imputation methods for various contexts was generated and stored in the KB. The methodology uses classifiers to provide tailored recommendations for users' specific situations. In particular, the generated knowledge was leveraged for implementing two classifiers predicting the best imputation methods to fill in missing values of datasets or single columns. The developed Knowledge Base and classifiers serve as useful resources to support users through the data preparation process, pointing them toward the best choices but always ensuring complete freedom.

References

[1] Carlo Batini and Monica Scannapieco. *Data and Information Quality - Dimensions, Principles and Techniques*. Data-Centric Systems and Applications. Springer, 2016.

[2] Laure Berti-Équille. Learn2clean: Optimizing the sequence of tasks for web data preparation. In Ling Liu, Ryen W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia, editors, *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 2580–2586. ACM, 2019.

[3] Joseph M. Hellerstein, Jeffrey Heer, and Sean Kandel. Self-service data preparation: Research to practice. *IEEE Data Eng. Bull.*, 41(2):23–34, 2018.

[4] Shrey Shrivastava, Dhaval Patel, Anuradha Bhamidipaty, Wesley M. Gifford, Stuart A. Siegel, Venkata Sitaramagiridharganesh Ganapavarapu, and Jayant R. Kalagnanam. Dqa: Scalable, automated and interactive data quality advisor. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 2913–2922, 2019.