JOHANNES KEPLER
UNIVERSITY LINZ

Author
Pascal Badzura

Submission
Institute of Business Informatics - Data & Knowledge Engineering

Thesis Supervisor
Assoz.-Prof. Mag. Dr. Christoph Schütz

Assistant Thesis Supervisor
Simon Staudinger, MSc

August 2023

Design and Implementation of a web-based User Interface for the guided Assessment of Reliability of Classification Results using the Perturbation Approach

Master's Thesis
to confer the academic degree of

Master of Science
in the Master's Program

Economic and Business Analytics

JOHANNES KEPLER
UNIVERSITY LINZ
Altenberger Straße 69
4040 Linz, Austria
jku.at

Contents

List of Figures

List of Code

List of Equations

List of Tables

Abstract

With the growing volume of data in organizations, data mining techniques have become a critical part of the decision making process. However, assessing the reliability of individual predictive analytics results remains a complex challenge, as global metrics such as accuracy, precision, and recall only measure the overall performance of a predictive model. To render this reliability assessment process more user-friendly and streamlined, a web-based user interface has been developed in this master thesis, which guides users through the process of assessing the reliability of predictive analytics results as outlined by Staudinger et al. (2023). Using the web-based user interface, users can document their activities during the reference process and store this information in a knowledge graph. This documented information serves as the basis for generating perturbation options, which are used to introduce systematic alterations in feature values. The perturbation approach identifies sensitive feature values - those that, when perturbed, cause a shift in the original prediction of a predictive model. Thus, by using the perturbation approach, the user can effectively evaluate the reliability of an individual predictive analytics result.

Kurzfassung

Mit der wachsenden Datenmenge in Organisationen sind Data-Mining-Techniken zu einem wichtigen Bestandteil des Entscheidungsprozesses geworden. Die Bewertung der Zuverlässigkeit einzelner prädiktiver Analyseergebnisse bleibt jedoch eine komplexe Herausforderung, da die globalen Metriken wie Accuracy, Precision und Recall nur die Gesamtleistung eines prädiktiven Modells messen. Um diesen Prozess der Zuverlässigkeitsbewertung benutzerfreundlicher und effizienter zu gestalten, wurde in dieser Masterarbeit eine webbasierte Benutzeroberfläche entwickelt. Diese webbasierte Benutzeroberfläche führt die Nutzer durch den Prozess der Bewertung der Zuverlässigkeit von prädiktiven Analyseergebnissen, wie er von Staudinger et al. (2023) beschrieben wird. Mit Hilfe der webbasierten Benutzeroberfläche können die Nutzer ihre Aktivitäten während des Referenzprozesses dokumentieren und diese Informationen in einem Wissensgraphen speichern. Diese dokumentierten Informationen dienen als Grundlage für die Generierung von Störungsoptionen, mit denen systematische Veränderungen der Merkmalswerte vorgenommen werden. Der Störungsansatz identifiziert empfindliche Merkmalswerte, die bei einer Störung eine Änderung der ursprünglichen Vorhersage eines Vorhersagemodells bewirken. So kann der Nutzer mit Hilfe des Störungsansatzes die Zuverlässigkeit eines einzelnen prädiktiven Analyseergebnisses effektiv bewerten.

1. Introduction

An increasingly important part of an organization's IT strategy is the ability to process large amounts of data to support decision making (Krcmar, 2015). As part of this process, companies are using statistical methods to create predictive models using historical data. The discipline of extracting valuable insights from data using statistical models and applications is called data mining (Zaki & Jr, 2014). In practice, the CRISP-DM is the standard for carrying out such a project, which is short for **Cr**oss **I**ndustry **S**tandard **P**rocess for data **M**ining (Schröer et al., 2021).

To measure the performance of a data mining model, there are metrics such as accuracy or precision. However, these metrics consider the overall performance of a prediction model and not its individual predictions. For example, if a particular data point is to be predicted for which there is no comparable historical data, this can lead to inaccuracy in the prediction, raising questions about the reliability of the individual result. An illustrative example would be a prediction model that is supposed to predict whether it will rain the next day based on features such as temperature, humidity and windspeed. If there was no comparable day in the past to fall back on, the prediction may be unreliable. In the end, prediction models always provide predictions that can help in decision making. The problem is that these predictions may be unreliable and costly errors and missed business opportunities can result.

Staudinger et al. (2023) propose a reference process for assessing the reliability of predictive analytics results, which builds on the CRISP-DM and allows a statement to be made about the reliability of an individual prediction of a predictive model. The reference process defines activities along the entire CRISP-DM life cycle and specifies the knowledge about the data mining process that needs to be captured in a knowledge graph. The documented knowledge is finally used to assess the reliability of an individual prediction of a prediction model. The perturbation approach presented in the reference process is applied in this master thesis. In the perturbation approach, the input values of a feature are altered to show the effects of minimal changes in the feature values on the individual predictions. As soon as minor changes lead to different predictions, the question of the reliability of these predictions arises. If the prediction model mentioned above predicts rainfall for the following day and the predictions changes with a slightly changed temperature feature input, the question of the reliability of this individual prediction arises.

In this paper, the research question is posed, "*How can a tool be developed to assist users in evaluating the reliability of predictive analytics results?*". This question highlights the need for a solution that makes the reliability assessment process more accessible, intuitive, and efficient for end users. In response to this need, this research aims to develop a web-based user interface specifically designed for assessing the reliability of classification models.

The remainder of this master thesis is structured as follows. Chapter 2 explains the basic knowledge necessary for the master thesis. Chapter 2.1 deals with the already mentioned CRISP-DM and explains it in more detail. Chapter 2.2, the various methods of predictive models are first distinguished from each other, and the basic functioning of the relevant classification models is explained. Furthermore, this chapter introduces the evaluation methods used in practice for evaluate the performance of prediction models. Chapter 2.3 concludes the state-of-the-art chapter, in which the Semantic Web is explained and defined. Chapter 3 describes the reference process in detail, with subchapters 3.1 to 3.5 going into detail about the individual steps and activities based on the CRISP-DM. Chapter 3.6 describes the knowledge graph used in the reference process. Chapter 4 deals with the implementation of the web-based user interface and the technologies used. Chapter 5 documents the user's point of view, explaining the individual steps and pages of the web-based user interface in detail. Chapter 6 explains the implementation of the web-based user interface and the associated creation of the knowledge graph. Chapter 7 explains the methods used to test the usability of the web-based user interface. Chapter 8 draws a conclusion of the master thesis.

2. State of the art

In chapter 2.1, the areas of data mining and CRISP-DM are discussed. In chapter 2.2, the methods used in data mining are explained in more detail with a focus on classification prediction models. Chapter 2.3 addresses knowledge representation.

2.1. CRISP-DM

Data mining is the process organizations use to create value by extracting knowledge from data. In general data mining is defined as the process of discovering insightful, interesting, and novel patterns, as well as descriptive, understandable, and predictive models from data (Zaki & Jr, 2014). This process is commonly known by the acronym CRISP-DM, so only the acronym will be used in the following. CRISP-DM divides the entire data mining process into 6 steps, which are organized in a lifecycle. The first version of CRISP-DM was described by a consortium in 2000 and has remained essentially unchanged to this day (Chapman et al., 2000).

The following Figure 1 illustrates the CRISP-DM where the steps can be obtained. The direction of the arrow indicates the next step, while arrows pointing at each other indicate repetitive interactions between steps. The inner circle symbolizes that a project following CRISP-DM does not end with the implementation but is a repeating cycle.

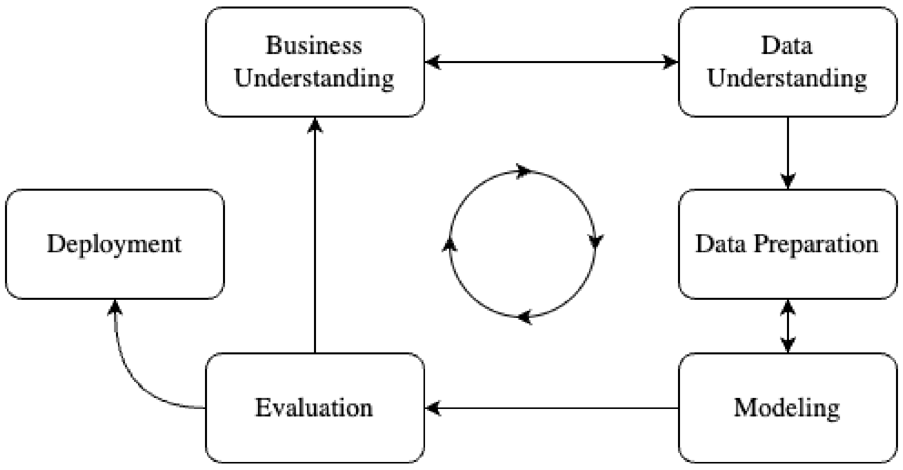


Figure 1 CRISP-DM (Chapman et al., 2000)

Business Understanding

According to CRISP-DM, Business Understanding is the first step. This step involves approaching the project from a business point of view and creating a detailed description. The objectives for a successful project are then established by translating the resulting business needs into data mining problems and objectives. The following steps of CRISP-DM are designed to be directed towards these problems and objectives.

Data Understanding

Building on the Business Understanding step, the Data Understanding step examines the data in more detail to gain initial insights. This can be used to identify correlations, quality issues in the data, or interesting subsets. In general, this step includes the activities of collecting, describing, examining, and evaluating data. These sub-steps can be summarized under the term Exploratory Data Analysis, abbreviated as EDA, which was introduced by John W. Tukey (Tukey, 1977). Some typical procedures would be to look at the distribution of the data, and the mean or median of a feature value (Morgenthaler, 2009). As Figure 1 shows, there is an interaction between the Business Understanding and Data Understanding steps. This implies that, following the assessment of the data during the Data Understanding step, the goals and objectives set in the Business Understanding step may be modified.

Data Preparation

The next step is Data Preparation, where the relevant data is selected, and a series of processing steps are performed to create a suitable data set. This includes data transformations and enrichments, such as aggregating or linking data to obtain more in-depth information, as well as data cleaning and checking, such as correcting missing values (Kordon, 2020b).

Modeling

The Modeling step in a CRISP-DM project involves selecting the appropriate prediction model and developing one or multiple test models. For the same data mining problem, several methods may be suitable. Therefore, different data mining techniques are selected, applied, and optimized to find the most appropriate model. During Modeling, problems in the data can be identified or new insights gained, so there is a link between the Data Preparation step and the Modeling step.

Evaluation

Before deploying a model, its performance needs to be assessed to determine if it meets the project objectives. This is achieved by reviewing both the steps and the results of the data mining process to identify areas for improvement. These improvement areas can then be incorporated into the next cycle to improve performance or meet new business objectives.

Deployment

The final step of a data mining project following the CRISP-DM is the Deployment step. This involves selecting the actions to be taken based on the requirements. For example, a report or a presentation might be an appropriate option.

2.2. Predictive models and evaluation

To identify patterns in data, various statistical methods are used. A general distinction is made between supervised and unsupervised machine learning models (Kordon, 2020a). The difference between supervised and unsupervised models is that the data for supervised models is labelled. This means that data is needed for which the true value of the prediction is already known. The purpose of this paper is to evaluate the predictions of a classification prediction model which is a supervised machine learning method, other supervised and unsupervised methods will not be discussed further.

Classification is the training of a predictive model to predict the class labels of input data. For later evaluation of the model, the data is separated into a training and a test set. Then the model learns from the given labelled training data and is afterwards capable of classifying the unseen test data into different categories accordingly. There are several different statistical methods for classifying data and each method has certain advantages and disadvantages, but as the description of each method is beyond the scope of this master's thesis, it is referred to other literature for further information (Kordon, 2020a).

Suppose a model was trained for identifying fraudulent credit card transactions. There are some variables for each purchase, such as the date, amount, and place of payment, as well as other variables. Based on these variables, the classification model decides whether a purchase is fraudulent or not. To evaluate the overall performance of the model, there are several metrics (Kordon, 2020a). The most common ones are briefly explained below.

The Confusion Matrix summarizes how many predictions were correct and how many were incorrect, as shown in Table 1. This is done by comparing the labelled data, of which the true values are available, with the predicted values. The predictions can be divided into *True Positive*, *True Negative*, *False Positive* and *False Negative*.

If a fraudulent credit card transaction was detected as such, this results in a *True Positive* prediction. If it was not detected, these values are aggregated under *False Negative*. Predictions that were not fraudulent and were classified as such by the model are counted as *True Negatives*. *False Positives* are therefore predictions where the prediction model predicts the purchase as fraudulent, which was not the case.

Table 1 Confusion Matrix with example values

		Reality (Truth)	
		Positive (Fraud)	Negative (No Fraud)
Model (Prediction)	Positive (Fraud)	True Positive (TP): Purchase was fraudulent, and the model classified it as fraudulent **40**	False Positive (FP): Purchase was not fraudulent, and the model classified it as fraudulent **10**
	Negative (No Fraud)	False Negative (FN): Purchase was fraudulent, and the model classified it as not fraudulent **20**	True Negative (TN): Purchase was not fraudulent, and model did not classify it as fraudulent **30**

Based on the Confusion Matrix, other metrics can be derived. Equation 1 shows the metric *accuracy*, which measures the proportion of correct predictions to all other predictions. Using Equation 1 below, the *accuracy* would be 70 % if the model correctly predicts 70 out of 100 cases.

Equation 1 Accuracy

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Precision, shown in Equation 2, assesses the proportion of *True Positive* predictions that are correct out of all positive predictions and is derived as the division of the total number of *True Positive* predictions and the total number of *True Positive* and *False Positive* predictions. For example, *precision* would be 80 % if the model predicted 50 positive events and 40 of them occurred.

Equation 2 Precision

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall, which is calculated as the ratio of *True Positive* predictions to both *True Positive* and *False Negative* predictions, measures the proportion of positive examples that the classifier has correctly detected. For instance, if there were 60 positives in the dataset and the model truly predicted 40 positives, it would have falsely labelled 20 predictions as negatives. In this example, the *Recall* would be 66.67 %. Equation 3 shows how *recall* is calculated.

Equation 3 Recall

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

The metrics presented can be applied in different ways depending on the target and distribution and other aspects. The purpose of this brief overview is to provide a basic understanding of how predictive model performance is evaluated. It is important to note that this form of performance measurement refers to the overall performance of a classifier.

2.3. Semantic Web

This chapter is an introduction to the Semantic Web. Information on the web is designed to be understandable by humans. The ability of humans to make connections allows them to understand and interpret the relationships between data. Machines, on the other hand, do not have this capacity. To make this information usable for machines, Berners Lee developed the Semantic Web (Berners-Lee et al., 2001).

The Semantic Web is an extension of the existing World Wide Web that adds machine-interpretable metadata which was created by The World Wide Web Consortium (W3C). Metadata is data that is used to describe other data (Bargmeyer & Gillman, 2000). W3C is an international community developing Web standards and has developed Semantic Web standards, specifically eXtensible Markup Language XML (W3C, 2008a), Resource Description Framework RDF (W3C, 2014b) and OWL (W3C, 2012).

The purpose of these standards is to enable machines to understand relationships, connections, and the meaning of data so that humans and machines can work better together. Rather than being a separate Web, the Semantic Web should be seen as complementary. On the one hand, XML is used to give data a structure of choice and does not add any additional meaning to the data. RDF and OWL, on the other hand, are special ontology languages designed for the Semantic Web, which make it possible to add meaning to data and to process the information received without losing the original meaning.

In RDF, data is packaged into so-called RDF triples. These triples consist of a subject, a predicate, and an object. To identify each subject, each predicate and each object, so-called Universal Resource Identifiers, abbreviated URIs, are used. An URI is defined by a string of characters that uniquely identifies resources (W3C). The predicate refers to the subject and the object and thus creates a semantic link between them. This makes it possible to create new unique subjects and objects and determine the relationships between them. A set of RDF triples is represented in a directed graph structure, as shown in Figure 2.

If an object is a value that is always valid for all applications, it can take on data values and types concretized by literals, such as strings, boolean values, numbers, or times. The nodes in Figure 2 reflect the subjects and objects that are connected by the directed edges of the predicate.

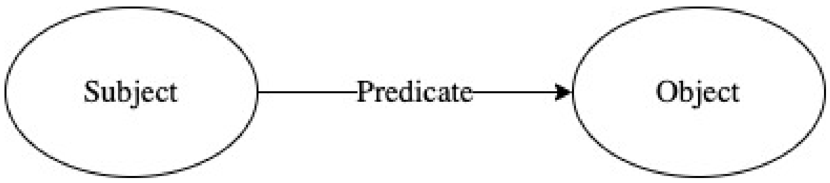


Figure 2 Graphical notation of RDF triple (W3C 2014)

RDF graphs, are written in triple syntax, such as N3 (W3C, 2011a), N-Triples (W3C, 2004) or Turtle (W3C, 2011b), and can be stored in a triple store, where several RDF graphs can be combined into one RDF dataset. A triple-store is a type of a graph database. To access and modify data stored in RDF format, a query language called SPARQL is used, which is a recommended standard by the W3C (W3C, 2008b). It can be used to query multiple data sources, extract information, and then modify the resulting data.

To further extend the semantics of RDF, the Web Ontology Language OWL was standardized by the W3C, which allows more complex relationships to be created. For example, OWL makes it possible to create semantic relationships such as. A is married to B, therefore B is married to A.

3. Reference process

The CRISP-DM presented in chapter 2.1 is used in practice to apply predictive models and to generate added value from data (Plotnikova et al., 2022). For this purpose, the evaluation metrics presented in chapter 2.2 can be used to assess the performance of a model. These metrics are an indicator of the overall performance of a model. However, no statement can be made about the reliability of individual predictions, because the metrics such as accuracy, precision and recall provide the same value for all predictions and therefore, do not indicate the reliability of an individual prediction. For example, it is possible that the model changes the prediction as soon as a feature value changes minimally. In this case, it could be an unreliable prediction because minimal changes may occur in feature values. If this were the case, a decision would be made based on an unreliable prediction and costly mistakes or missed business opportunities could result. It is possible for a model to perform particularly well in an area where there are many data points in the training data and worse in an area where there are not many data points. For example, a temperature feature, among others, is used to predict rainfall. If the data was collected during the summer in Austria, there will be more data points below 30 degrees in the summer than above 30 degrees. Therefore, for days below 30 degrees, more data points would be available, and the accuracy would most likely be better.

To solve this problem, Staudinger et al. (2023) developed a reference process for assessing the reliability of predictive analytics results. In this master thesis, only classification methods from chapter 2.2 are considered, but the reference process can be applied to other prediction models with some adaptations. The reference process is based on the CRISP-DM and the information collected during the process is stored in a knowledge graph. A knowledge graph is a graph-theoretic representation of human knowledge in such a way that it can be captured with semantics by a machine (Kejriwal, 2019). This ensures that all information generated in the process is available at the end to assess the individual prediction. In the remainder of this paper, that process will be referred to as the reference process. The knowledge graph is created using the RDF triples described in chapter 2.3 and is thus given semantic meaning.

In the following the reference process for assessing the reliability of predictive analytics results is summarized. Figure 3 shows the different example activities of the reference process for each CRISP-DM step. Chapter 3.1 describes the activities of the Business Understanding step. In chapter 3.2, the activities of the Data Understanding step are explained in more detail. In chapter 3.3 the activities during the Data Preparation step are explained. This information is then combined and evaluated into perturbation options during the Modeling step in chapter 3.4. In chapter 3.5, the Deployment activities are explained. The general process of the knowledge graph is discussed in chapter 3.6.

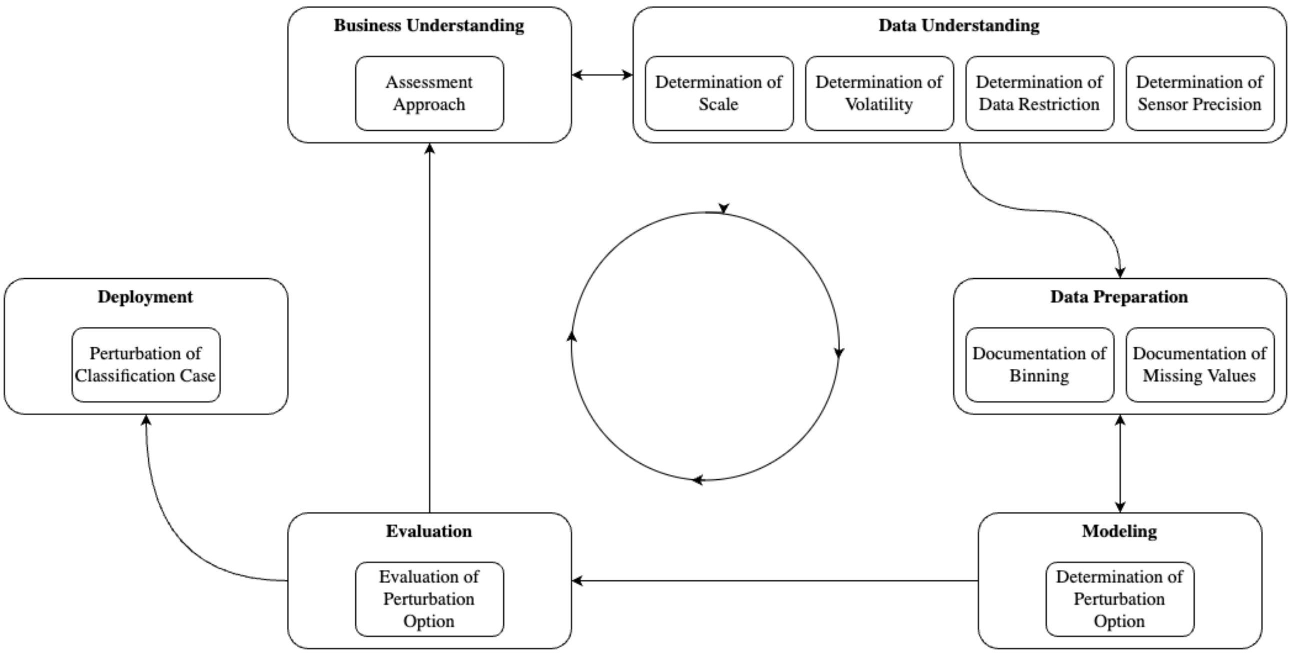

Business Understanding
Assessment Approach
Data Understanding
Determination of Scale
Determination of Volatility
Determination of Data Restriction
Determination of Sensor Precision
Deployment
Perturbation of Classification Case
Data Preparation
Documentation of Binning
Documentation of Missing Values
Evaluation
Evaluation of Perturbation Option
Modeling
Determination of Perturbation Option

Figure 3 Example activities for assessing the reliability of classification results

3.1. Business Understanding

As in the CRISP-DM, the Business Understanding step is the first step of the reference process. In contrast to the CRISP-DM, where the focus is on defining the business and data mining objectives, the reference process aims to determine the appropriate assessment approach for assessing the reliability of predictive analytics results based on the specific business requirements, which can be seen in the Business Understanding step in Figure 3. The chosen assessment approach is then stored in the knowledge graph.

The reference process presents two methods for assessing the reliability of classification predictions. The perturbation approach (I) slightly alters feature values and therefore creates new input cases for the prediction model. In this way, it is possible to identify sensitive feature values that lead to a different prediction and conclusions about the reliability of the individual predictions can be drawn.

The local quality measures approach (II) assesses reliability across different data areas and requires complete training data. However, as the perturbation approach (I) is used in this thesis, the local quality measures approach (II) is not discussed, for further information it is referred to the reference process.

3.2. Data Understanding

As in CRISP-DM, during the Data Understanding step the required data is collected, and the quality of the data is assessed. Figure 3 shows the example Data Understanding activities which are divided as follows: *Determination of Scale*, *Determination of Volatility*, *Determination of Data Restriction* and

Determination of Sensor Precision. The generated information is collected and stored in the knowledge graph. The activities of during the Data Understanding step are explained below.

Scale

In the Data Understanding step, the scale level for each feature is set. This is an important and necessary activity for the perturbation approach, as different methods for altering the features are available for each level. Features can be divided into three different level of scales: *cardinal*, *ordinal* and *nominal* (Fisher & Marshall, 2009). Feature values are *cardinal* as soon as the difference between the values can be mathematically calculated and mathematically quantified. *Ordinal* characteristics have a natural ranking for which the difference cannot be quantified mathematically. *Nominal* values are all other values that do not have a ranking. *Cardinal* data can be, for example, all measured values expressed in centimeters or kilograms. *Ordinal* data does not have a natural origin, but a natural order. Compared to *cardinal* data, however, no mathematical operators can be used. For example, one can say that grade 1 is better than grade 2 in the Austrian school system, but the difference between the individual grades cannot be mathematically calculated. *Nominal* data is any data that cannot be ranked, such as hair or eye color.

Volatility of features

To assess the reliability of classification predictions, it is important to know the volatility of a feature value. The reason is that information about the volatility of a feature provides clues as to whether a perturbation of the feature should be performed. A distinction can be made between *low*, *medium,* and *high volatility*. The division between the volatilities does not follow any standard and can be used depending on the use case. For example, perturbing a feature that represents wind speed, which is considered a volatile feature, as it can change very quickly. If the slightly altered value changes the prediction, this may indicate an unreliable prediction. In contrast, some characteristics do not change often, such as a client's education or marital status. In this case, a change in the prediction may not indicate an unreliable prediction.

Data restriction

In the Data Understanding step, it is crucial to gather information about the data restrictions that apply to the features to be analyzed. This activity can be used to determine if there are any data restrictions imposed by the organization or the data mining objectives. For example, consider a credit scheme where, according to the law, credit can only be granted to persons over eighteen years of age. If other conditions need to be met for the predictive model values to meet the business or data mining objectives, they are documented during this step. Therefore, the determination of data restrictions ensures that the predictive model operates within the relevant rules and guidelines.

Sensor precision

Feature values can be captured in several ways. For example, cardinal features can be generated by a sensor. This sensor has a certain precision, which indicates the accuracy with which the sensor detects the value of the feature. During this activity it is possible to store information about the precision of the sensor. Since the true value of the feature must be within this precision range, feature values generated by a sensor should not change the prediction within this range. If a prediction model predicts differently when a sensor value is perturbed compared to the original prediction, this could be a strong indication of an unreliable prediction. An example could be a temperature measurement. If a temperature of 20 °C was measured with an accuracy of 1 %, the prediction should not change in a range of 19.8 °C to 20.2 °C.

3.3. Data Preparation

During Data Preparation, modifications are applied to the training data aiming to solve data quality issues. This includes activities such as grouping the data into bins and documenting the methods used to deal with missing values.

Binning

Cardinal features can be grouped into bins, where all values within a certain range are aggregated into one bin. The original data can be replaced with the bin data, such as the mean, reducing the uniqueness of the data and allowing for the transformation of cardinal features into ordinal or nominal features. However, the loss of original data means that perturbing feature values within the bin is useful to check for the reliability of the prediction model. If the prediction changes with perturbed values within the bin, it could be an indication of an unreliable prediction. There are several different methods for binning cardinal values, *Equal Width* and *Equal Frequency* for example (Wu et al., 2013). While *Equal Width* binning takes a range in the data field, the bins can contain values of different frequencies, whereas *Equal Frequency* binning considers how many values are within a bin. An example could be the binning of temperature into low temperature 0 – 10 °C, medium temperature 10 – 20 °C and high temperature 20, 30 °C.

Missing values

When missing values are encountered in the data, several methods can be used to handle them, such as deleting the row, inserting the mean or other methods which can be used depending on the use case (Kang, 2013). Each method has its advantages and disadvantages. For example, when deleting the entire data point, valuable information about the other features is lost. In addition, using the mean for a missing value for that data point may be inaccurate. The documentation of missing values should be seen as a source of information that the true values were not available when the model was trained. However, it is crucial to document the methods used to handle missing data, as the absence of the true value may impact the reliability of the prediction. Therefore, a perturbation for the features with missing values is useful to assess the reliability of the prediction model. For missing values, the example of a temperature measurement can be used. Due to a technical defect in the sensor, a measurement may not have taken place and the data is missing. Now different methods can be used and e.g., replacing the missing value with the average temperature at the corresponding time.

3.4. Modeling

In this chapter the perturbation approach is explained in detail. In this approach, the documented information collected from the steps of Business Understanding, Data Understanding and Data Preparation is transformed into perturbation options. Perturbation options refer to algorithms that systematically change the values of a feature. Based on the collected information different perturbation options are available which are explained below. The perturbed values are then inserted into the original case resulting in a perturbed case which is forwarded to the classification model. The resulting predictions are then used to assess the reliability of the predictions by comparing them to the original predicted value. Doing so, the feature values which have a significant influence on the prediction can be identified.

The reference process uses the perturbation level to determine whether a change in the prediction is permitted. To indicate the possible consequences of changing the prediction, the perturbation options can be divided into *red, orange*, and *green* perturbation levels. The *red* perturbation level means that the prediction must not change if the feature values are perturbed. If a prediction for a feature with a *red* perturbation level changes, it is considered unreliable. The *orange* perturbation level means that the prediction may change in borderline cases, which means that a changed prediction may not always indicate an unreliable case, but further examination of the case would be advisable. The *green* perturbation level means that the prediction is likely to change if the test cases are changed. These levels of perturbation can be used to provide additional information to determine the reliability of the prediction. The tables 2 - 8 below show the perturbation levels which are recommended by the reference process.

Having explained the influence of *volatility*, *data restriction*, *sensor precision*, *binning* and *missing values* in the previous chapter, some examples for perturbation options, and the influence of the scale on them are explained below.

Scale

The information about the level of scale has a direct impact on the selection of available perturbation options for the respective feature. For each level, examples of perturbation options are provided. Features have already been classified into *cardinal*, *ordinal* and *nominal* levels of scale, as explained in Chapter 3.2.

Cardinal features can be perturbed using various perturbation options, including *Percent Perturbation, Fixed Amount Perturbation, Range Perturbation, Sensor Precision Perturbation* and *Bin Perturbation*. The latter two options require sensor precision and binning information which is collected during the Data Understanding and Data Preparation steps. The perturbation options defined in the reference process for cardinal features are outlined in tables 2-6. These are to be considered as examples, further perturbation options can be created depending on the use case.

Table 2 Perturbation option - Percentage Perturbation

Name	**Percentage Perturbation**
Scale of Feature	Cardinal
Additionally required values	Percentage
Recommended perturbation level	Orange

Table 2 shows the *Percentage Perturbation* option, where a cardinal value is changed by a percentage to create new values within a percentage range. For example, a feature with a value of 10 can be perturbed by 10 % to create new perturbed values between 9 and 11. The desired percentage must be specified for this perturbation option. The recommended perturbation level for this perturbation option is orange.

Table 3 Perturbation option – Fixed Amount Perturbation

Name	**Fixed Amount Perturbation**
Scale of Feature	Cardinal
Additionally required values	Amount, steps
Recommended perturbation level	Orange

Table 3 shows the perturbation option *Fixed Amount Perturbation*. This perturbation option can be used for cardinal features to create new perturbed values that are within a defined value distance of each other. The distance is passed to the method via the additionally required value amount. Additionally, the steps on how many new values should be created are required. For example, if the initial value 20 is to be perturbed by the amount 5 in 3 steps, the following perturbed values are created (5, 10, 15) for the lower range and (25, 30, 35) for the upper range. The recommended perturbation level for this perturbation option is orange.

Table 4 Perturbation option - Range Perturbation

Name	**Range Perturbation**
Scale of Feature	Cardinal
Additionally required values	Lower bound, upper bound, steps
Recommended perturbation level	Orange

Range Perturbation allows to perturb a cardinal value within a range defined by the lower and upper bound. In addition, the number of steps is required for this option to determine the number of perturbed values. If the lower bound is 100, the upper bound is 200 and it should be perturbed in 6 steps, the following new perturbed values are created (100, 120, 140, 160, 180, 200). Table 4 shows the summary of this perturbation option. As the prediction may change, the recommended perturbation level for this perturbation option is orange.

Table 5 Perturbation option - Sensor Precision Perturbation

Name	**Sensor Precision Perturbation**
Scale of Feature	Cardinal
Additionally required values	Sensor precision, steps
Recommended perturbation level	Red

Table 5 provides an overview of the *Sensor Precision Perturbation* option for cardinal features. This option requires information about the sensor precision of a feature, obtained during the Data Understanding step. For example, if a feature with a value of 100 measured by a sensor has a precision of 5%, new values will be generated in the range of 95 to 105. In addition, the number of steps can be specified to create several values. Assuming 6 steps have been defined, the following perturbated values are created (95, 97, 99, 101, 103, 105) The recommended perturbation level for this perturbation option is red, as the prediction must not change within the sensor precision range.

Table 6 Perturbation option - Bin Perturbation

Name	**Bin Perturbation**
Scale of Feature	Cardinal
Additionally required values	Bins, steps
Recommended perturbation level	Red

Table 6 provides an overview of the *Bin Perturbation* option for cardinal features. This perturbation option is created using the same procedure as the *Range Perturbation*, but this time the upper and lower bounds are replaced by the upper and lower bound of the bins created during the Data Preparation step. An example would be the temperature example in chapter 3.3, where temperatures could be divided into low, medium, and high temperatures bins. If *Bin Perturbation* is performed the original value of the use case will be perturbed in between the lower and upper bound of the bin. For example, a temperature of 15 falls into the bin of medium temperature and the perturbation option was defined with 6 steps. The new perturbed values would be (10, 12, 14, 16, 18, 20) °C. As the prediction must not change within the range of a bin, the recommended perturbation level for this perturbation option is red.

Compared to nominal characteristics, ordinal characteristics have a natural ranking. Therefore, for ordinal features, it is possible to create new feature values according to the ranking with *Perturb In Order* or to *Perturb All Values*. The perturbation options defined in the reference process for ordinal features are summarized in Table 7 and Table 8. These are to be considered as examples, further perturbation options can be created depending on the use case.

Table 7 shows the *Perturb In Order* option for ordinal features. For this perturbation option, the number of steps and the ranking of the values are required. Values are going to be perturbed according to these additional values in both directions. The school grade example from chapter 3.2 can be used as an example. Suppose a model has a feature about school grades and this feature is to be perturbed.

In this way, the number of steps in which new values are to be created can be set. The order then gives the perturbed values. If the grade 2 is to be perturbed by 2 steps, the following perturbed values (1, 3, 4) are created. Since 1 is the lower limit, it is not perturbed below this value. The recommended perturbation level for *Perturb In Order* is orange.

For nominal features, *Perturb All Values* is summarized in Table 8. The color of a person's hair can be used as an example here since it cannot be put in any order. If a model contains this feature, the only solution is to perturb all nominal values and create a new prediction for all available feature values. The recommended perturbation level for this *Perturb All Values* is orange.

Table 7 Perturbation option - Perturb In Order

Name	**Perturb In Order**
Scale of Feature	Ordinal
Additionally required values	Steps, values in order
Recommended perturbation level	Orange

Table 8 Perturbation option - Perturb All Values

Name	**Perturb All Values**
Scale of Feature	Ordinal / Nominal
Additionally required values	Steps, values
Recommended perturbation level	Orange

The reference process introduces the perturbation mode, which serves as a method to regulate the order of perturbation options when multiple options are applied simultaneously. The number of new perturbations can increase exponentially, and the perturbation mode allows to control the order of the features to be perturbed. Three different modes are available: *full mode, priority mode* and *selected* mode. *Full mode* performs all perturbation options without order. The *priority mode* prioritizes a perturbation option based on the ranking. The *selected mode* only perturbs the selected features.

In the Evaluation step, as in the CRISP-DM, the perturbation options are reassessed to determine whether they meet the objectives set in the Business Understanding. Since the Evaluation step is not implemented specifically in the web-based user interface and does not differ significantly from the Modeling step, it will not be discussed further in this master thesis.

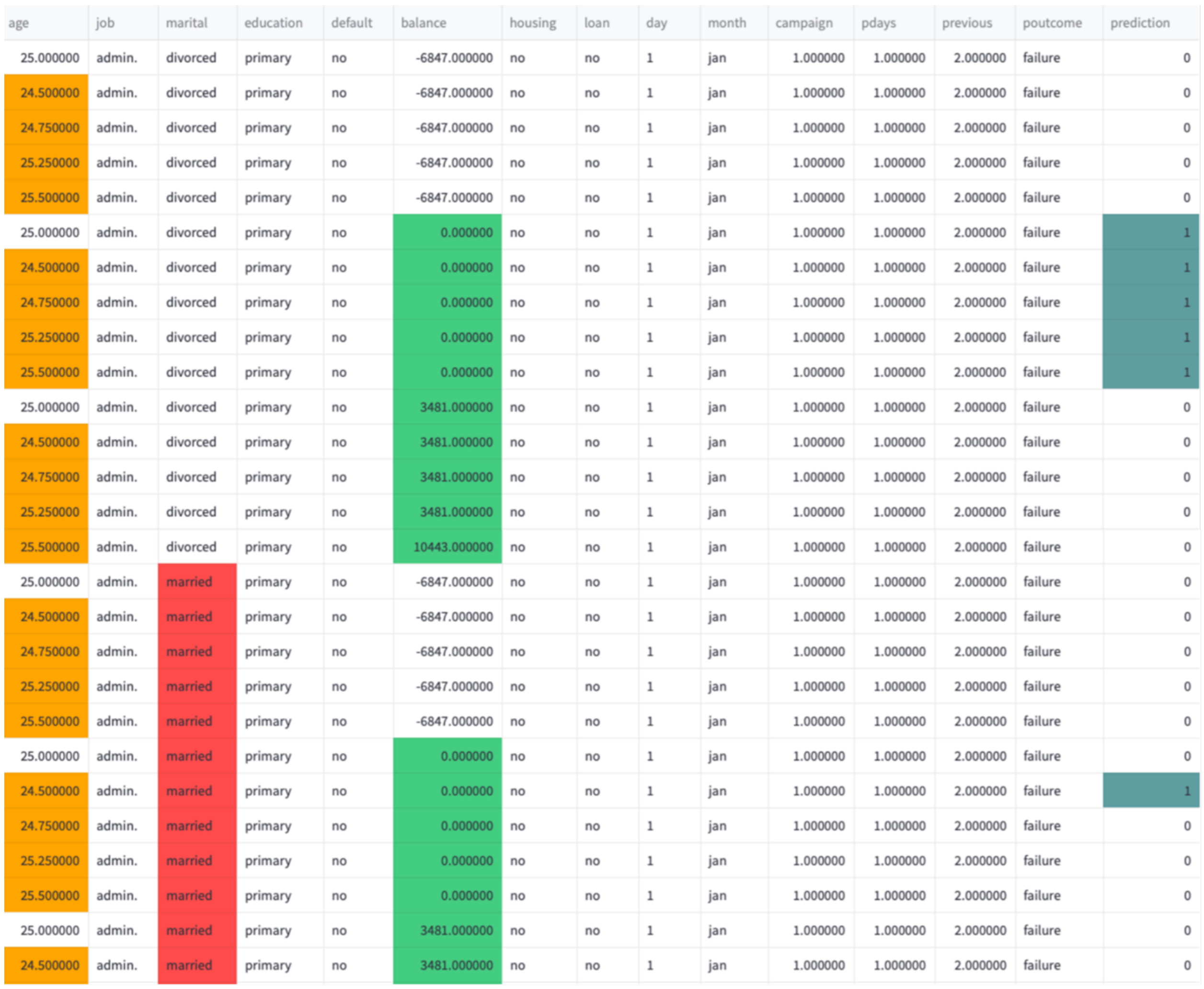
3.5. Deployment

During the Deployment step, the perturbed feature values are created using the perturbation options and perturbation mode, and the perturbed test cases are generated. Which are then inserted into the classification model and the predictions for these perturbed test cases are then evaluated to identify if the prediction changed compared to the original prediction. As an example, a classification model was used which classifies whether a customer has agreed to place a long-term deposit, which is illustrated in Figure 4. A prediction value of 0 indicates that no deposit was made, and a value of 1 indicates that a deposit was made. The dataset contains features about the person such as age, job, marital, education, default, balance, housing, and loan. As well as other features about the date and the campaign. The duration feature has been removed from the data set as this is not known before a call (Moro et al., 2014). Table 9 shows an overview of the feature descriptions for the telemarketing dataset.

Table 9 Feature description telemarketing dataset

Feature name	Explanation
Age	Client’s age (numerical)
Job	Client’s education (categorical)
Marital	Client’s marital status (categorical)
Education	Client’s education status (categorical)
Default	If a client has credit in default (categorical)
Housing	If a client has a housing loan (categorical)
Loan	If a client has a personal loan (categorical)
Day	Last contact day (categorical)
Month	Last contact month of the year (categorical)
Campaign	Number of contacts performed during this campaign and for this client (numeric)
Pdays	Number of days that passed by after the client was last contacted from a previous campaign (numeric)
Previous	Number of contacts performed before this campaign and for this client (numeric)
Poutcome	The outcome of the previous marketing campaign (categorical)
Prediction	Target variable if client placed a deposit; 0 = no, 1 = yes (categorical)

As can be seen in Figure 4, the cardinal feature age, the nominal feature marital and the cardinal feature balance were perturbed. The color indicates the perturbation level of the perturbation option. The age feature is perturbed first, then the balance feature and finally the marital feature. For the age feature, a 2 % *Percentage Perturbation* was applied. For the balance feature, the perturbation option *Range Perturbation* with the lower bound of -6847 USD and the upper bound of 10443 USD with three steps was applied and for the feature marital *Perturb All Values* was chosen. The prediction changed as soon as the feature balance is 0 USD, the additional perturbations of age and marital status do not change the prediction. The individual perturbations of age, marital status and balance do not change the prediction.



age	job	marital	education	default	balance	housing	loan	day	month	campaign	pdays	previous	poutcome	prediction
25.000000	admin.	divorced	primary	no	-6847.000000	no	no	1	jan	1.000000	1.000000	2.000000	failure	0
24.500000	admin.	divorced	primary	no	-6847.000000	no	no	1	jan	1.000000	1.000000	2.000000	failure	0
24.750000	admin.	divorced	primary	no	-6847.000000	no	no	1	jan	1.000000	1.000000	2.000000	failure	0
25.250000	admin.	divorced	primary	no	-6847.000000	no	no	1	jan	1.000000	1.000000	2.000000	failure	0
25.500000	admin.	divorced	primary	no	-6847.000000	no	no	1	jan	1.000000	1.000000	2.000000	failure	0
25.000000	admin.	divorced	primary	no	0.000000	no	no	1	jan	1.000000	1.000000	2.000000	failure	1
24.500000	admin.	divorced	primary	no	0.000000	no	no	1	jan	1.000000	1.000000	2.000000	failure	1
24.750000	admin.	divorced	primary	no	0.000000	no	no	1	jan	1.000000	1.000000	2.000000	failure	1
25.250000	admin.	divorced	primary	no	0.000000	no	no	1	jan	1.000000	1.000000	2.000000	failure	1
25.500000	admin.	divorced	primary	no	0.000000	no	no	1	jan	1.000000	1.000000	2.000000	failure	1
25.000000	admin.	divorced	primary	no	3481.000000	no	no	1	jan	1.000000	1.000000	2.000000	failure	0
24.500000	admin.	divorced	primary	no	3481.000000	no	no	1	jan	1.000000	1.000000	2.000000	failure	0
24.750000	admin.	divorced	primary	no	3481.000000	no	no	1	jan	1.000000	1.000000	2.000000	failure	0
25.250000	admin.	divorced	primary	no	3481.000000	no	no	1	jan	1.000000	1.000000	2.000000	failure	0
25.500000	admin.	divorced	primary	no	10443.000000	no	no	1	jan	1.000000	1.000000	2.000000	failure	0
25.000000	admin.	married	primary	no	-6847.000000	no	no	1	jan	1.000000	1.000000	2.000000	failure	0
24.500000	admin.	married	primary	no	-6847.000000	no	no	1	jan	1.000000	1.000000	2.000000	failure	0
24.750000	admin.	married	primary	no	-6847.000000	no	no	1	jan	1.000000	1.000000	2.000000	failure	0
25.250000	admin.	married	primary	no	-6847.000000	no	no	1	jan	1.000000	1.000000	2.000000	failure	0
25.500000	admin.	married	primary	no	-6847.000000	no	no	1	jan	1.000000	1.000000	2.000000	failure	0
25.000000	admin.	married	primary	no	0.000000	no	no	1	jan	1.000000	1.000000	2.000000	failure	0
24.500000	admin.	married	primary	no	0.000000	no	no	1	jan	1.000000	1.000000	2.000000	failure	1
24.750000	admin.	married	primary	no	0.000000	no	no	1	jan	1.000000	1.000000	2.000000	failure	0
25.250000	admin.	married	primary	no	0.000000	no	no	1	jan	1.000000	1.000000	2.000000	failure	0
25.500000	admin.	married	primary	no	0.000000	no	no	1	jan	1.000000	1.000000	2.000000	failure	0
25.000000	admin.	married	primary	no	3481.000000	no	no	1	jan	1.000000	1.000000	2.000000	failure	0
24.500000	admin.	married	primary	no	3481.000000	no	no	1	jan	1.000000	1.000000	2.000000	failure	0

Figure 4 Deployment example perturbation telemarketing

3.6. Knowledge graph

To be able to use the information collected during the CRISP-DM steps semantically, it is stored in a knowledge graph, which is filled with the RDF triples known from chapter 2.3 and is thus machine-readable. As the data is stored historically, it can be used for documentation and auditing purposes, allowing the reasoning behind a particular decision to be understood in retrospect. By integrating the knowledge into the CRISP-DM it is possible to monitor and improve the quality. It is also possible to provide a recommendation system, as the data stored in graph databases are suitable for recommendation systems (Mohammedali, 2019).

In the reference process, the PROV ontology is chosen to represent provenance information (W3C, 2013). The PROV-O is an ontology which provides a set of classes, properties and constraints that can be used to represent and exchange provenance information produced in different systems and contexts. It can be used to specialize in the development of new classes and attributes for the modeling of provenance information in a range of applications and domains. Depending on their needs and the level of detail they wish to represent their provenance information, PROV-O users may only need to use parts of the whole ontology. For this purpose, PROV-O terms (classes and properties) are divided into three categories: starting points, extended terms, and terms for qualifying relationships. In the context of the master's thesis, a basic understanding of PROV-O is sufficient. A comprehensive description would go beyond the scope of the thesis. Further information can be found in the PROV-O documentation (W3C, 2013).

The Starting Point Terms provide the general framework for the rest of the PROV ontology. The use of these terms enables the creation of simple provenance descriptions that can be further processed with terms from the Expanded and Qualified Terms. The Starting Point Terms of PROV-O consist of the following three classes illustrated in Table 10.

Table 10 PROV-O Starting Point Terms (W3C, 2013)

Terms	Description
Activity	An activity is anything that takes place over a period of time and interacts with or affects other entities.
Entity	An object created by an activity is called an entity in PROV-O, whether it is physical, digital, conceptual or something else.
Agent	An agent is something that is responsible for an activity.

Since the master thesis is about the general functioning of the reliability assessment, some aspects of the knowledge graph from the reference process were not considered for the implementation of the master thesis for example, the agent.

The knowledge graph is structured in 3 different levels: *generic level, method level* and *problem level*, which build on each other and are explained in more detail in the following. In the reference process, each level is divided into a *development view* and a *deployment view*, which track knowledge about the *development* of a predictive model and its *deployment*, respectively. During the *development view* the perturbation options based on the knowledge documented in the Business Understanding, Data Understanding, Data Preparation, Modeling, and Evaluation steps are developed. In the *deployment view* these perturbation options are used to perturb the input data and assess the reliability of the prediction model by comparing the resulting predictions.

3.6.1. Generic level

The generic level is the fundamental element of the reference process. Figure 5 shows the generic level in the development view. It shows that for each step in the CRISP-DM, an entity was created which has a property *wasGeneratedBy* for the respective activity. For example, the *BusinessUnderstandingEntity wasGeneratedByBUA BusinessUnderstandingActivity*. The wasGeneratedBy property was taken from the PROV-O and is defined by the documentation as follows: Generation is the completion of production of a new entity by an activity. This entity did not exist before generation and becomes available for usage after this generation (W3C, 2013).

Figure 5 shows that the Modeling entity has a connection, named *modelingEntityWasDerivedFrom* to the *BusinessUnderstandingEntity*, *DataUnderstandingEntity* and *DataPreparationEntity*. This connection illustrates the influences of the collected data during these steps on the Modeling entity and thus on the perturbation option. The Modeling entity has a connection to the associated *ModelingActivity* with the property *wasGeneratedByMA*.

The Evaluation step of the reference process will not be discussed further, as it is not implemented in the web-based user interface, but for the sake of completeness, it is included in Figure 5. In the knowledge graph, the *EvaluationEntity* has a recursive property connection and a connection to the *ModelingEntity* named *wasBasedOn*, which is based on wasDerivedFrom from PROV-O and is defined as the construction of a new entity based on a pre-existing entity (W3C, 2013). An *EvaluationEntity* can therefore be a modified version of a *ModelingEntity* or an *EvaluationEntity*. Furthermore, an *EvaluationEntity* has a connection to the *EvaluationActivity* which in turn has a connection to the *ModelingActivity* through the property *wasInformedBy*. This property comes from PROV-O and is defined as the exchange of an entity between two activities, where one activity uses the entity created by the other (W3C, 2013).

The levels of scale are not used exclusively for the classification methods. This means, that this information is gathered for every data mining method equally. Therefore, this information is stored in the generic level of the knowledge graph. The three levels of scales are subclasses of the class *Scale*, as shown in the lower section of Figure 5.

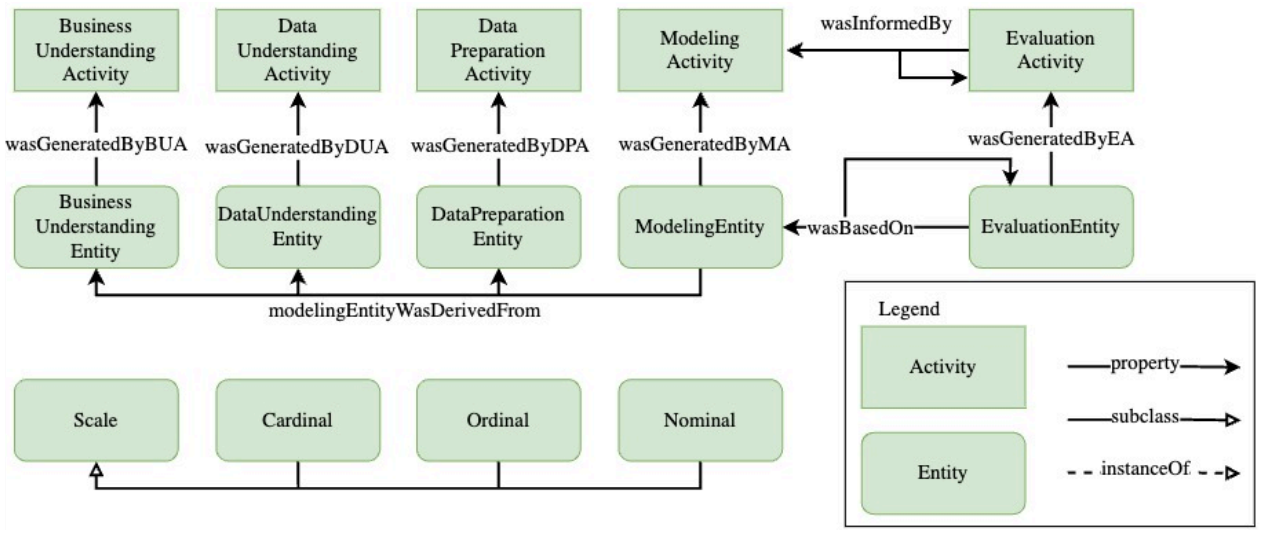


Figure 5 Generic level knowledge graph development view

In the deployment view of the generic level, the knowledge graph is created as shown in Figure 6. The *ModelingEntity* and *EvaluationEntity* created in the development view act as the basis for the *DeploymentEntity*, the connection is created with the property *deploymentEntityWasDerivedFrom.* It should be noted that each *DeploymentEntity* can have several *ModelingEntities* or *EvaluationEntities* as a basis. The *DeploymentEntity* has a connection to the *Case*, which is described with the property *wasAssignedToDeploymentEntity*, which is based on PROV-O *wasDerivedFrom*. In turn, the *DeploymentEntity* has the property *wasGeneratedByDA* with the connection to the *DeploymentActivity*.

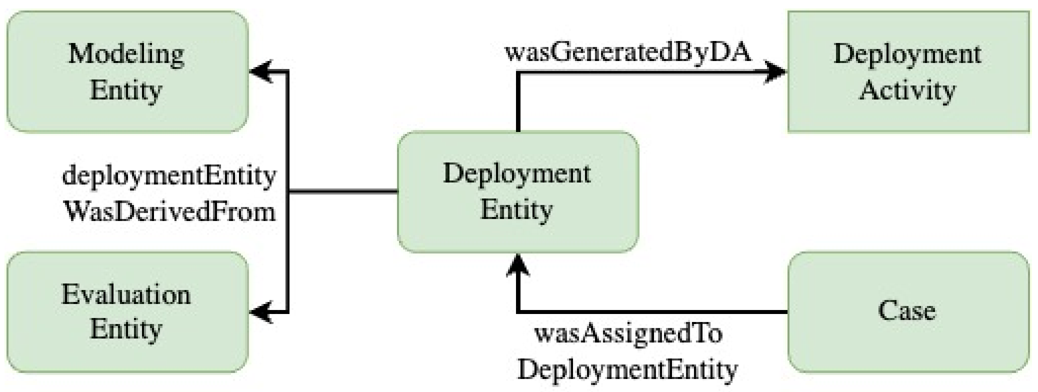


Figure 6 Generic level knowledge graph deployment view

3.6.2. Method level

The method level provides deeper insight into the subclasses of the respective CRISP-DM steps and reference process activities. All aspects addressed in chapters 3.1 - 3.4 are presented as subclasses of the respective entities and activities, illustrated in Figure 7. Starting with *BusinessUnderstandingEntity*, which contains the *AssessmentApproach* as a subclass, which in turn is linked to the *PerturbationApproach*. The *BusinessUnderstandingActivity* is in turn the superclass of the *ChoiceOfAssessmentApproachActivity*.

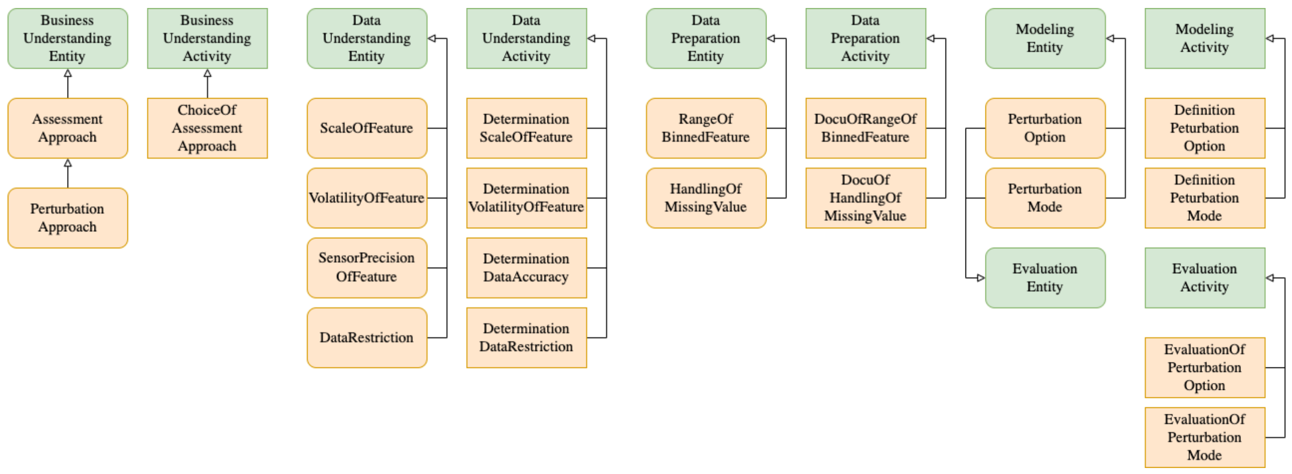

Business Understanding Entity
Assessment Approach
Perturbation Approach
Business Understanding Activity
ChoiceOf Assessment Approach
Data Understanding Entity
ScaleOfFeature
VolatilityOfFeature
SensorPrecision OfFeature
DataRestriction
Data Understanding Activity
Determination ScaleOfFeature
Determination VolatilityOfFeature
Determination DataAccuracy
Determination DataRestriction
Data Preparation Entity
RangeOf BinnedFeature
HandlingOf MissingValue
Data Preparation Activity
DocuOfRangeOf BinnedFeature
DocuOf HandlingOf MissingValue
Modeling Entity
Perturbation Option
Perturbation Mode
Evaluation Entity
Modeling Activity
Definition Petrubation Option
Definition Petrubation Mode
Evaluation Activity
EvaluationOf Perturbation Option
EvaluationOf Perturbation Mode

Figure 7 Method level knowledge graph development view

The *DataUnderstandingEntity* and *DataUnderstandingActivity*, contain the activities and entities subclasses for *ScaleOfFeature*, *DataRestriction*, *VolatilityOfFeature* and *SensorPrecisionOfFeature*. The *DataPreparationEntity* and *DataPreparationActivity* superclasses consist of the activities and entities for *RangeOfBinnedFeature* and *HandlingOfMissingValue*.

The *ModelingEntity* and *ModelingActivity* consist of the subclasses of the activities and entities for *PerturbationMode* and *PerturbationOption*. The *ModelingEntity* subclasses can be shared with the

EvaluationEntity. For the *EvaluationActivity*, separate Evaluation subclasses for *PerturbationOption* and *PerturbationMode* have been created.

The deployment view of the method level is shown in Figure 8. The *DeploymentEntity* consists of the subclass *Assessment* which in turn has the *PerturbationAssessment* entity as a subclass. It can be noted that the *PerturbationOfClassificationCase* is a subclass of the *DeploymentActivity*. This activity is the perturbation of a *ClassificationCase* which is a subclass of *Case*.

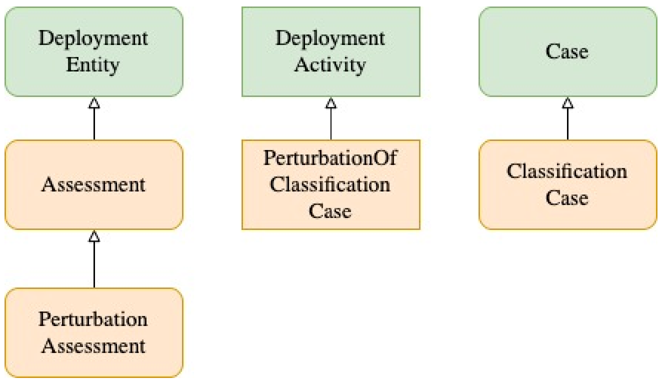
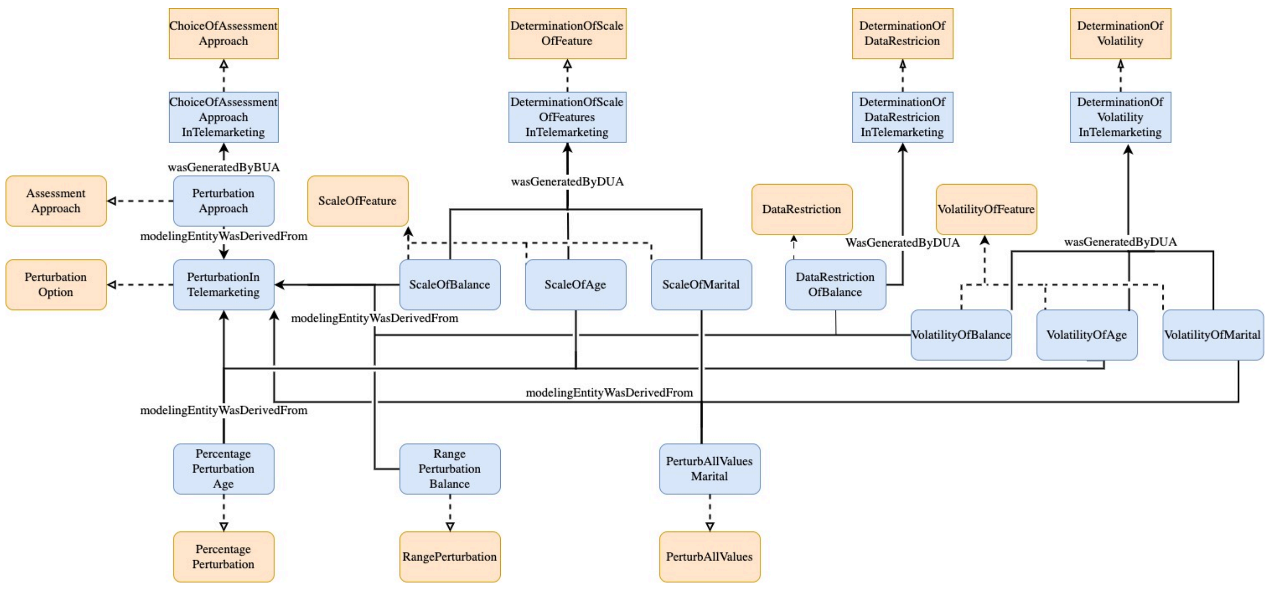


Figure 8 Method level knowledge graph deployment view

3.6.3. Problem level

In the problem level of the knowledge graph, the method level entities are instantiated for the respective use case, which is shown in Figure 9. The example from chapter 3.5, exemplarily perturbed in Figure 4 is visualized here.


ChoiceOfAssessment Approach
DeterminationOfScale OfFeature
DeterminationOf DataRestriction
DeterminationOf Volatility
ChoiceOfAssessment Approach InTelemarketing
DeterminationOfScale OfFeatures InTelemarketing
DeterminationOf DataRestricion InTelemarketing
DeterminationOf Volatility InTelemarketing
wasGeneratedByBUA
wasGeneratedByDUA
Assessment Approach
Perturbation Approach
ScaleOfFeature
DataRestriction
VolatilityOfFeature
modelingEntityWasDerivedFrom
WasGeneratedByDUA
wasGeneratedByDUA
Perturbation Option
PerturbationIn Telemarketing
ScaleOfBalance
ScaleOfAge
ScaleOfMarital
DataRestriction OfBalance
VolatilityOfBalance
VolatilityOfAge
VolatilityOfMarital
modelingEntityWasDerivedFrom
modelingEntityWasDerivedFrom
modelingEntityWasDerivedFrom
Percentage Perturbation Age
Range Perturbation Balance
PerturbAllValues Marital
Percentage Perturbation
RangePerturbation
PerturbAllValues
Figure 9 Problem level knowledge graph development view

A total of three different perturbation options were chosen. It can be seen that all created perturbation options have the predicate *modelingEntityWasDerivedFrom PerturbationApproach* which is an instance of *AssessmentApproach* and *wasGeneratedByBUA ChoiceOfAssessmentApproachInTelemarketing* which is an instance of *ChoiceOfAssessmentApproach*.

The *PercentagePerturbationAge* is an instantiation of the *PercentagePerturbation* option and *modelingEntityWasDerivedFrom* connects the perturbation option with the entities for *ScaleOfAge* and *VolatilityOfAge*. The volatilities of all perturbation options are linked with the property *wasGeneratedByDUA* to *DeterminationOfVolatilityInTelemarketing*, which in turn is an instance of the activity *DeterminationOfVolatility*. Furthermore, all scales of the features are linked with the property *wasGeneratedByDUA* to *DeterminationOfScaleInTelemarketing*, which in turn is an instance of the activity *DeterminationOfScaleOfFeature*.

The *RangePerturbationBalance* is an instantiation of the *RangePerturbation* option and *modelingEntityWasDerivedFrom* connects the perturbation option with the entities for *ScaleOfBalance*, *VolatilityOfBalance* and *DataRestrictionOfBalance*. The data restriction is an instance of the entity *DataRestriction* and *wasGeneratedByDUA* by the activity *DeterminationOfDataRestrictionInTelemarketing*, which in turn is an instance of the activity *DeterminationOfDataRestriction*.

The *PerturbAllValuesMarital* is an instantiation of the *PerturbAllValues* perturbation option and *modelingEntityWasDerivedFrom* connects the perturbation option with the entities for *ScaleOfMarital* and *VolatilityOfMarital*.

Finally, the Modeling entity contains the connection *modelingEntityWasDerivedFrom* to the *PerturbationApproach* entity, which was created during Business Understanding and is an instance of the *AssessmentApproach*.

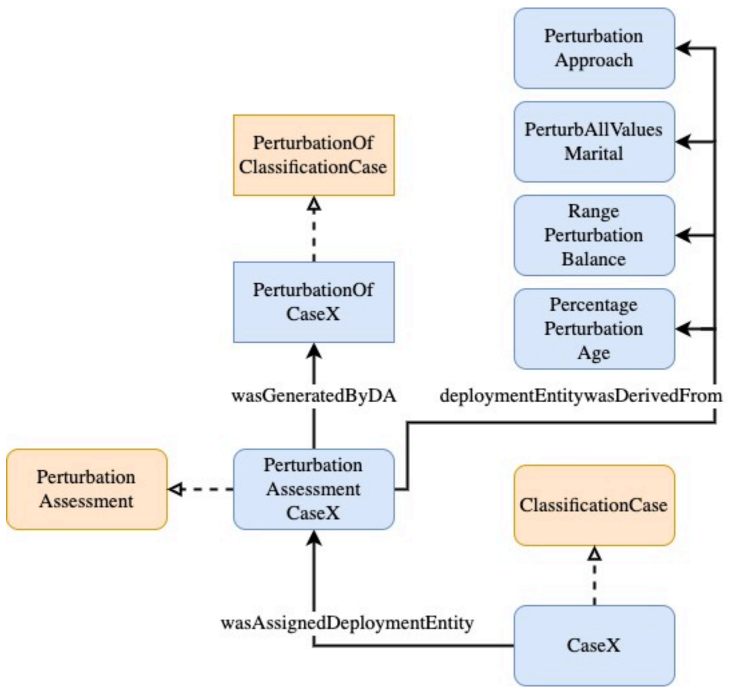

Perturbation Approach
PerturbAllValues Marital
Range Perturbation Balance
Percentage Perturbation Age
PerturbationOf ClassificationCase
PerturbationOf CaseX
wasGeneratedByDA
deploymentEntitywasDerivedFrom
Perturbation Assessment
Perturbation Assessment CaseX
ClassificationCase
wasAssignedDeploymentEntity
CaseX

Figure 10 Problem level knowledge graph deployment view

Figure 10 shows the deployment view of the knowledge graph at the problem level. A *ClassificationCase* named *CaseX* is instantiated which is assigned to the entity *PerturbationAssessmentCaseX* via the property *wasAssignedDeploymentEntity*. This entity is an instance of the *PerturbationAssessment* and *wasGeneratedByDA PerturbationOfCaseX* which in turn is an instance of *PerturbationOfClassificationCase*. The *PerturbationAssessmentX* entity was derived from the 3 perturbation options: *PercentagePerturbationAge*, *RangePerturbationBalance* and *PerturbAllValuesMarital*.

4. Architecture

The web-based user interface was written in Python using the Streamlit framework (Streamlit). This framework makes it possible to write web-based user interfaces by providing basic functions of a web app and therefore do not need to be created by the developer. For example, it is possible to render text elements, and display elements such as tables or chart elements. Furthermore, input widgets such as buttons, select boxes, multiselects, sliders, text, and number input as well as download and upload buttons can be called via implemented methods.

The corresponding storage is an RDF triple store that communicates with the web-based user interface and stores the necessary information in a knowledge graph. The procedure for using a triple store typically involves the following steps:

Loading RDF data

The first step is to load the RDF data into the triple store. This can be done by importing RDF files, connecting to a SPARQL endpoint, or using APIs to load the data programmatically.

Querying the data

Once the RDF data is loaded into the triple store, SPARQL can be used to query the data and retrieve specific information. SPARQL is a query language for RDF data and allows complex questions to be asked about the data and the results retrieved in a structured format, as already mentioned in chapter 2.3.

Modifying the data

SPARQL can be used to update RDF data stored in the triple store. This includes adding, changing, or deleting triples.

For this web-based user interface, Apache Jena Fuseki was chosen. Apache Jena Fuseki is an open-source framework for serving and managing RDF data (Foundation, 2023). Jena Fuseki has a modular architecture, allowing it to be used in different deployment scenarios. It includes a SPARQL endpoint, which provides access to the RDF data through the SPARQL query language, as well as an administrative interface for managing the data. Jena Fuseki is commonly used in the areas of knowledge management, semantic web-based user interfaces and linked data. Its main objective is to provide a scalable, efficient, and flexible solution for serving and managing RDF data, making it easier to build and maintain semantic web-based user interfaces.

The communication between the web-based user interface and the triple store takes place via ARQ, a query engine for Jena that supports the RDF query language SPARQL. The general communication process is as follows. The web-based user interface sends a SPARQL query to the Apache Jena Fuseki server via the SPARQL protocol. The protocol is a standard for accessing SPARQL queries and updates to a SPARQL processing service and returning the results via HTTP. The Apache Jena Fuseki server receives the SPARQL query and processes it by executing the query against the RDF data stored in the triple store. Subsequently, the Apache Jena Fuseki server returns the results of the SPARQL query to the web-based user interface in the desired format, e.g., JSON (Java Script object Notation). JSON is a human- and machine-readable data format for data exchange between web-based systems (ECMA, 2017). The results of the query can then be used in the web-based user interface to display information to the user or to perform further processing.

The web-based user interface is structured in two views as defined in the reference process. A distinction is made between the *Developer View*, which is allowed to see the development and deployment view of the reference process, and the *Analyst View*, which is only allowed to see the deployment view. Authorization is based on a Streamlit authentication module that enables the validation of an *Actor* in a Streamlit app. Furthermore, new users could be added. Figure 11 summarizes the architecture of the web-based user interface. An *Actor*, which must authenticate itself as a developer or analyst and is given access to the respective view. The views in turn access the Apache Fuseki triple store. The frontend is called up via a browser.

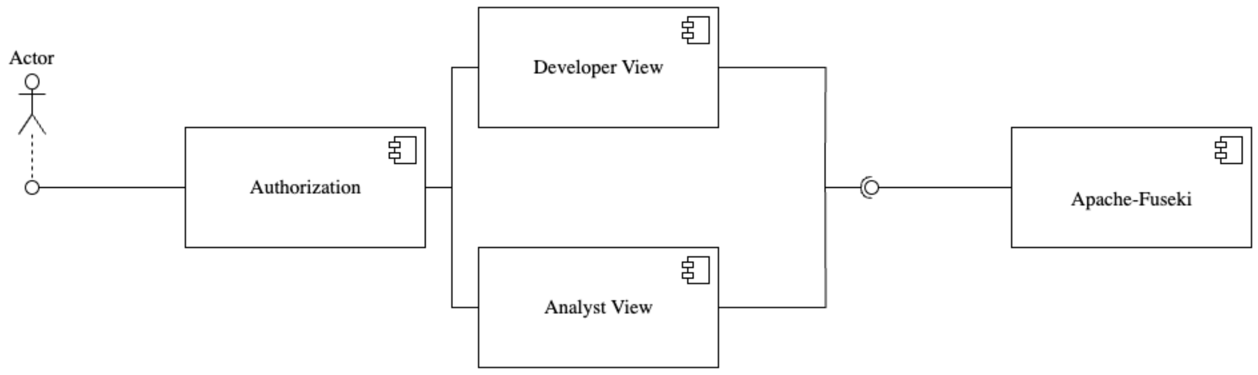


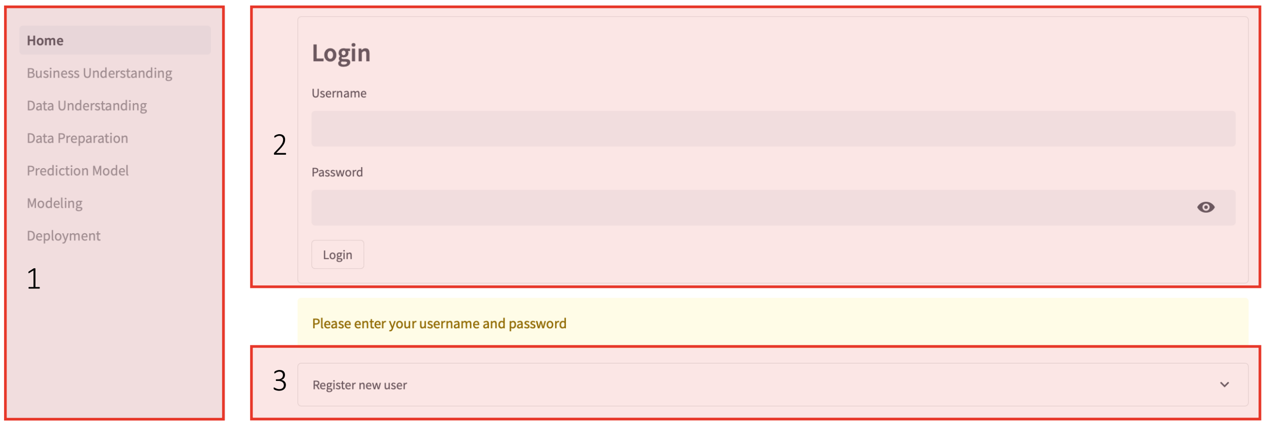
Figure 11 Architecture web-based user interface

5. User perspective

The user can switch between the different steps of the reference process via the pages when opening the web-based user interface with a browser. This chapter explains the pages of the web-based user interface and their functionality. Every time the web-based user interface is accessed, the user is presented with the *Home* page first, which is described in chapter 5.1, followed by the *Business Understanding* page in chapter 5.2, and *Data Understanding* page in chapter 5.3. In chapter 5.4, the *Data Preparation* page is presented. Chapter 5.5 contains the *Prediction Model* page. Chapter 5.6 describes the *Modeling* page. Chapter 5.7 describes the *Deployment* page.

5.1. Home

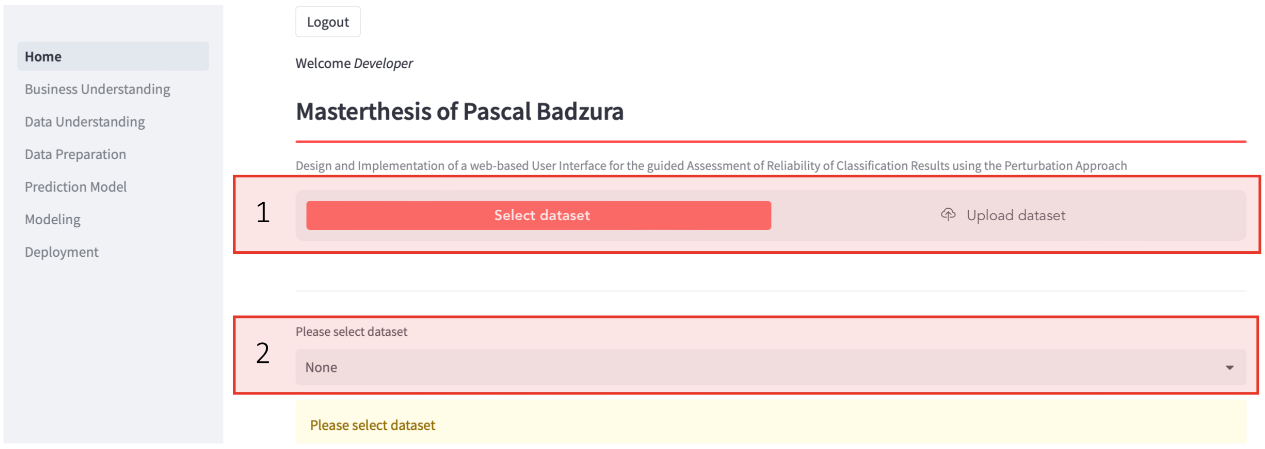
The *Home* page is the starting point for the web-based user interface, shown in Figure 12. The sidebar on the left in the highlighted area 1 is used to navigate between the pages. The user is authenticated via a login field (highlighted area 2) and depending on the authentication, obtains the respective view of the web-based user interface. There is a developer and an analyst in the scenario of the master thesis. If logged into the web-based user interface via the developer profile, access is granted to all pages. The analyst can only access the *Home*, *Prediction Model* and *Deployment* pages. On the restricted pages, the user is informed that he should call up the *Deployment* page, as can be seen in Figure 16 and Figure 18. This distinction was implemented to integrate the development view and the deployment view of the knowledge graph, described in chapter 4, into the web-based user interface. The developer gets the development and deployment view, and the analyst gets the deployment view. In addition, new users can be created via the expander in the lower area of Figure 12, visible in the highlighted area 3.



Please enter your username and password

Figure 12 Home - login form

After authentication as a developer has been successfully completed, the user interface changes as illustrated in Figure 13. The user has two options to continue. If no dataset is uploaded yet, the user must click on the tab *Upload dataset* in the highlighted area 1. If a dataset is already uploaded the user may select and load the data from the Apache Jena Fuseki server via a selection box, visible in the highlighted area 2.



Welcome *Developer*

Masterthesis of Pascal Badzura

Design and Implementation of a web-based User Interface for the guided Assessment of Reliability of Classification Results using the Perturbation Approach

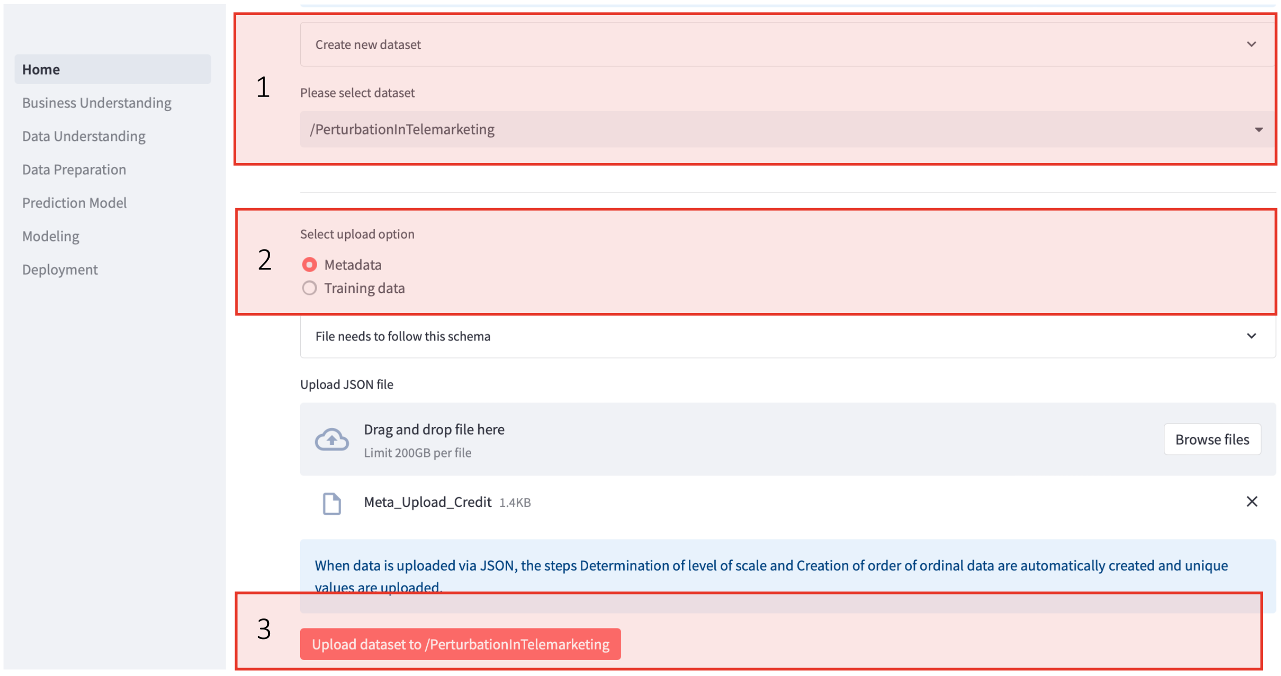
Please select dataset

Please select dataset

Figure 13 Home - development view database selection

If the tab *Upload dataset* in the highlighted area 1 in Figure 13 is selected, the user interface is displayed as shown in Figure 14. If no corresponding dataset exists on the Apache Jena Fuseki server, it can be created under *Create new dataset* (highlighted area 1) and then selected with the select box *Please select dataset*.

Once a dataset is selected, a radio button appears showing the upload options, as can be seen in the highlighted area 2 in Figure 14. A developer can decide between a metadata (I) and a training data (II) upload. A metadata (I) upload must contain the feature names, their scale levels as well as the minimum and maximum of the values for cardinal data, the order of the unique values for ordinal data and all unique values in random order for nominal data. The following scheme and naming shall be followed in Code 1. The information about the unique values per feature is needed so that the perturbation options only create feature values present in the training data.



Please select dataset

/PerturbationInTelemarketing

Select upload option

[✓ Metadata]

Training data

File needs to follow this schema

Upload JSON file

Drag and drop file here

Limit 200GB per file

Meta_Upload_Credit 1.4KB

When data is uploaded via JSON, the steps Determination of level of scale and Creation of order of ordinal data are automatically created and unique values are uploaded.

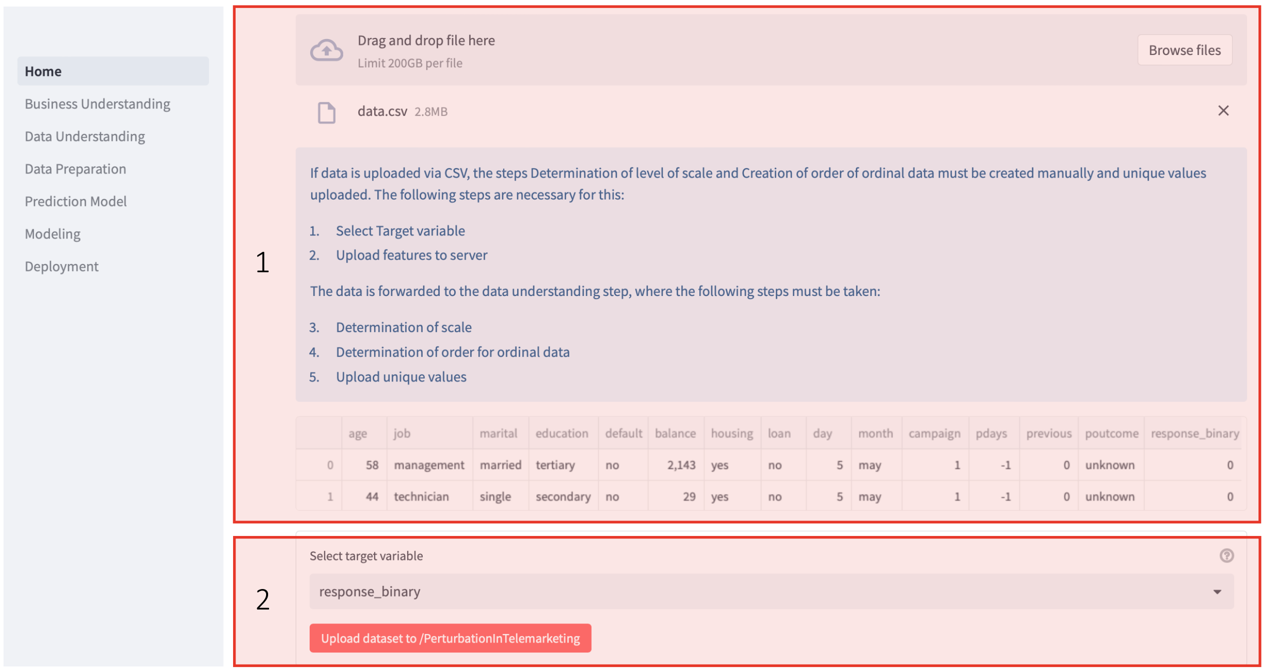
Figure 14 Home - development view upload dataset

Code 1 Example metadata upload JSON

```
{
  "Feature1": {"levelOfScale": "Cardinal",   "uniqueValues": [min, max]},
  "Feature2": {"levelOfScale": "Nominal",    "uniqueValues": ["x", "y", …]},
  "Feature3": {"levelOfScale": "Ordinal",    "uniqueValues": ["1", "2", …]}
}
```

Using the *Upload dataset to /PerturbationInTelemarketing* button in the highlighted area 3 in Figure 14, the uploaded data is written to the Apache Jena Fuseki server and the selection of the level of scale, the definition of the order of the ordinal data as well as the upload of the unique values, visualized in Figure 15, Figure 19, Figure 20, Figure 22 and Figure 23, are no longer necessary. The developer will immediately be forwarded to the view in Figure 24.

If the training data (II) upload in the marked area 2 in Figure 14 is chosen, the user interface is displayed as shown in Figure 15. The upload field and a table with the data values can be seen in the highlighted area 1. The *Upload dataset to /PerturbationInTelemarketing* button in the highlighted area 2 can be used to set the target variable of the model. The information of the target variable is needed to train a model on the *Prediction Model* page. It should be noted that training a model with a metadata (I) upload is not possible, since the training data is not available. When the dataset is uploaded to the Apache Jena Fuseki server, the developer is automatically directed to the *Data Understanding* page, described in Chapter 5.3, to perform the steps of selecting the level of scale, creating the order of ordinal data, and uploading the unique values.



If data is uploaded via CSV, the steps Determination of level of scale and Creation of order of ordinal data must be created manually and unique values uploaded. The following steps are necessary for this:

1. Select Target variable
2. Upload features to server

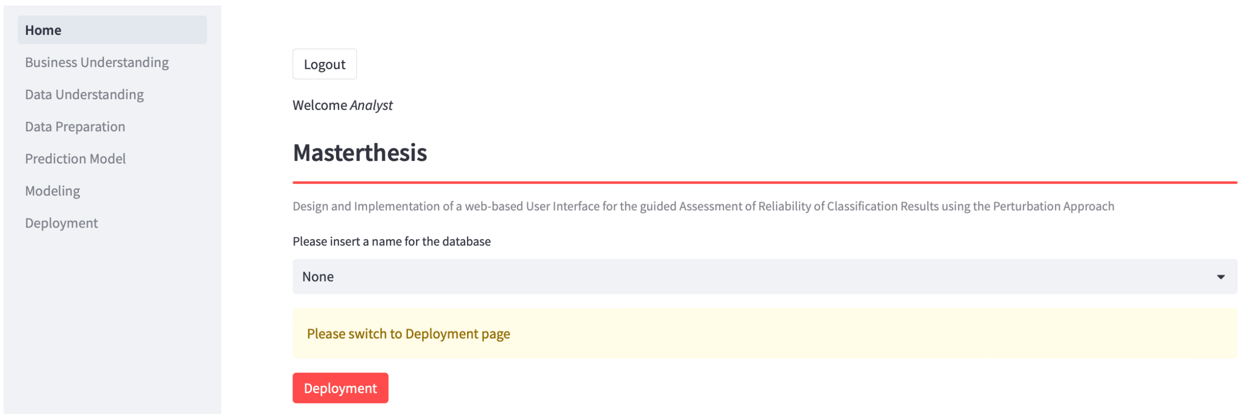
The data is forwarded to the data understanding step, where the following steps must be taken:

3. Determination of scale
4. Determination of order for ordinal data
5. Upload unique values

	age	job	marital	education	default	balance	housing	loan	day	month	campaign	pdays	previous	poutcome	response_binary
0	58	management	married	tertiary	no	2,143	yes	no	5	may	1	-1	0	unknown	0
1	44	technician	single	secondary	no	29	yes	no	5	may	1	-1	0	unknown	0

Figure 15 Home - development view Training data upload

The view of an analyst differs from that of a developer because an analyst. Figure 16 shows the analyst view of the *Home* page. He is given the option to select a dataset via the select box and is prompted to switch to the *Deployment* page.



Welcome *Analyst*

Masterthesis

Design and Implementation of a web-based User Interface for the guided Assessment of Reliability of Classification Results using the Perturbation Approach

Please insert a name for the database

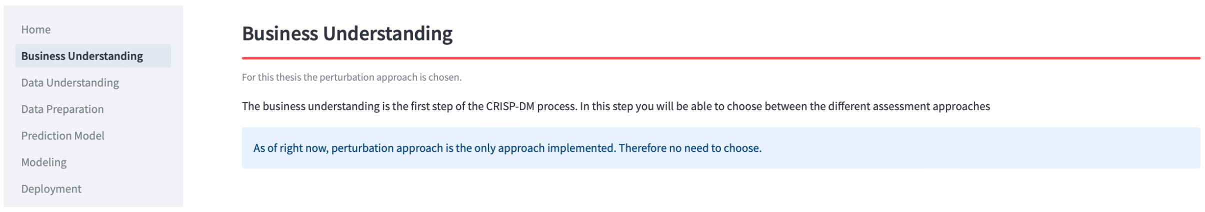
None

Please switch to Deployment page

Figure 16 Home - database selection deployment view

5.2. Business Understanding

This chapter addresses the Business Understanding step which contains selecting the assessment approach. As only the perturbation approach is considered in this master thesis, only the perturbation approach is described on this page and no selection can be made. Figure 17 shows the development view of the *Business Understanding* page.



Business Understanding

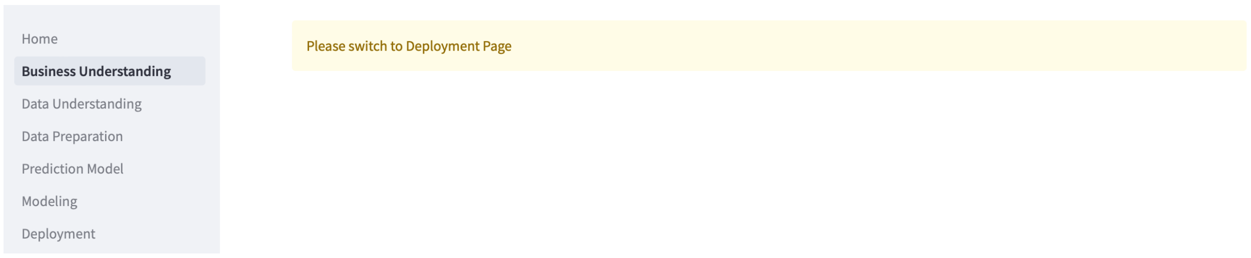
For this thesis the perturbation approach is chosen.

The business understanding is the first step of the CRISP-DM process. In this step you will be able to choose between the different assessment approaches

As of right now, perturbation approach is the only approach implemented. Therefore no need to choose.

Figure 17 Business Understanding - development view

If the user is authenticated as an analyst, the *Business Understanding* page will look like Figure 18. This information is repeated on the *Data Understanding*, *Data Preparation* and *Modeling* page for an analyst. The analyst is given the message *Please switch to Deployment Page*, as this page is in the development view of the reference process.



Please switch to Deployment Page

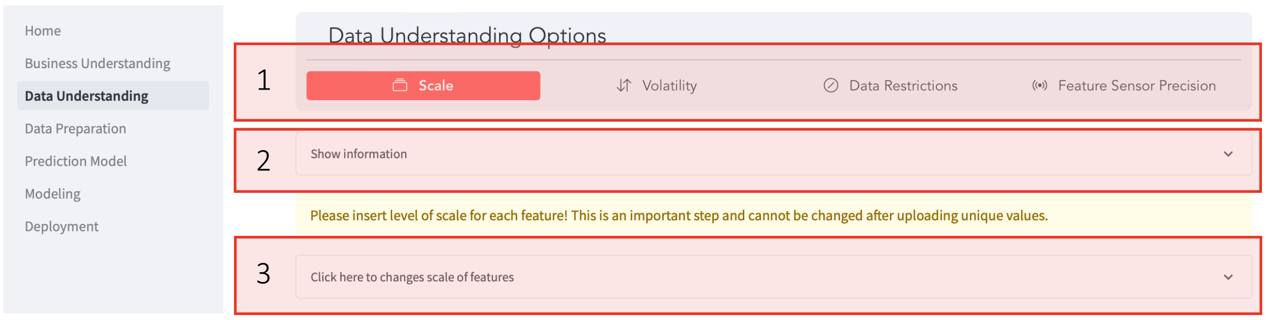
Figure 18 Business Understanding - deployment view

5.3. Data Understanding

This section shows the user interface for a developer, an analyst does not have access to these pages and is redirected to the *Deployment* page. This chapter covers the Data Understanding steps, when uploading training data (II) is selected the level of scale of the features must be set, and the unique values must be uploaded in chapter 5.3.1. The volatility of the individual features can be set in chapter 5.3.2. In chapter 5.3.3 the developer can enter data restrictions and chapter 5.3.4 addresses the feature sensor precision of cardinal features. The user can navigate between the tabs *Scale*, *Volatility*, *Data Restrictions* and *Feature Sensor Precision* activities using the toolbar shown in the highlighted area 1 in Figure 19.

5.3.1. Scale and unique values

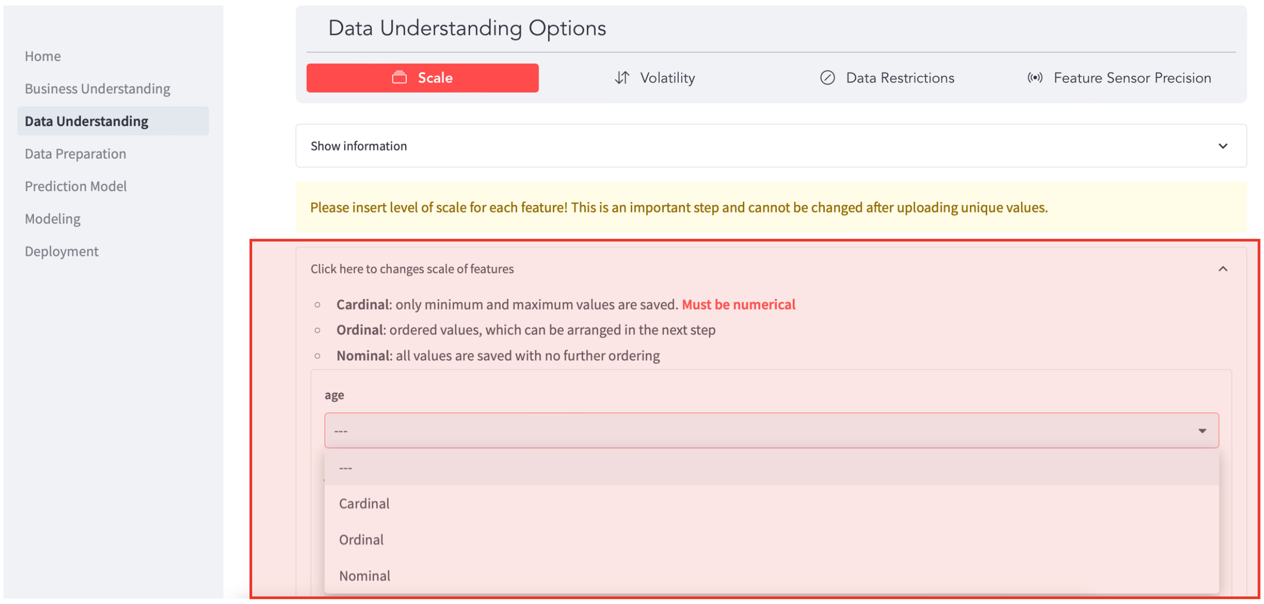
After the training data (II) is uploaded, the user interface is displayed as shown in Figure 19. In Figure 19 the *Scale* tab is selected. The first expander on *Show information* (marked area 2) contains information about missing Data Understanding activities. The second expander *Click here to change scale of features* (marked area 3) is used to determine the scale of each feature.



Please insert level of scale for each feature! This is an important step and cannot be changed after uploading unique values.

Figure 19 Data Understanding - development view after training data upload

Figure 20 shows the expander for setting the level of scale for each feature where the developer can choose between *Cardinal*, *Ordinal* and *Nominal* in the highlighted area. If no level of scale is selected for a feature or a cardinal feature contains non-numeric values, an error message will be displayed, as shown in Figure 21.



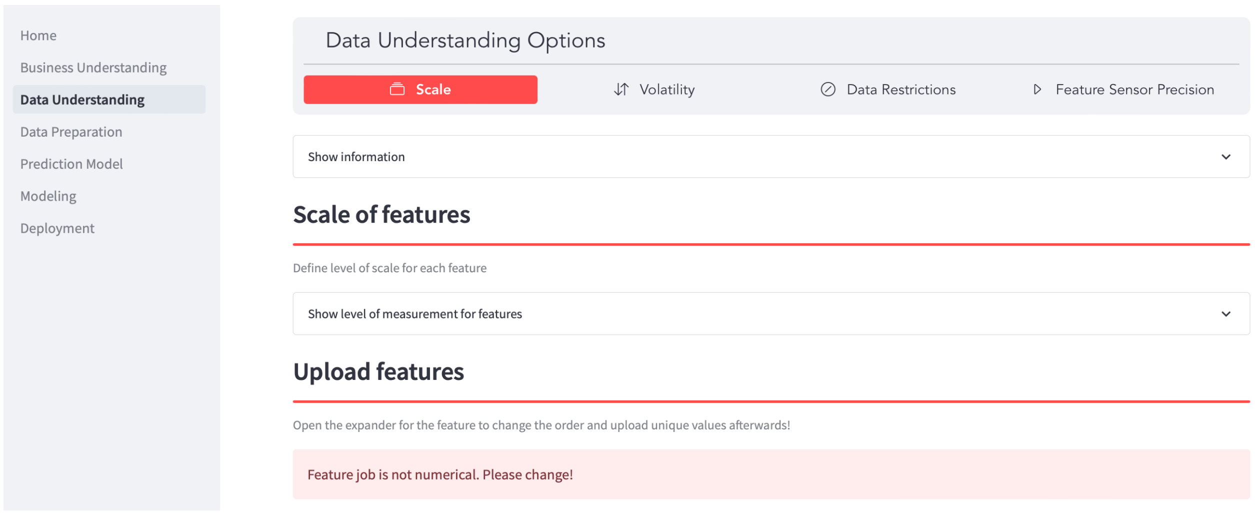
Please insert level of scale for each feature! This is an important step and cannot be changed after uploading unique values.

Click here to changes scale of features

- **Cardinal**: only minimum and maximum values are saved. **Must be numerical**
- **Ordinal**: ordered values, which can be arranged in the next step
- **Nominal**: all values are saved with no further ordering

age

Figure 20 Data Understanding - development view selection of the level of measurement



Scale of features

Define level of scale for each feature

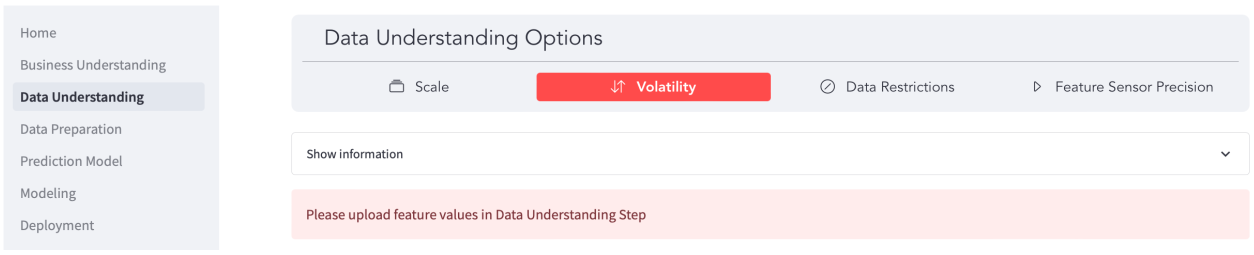
Upload features

Open the expander for the feature to change the order and upload unique values afterwards!

Feature job is not numerical. Please change!

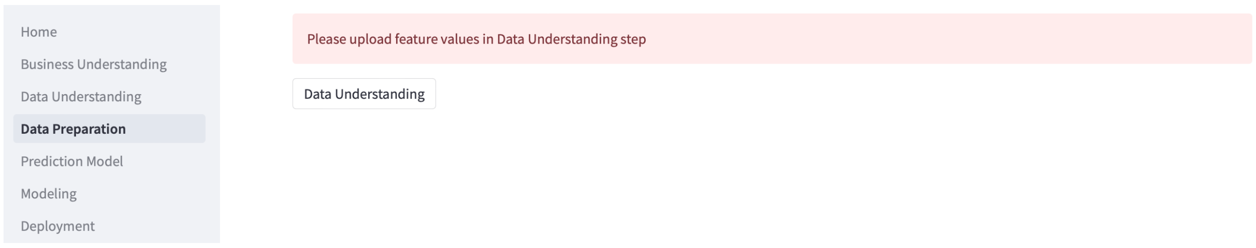
Figure 21 Data Understanding - development view error message level of measurement

If the setting of scale and the uploading of the unique values is not performed correctly, the error messages in Figure 22 will be shown for any other Data Understanding activities and Figure 23 is displayed on the other pages and further steps cannot be performed. This was implemented because the setting of the scale and the upload of the unique values of the data are mandatory steps without which the web-based user interface will not work, because other activities depend on this information. For example, different perturbation options are available for each level of scale. The information of unique values for each feature is important for determining how the feature can be perturbed. For example, nominal and ordinal features are only allowed to take values present in the data. Cardinal features are allowed to be in the range between the lowest and highest available training value.



Please upload feature values in Data Understanding Step

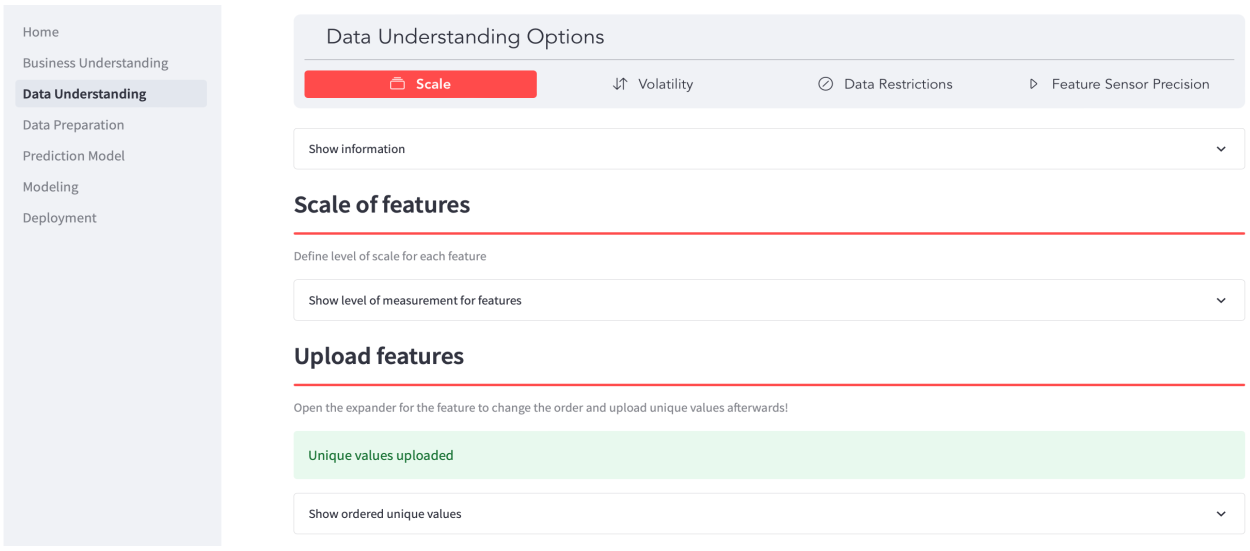
Figure 22 Data Understanding - development view error missing unique values



Please upload feature values in Data Understanding step

Figure 23 Data Preparation - development view error missing unique values

The levels of scale can be changed until the upload of the unique values is completed, a later change is not possible. This design decision was made because the levels of scale do not change historically and furthermore different characteristics for cardinal, ordinal and nominal features are saved. For cardinal features, only the minimum and maximum values are stored and for ordinal features, all ordered values are stored and for nominal features, all values are stored without any order. After successfully uploading the unique values, the user interface is displayed as shown in Figure 24.



Scale of features

Define level of scale for each feature

Upload features

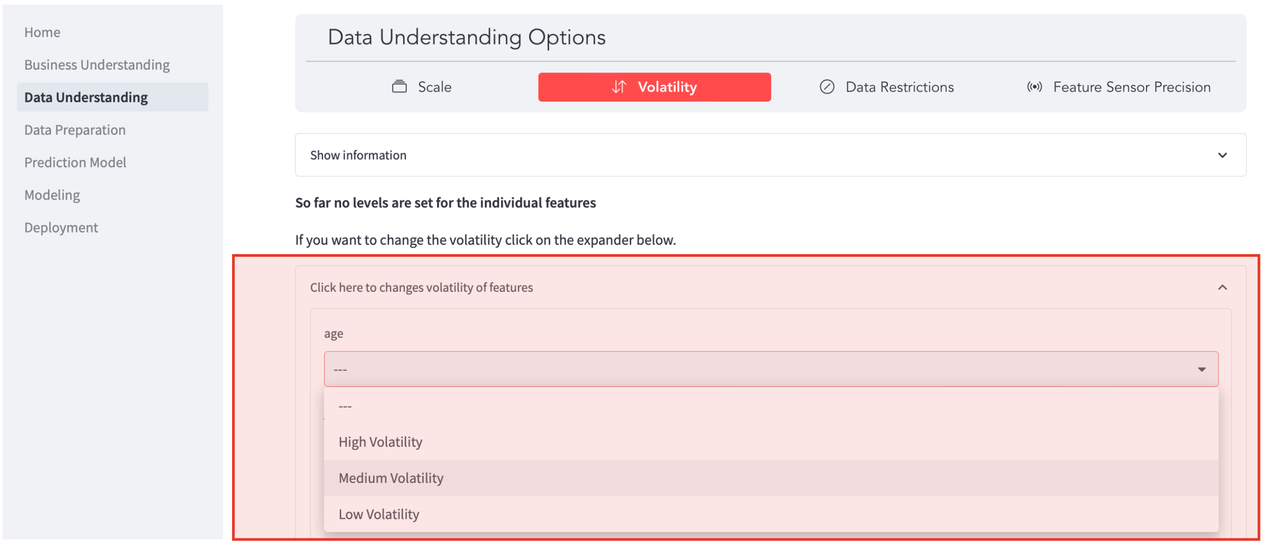
Open the expander for the feature to change the order and upload unique values afterwards!

Unique values uploaded

Figure 24 Data Understanding - development view successfully uploaded unique values

5.3.2. Volatility

In the *Volatility* tab volatility levels can be selected for each feature with the option's *Low volatility, Medium volatility* and *High volatility*, which is illustrated in the marked area in Figure 25. If a volatility level is to be saved, it must be set for all features. It is possible to skip this activity and not set any volatility levels. The classification between *Low volatility, Medium volatility* and *High volatility* does not follow any standard and depends on the use case. Once the volatility levels have been uploaded, they can be cleared and reset at a later stage. In this case, the old volatility levels become invalid for new perturbation options and are only available for perturbation options that have already been set with these volatility levels. This design decision was made because of the cyclical nature of CRISP-DM.



So far no levels are set for the individual features

If you want to change the volatility click on the expander below.

Click here to changes volatility of features

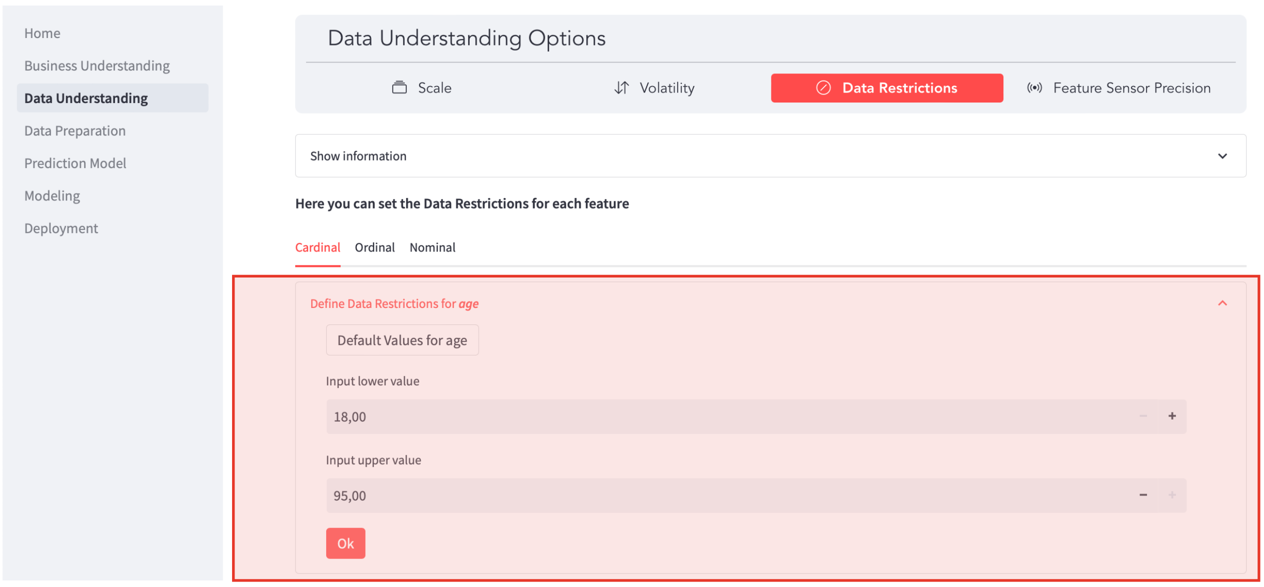
age

High Volatility
Medium Volatility
Low Volatility

Figure 25 Data Understanding - determination of volatility

5.3.3. Data restriction

If there are data restrictions, the developer can declare them in the *Data Restriction* tab, it is possible to skip this activity and not set any data restrictions if there are none. For cardinal data, data restrictions are determined via two number input fields, which must be confirmed with an *Ok* button. The lower and upper values represent the minimum and the maximum value of the respective feature, as exemplified in the marked area in Figure 26 for the feature age. This information was saved during the upload of unique values of the data. The minimum value saved for the age feature is 18 and the maximum value saved is 95. The *Default Values for age* button can be used to set the default values. If the developer presses the *Ok* button, he will be informed whether his values have been accepted and that the data restriction must be uploaded to the Apache Jena Fuseki server in an additional step.



Here you can set the Data Restrictions for each feature

Define Data Restrictions for *age*

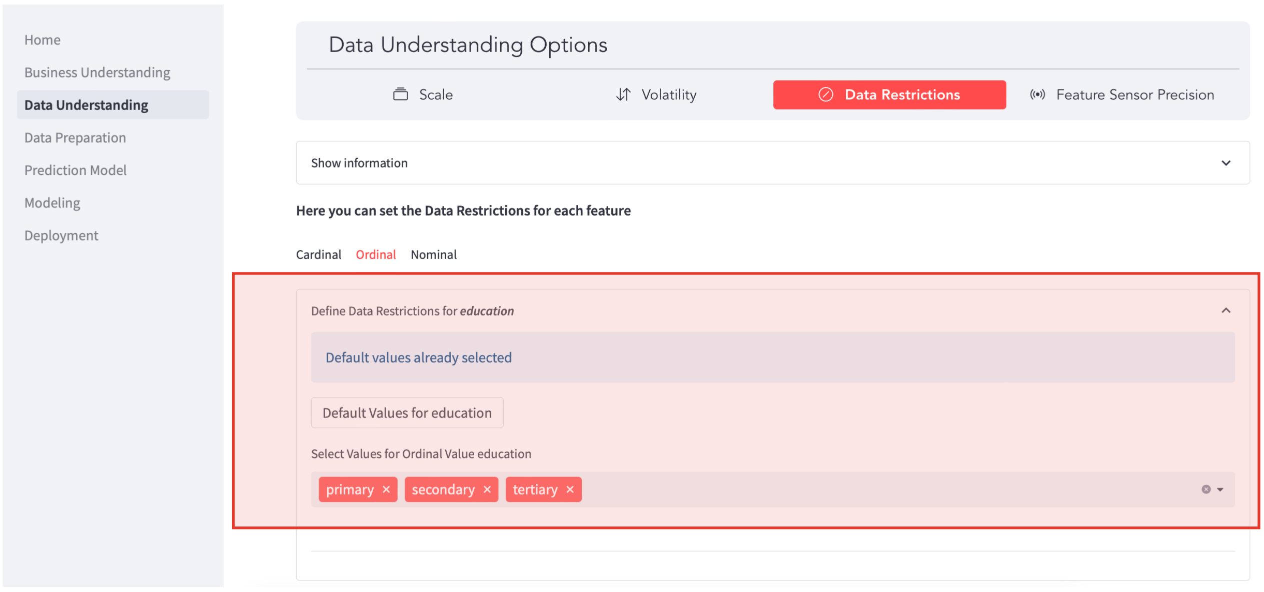
Input lower value: 18,00

Input upper value: 95,00

Figure 26 Data Understanding - determination of data restriction cardinal feature

In the case of ordinal and nominal features, a multiple selection field shows up in which all saved unique values are pre-entered, as the highlighted area in Figure 27 illustrates for the feature education. The user has the option to filter out unwanted values from the ordered values *primary*, *secondary,* and *tertiary*. If the previously restricted education feature should be reset, the feature can be set back to the default value via *Default Values for education*.

As soon as a restriction is set for a feature, an *Upload data restrictions* button appears, which must be pressed to write the data restriction to the RDF triple store, as well as an expander with the restrictions set per feature to summarize the data restrictions.



Here you can set the Data Restrictions for each feature

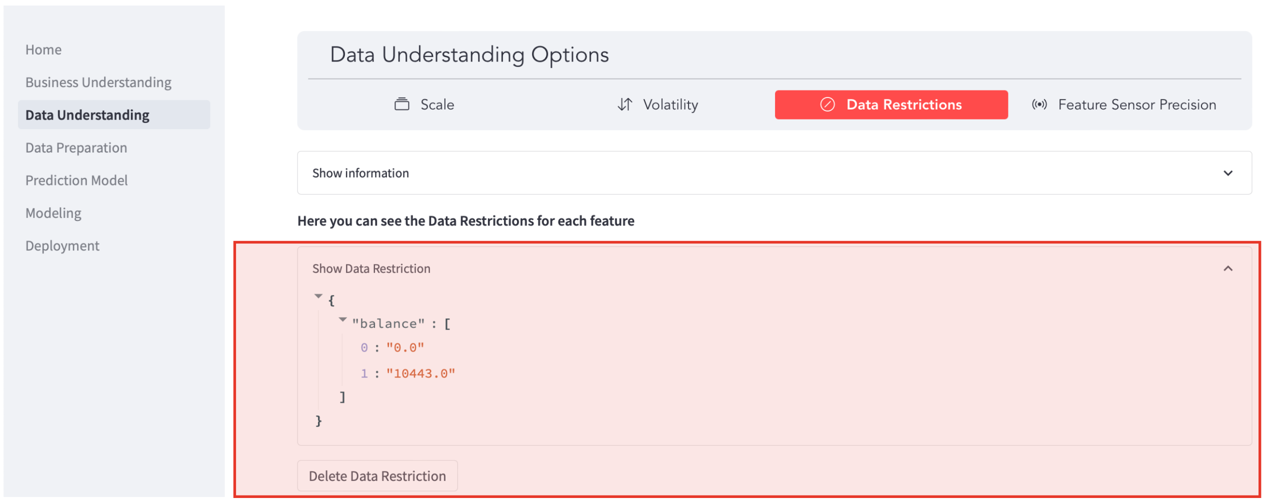
Define Data Restrictions for *education*

Default values already selected

Select Values for Ordinal Value education: primary, secondary, tertiary

Figure 27 Data Understanding - determination of data restriction ordinal feature

If a data restriction is uploaded, the developer will see a *Show data restriction* expander and a *Delete data restriction* button highlighted in Figure 28. As example a data restriction for the feature balance was defined. If this button is pressed, the data restrictions remain in the triple store and can therefore still be used for older perturbation options that contain this information but become invalid for new perturbation options. A new data restriction can now be defined for new perturbation options.



Here you can see the Data Restrictions for each feature

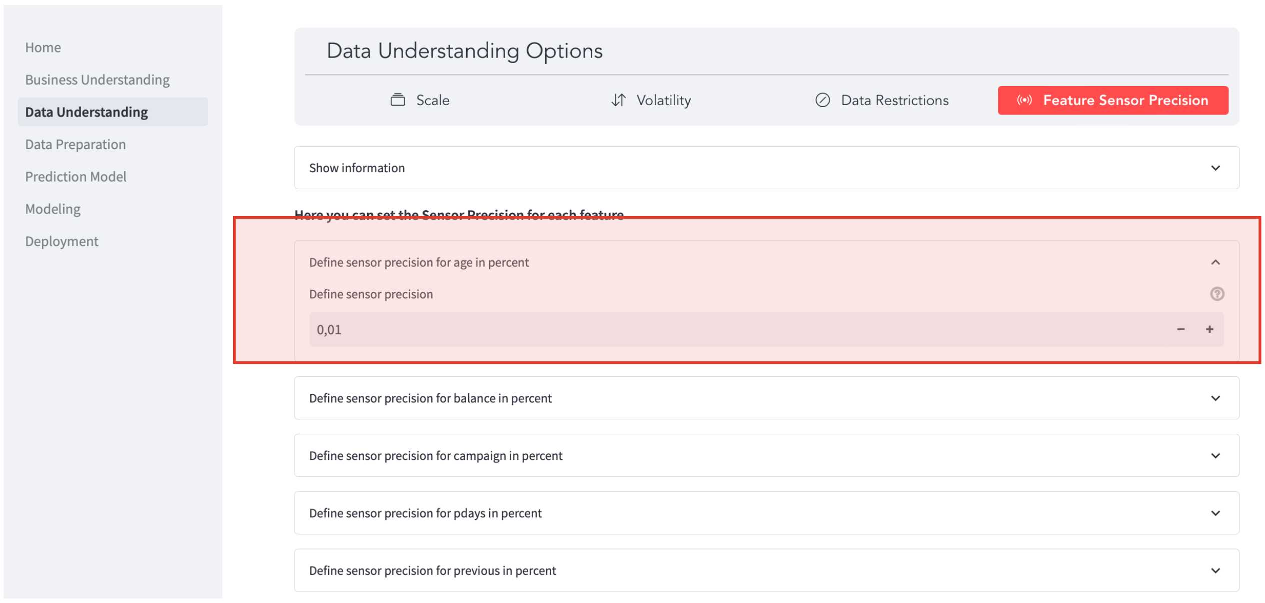
Show Data Restriction

```
{
  "balance" : [
    0 : "0.0"
    1 : "10443.0"
  ]
}
```

Figure 28 Data Understanding - example for data restriction

5.3.4. Feature sensor precision

In the *Feature Sensor Precision* tab, the developer has the option to set a sensor precision for cardinal features, illustrated in the highlighted area in Figure 29. As soon as a feature sensor's precision is greater than 0, it can be stored. There is the option to invalidate the existing sensor precision and create a new one. In this case, the old sensor precision remains in the triple store for older perturbation options that contain this information and cannot be used for new perturbation options. Once a sensor precision has been set for a feature, the feature can be perturbed using Sensor Precision Perturbation option.



Here you can set the Sensor Precision for each feature

Define sensor precision for age in percent

Define sensor precision: 0,01

Define sensor precision for balance in percent

Define sensor precision for campaign in percent

Define sensor precision for pdays in percent

Define sensor precision for previous in percent

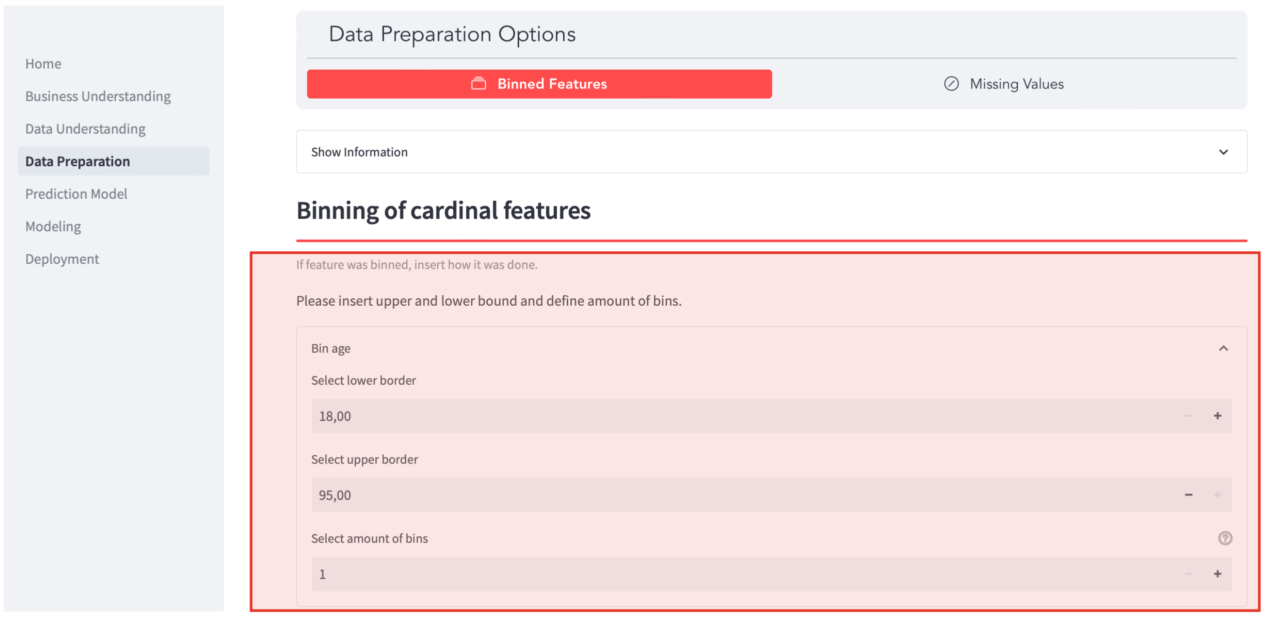
Figure 29 Data Understanding - determination of feature sensor precision

5.4. Data Preparation

This chapter shows the user interface for a developer, an analyst has no access to these pages and is referred to the *Deployment* page. Chapter 5.4.1 addresses the binning for the respective cardinal features. Chapter 5.4.2 addresses the documentation of the missing values.

5.4.1. Binning

If binning of the cardinal data was performed during the Data Preparation step, this can be documented here. Equal Width binning, explained in chapter 3.2, is implemented in this master thesis. Therefore, the user is given the option to enter a lower and upper border as well as the number of bins. By default, the currently valid data restrictions of the cardinal data are set as the lower border and upper border, these cannot be undercut or exceeded. The highlighted area in Figure 30 shows an example of the user interface for creating bins for the feature age. As soon as the *Select amount of bins* input number is greater than one, the binning is cached and available for upload. The maximum number of bins is not limited. Once a feature has been binned, the Bin Perturbation can be used. If binning information is deleted it remains in the triple store for older perturbation options that contain this information and cannot be used for new perturbation options.



Binning of cardinal features

If feature was binned, insert how it was done.

Please insert upper and lower bound and define amount of bins.

Bin age

Select lower border

18,00

Select upper border

95,00

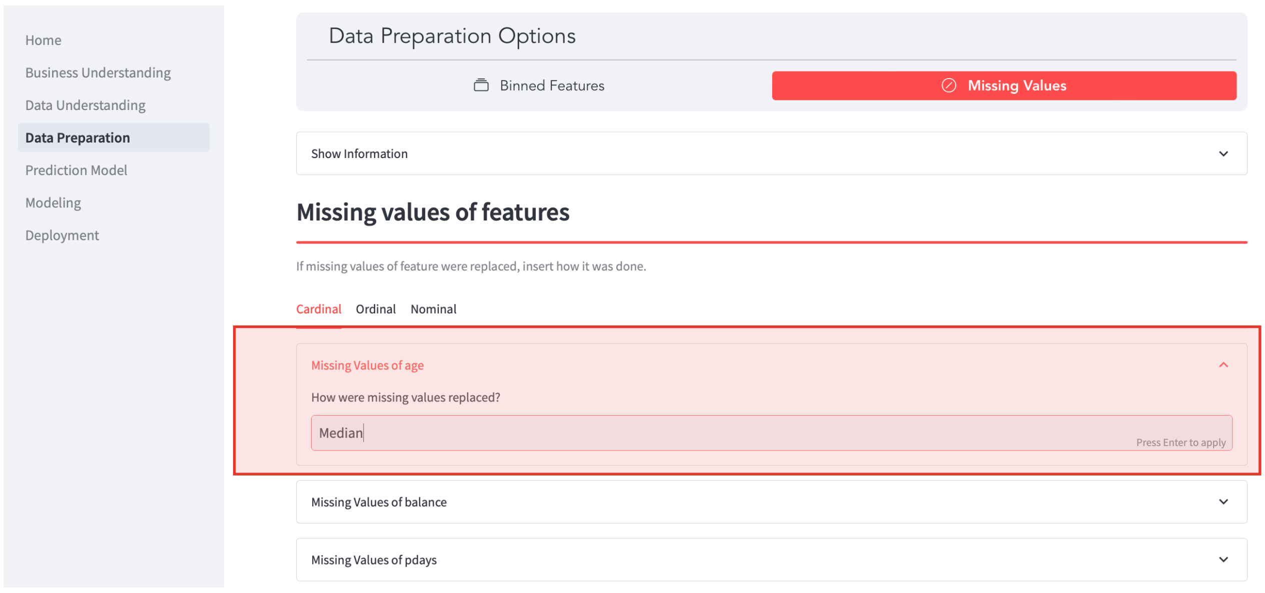
Select amount of bins

1

Figure 30 Data Preparation - determination of binned features

5.4.2. Missing values

If missing values for a feature were replaced during the Data Preparation step, the input field from Figure 31 can be used to document how missing values were replaced. Since there are many different techniques for handling missing values, a text field was chosen as the input mask. The length of the text is not limited. Furthermore, this information is only used as a hint for the perturbation option, as it has no direct influence on the available selection of perturbation option for the features, as explained in chapter 3. The marked area in Figure 31 illustrates how the documentation of replaced missing values can be performed in the user interface. Deleted information about the replacement of missing values are remaining for in the triple store for perturbation options created with this information. New perturbation options cannot be created with this information.



Missing values of features

If missing values of feature were replaced, insert how it was done.

Missing Values of age

How were missing values replaced?

Median

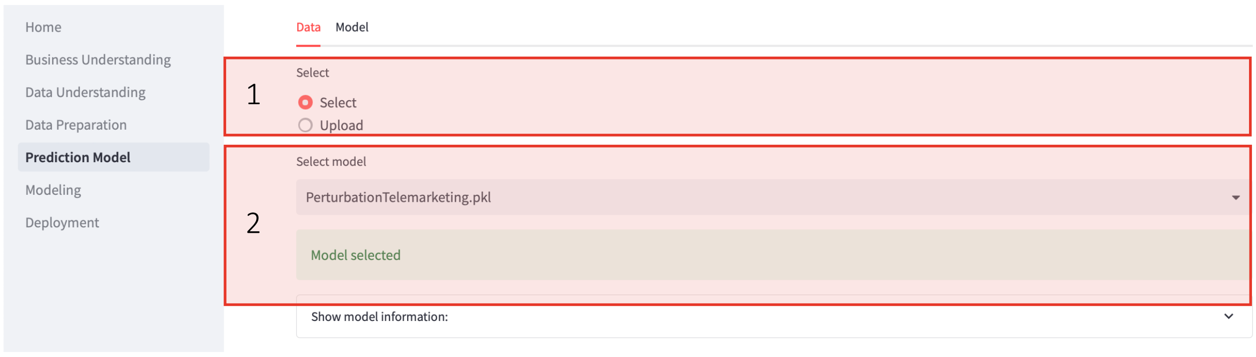
Figure 31 Data Preparation - determination of missing values

5.5. Prediction Model

This chapter describes the possibilities to load a classification prediction model into the web-based user interface. There are two different possibilities, illustrated in the highlighted area 1 in Figure 32. The web-based user interface offers the possibility to upload a classification model that was trained elsewhere and use it to predict perturbed cases. Once a model has been uploaded, it can be selected via a selection box, as can be seen in the highlighted area 2. A new folder with the name of the dataset is created in the current working directory for this purpose, in which all uploaded models for the dataset are saved.

The features of the prediction model must match the features set in the web-based user interface and be classified the same level of scale. The data runs through a Column Transform Pipeline, which transforms nominal values through a One Hot Encoder and ordinal columns through an Ordinal Encoder. A One Hot Encoder encodes categorical features as a numeric array, creating a binary column for each category. All the unique values of a nominal feature are converted into a separate column (scikit, 2023a). An ordinal encoder encodes an ordinal feature into ordinal integers, resulting in a single column of integers (0 to n_categories - 1) per feature (scikit, 2023b).

For testing purposes, the possibility to create a model was implemented in the web-based user interface, but the training data must be uploaded via the training data (II) upload described in chapter 5.1. This is necessary because to train a model the individual rows with the labelled target variables are required.



Select

[✓ Select] Upload

Select model

PerturbationTelemarketing.pkl

Model selected

Figure 32 Prediction Model - options for classification model

5.6. Modeling

This chapter shows the user interface for a developer, an analyst has no access to these pages and is referred to the *Deployment* page. This chapter describes how the Modeling step of the reference process from chapter 3.4 was implemented in the web-based user interface. Chapter 5.6.1 describes how the perturbation options for the individual features are selected. Chapter 5.6.2 addresses the configuration and upload of the perturbation options.

5.6.1. Choosing features

In the tab *Choose Perturbation Option* of the *Modeling* page, the perturbation options for the respective features are selected. The marked area in Figure 33 shows an example of the selection of perturbation options for the cardinal features.

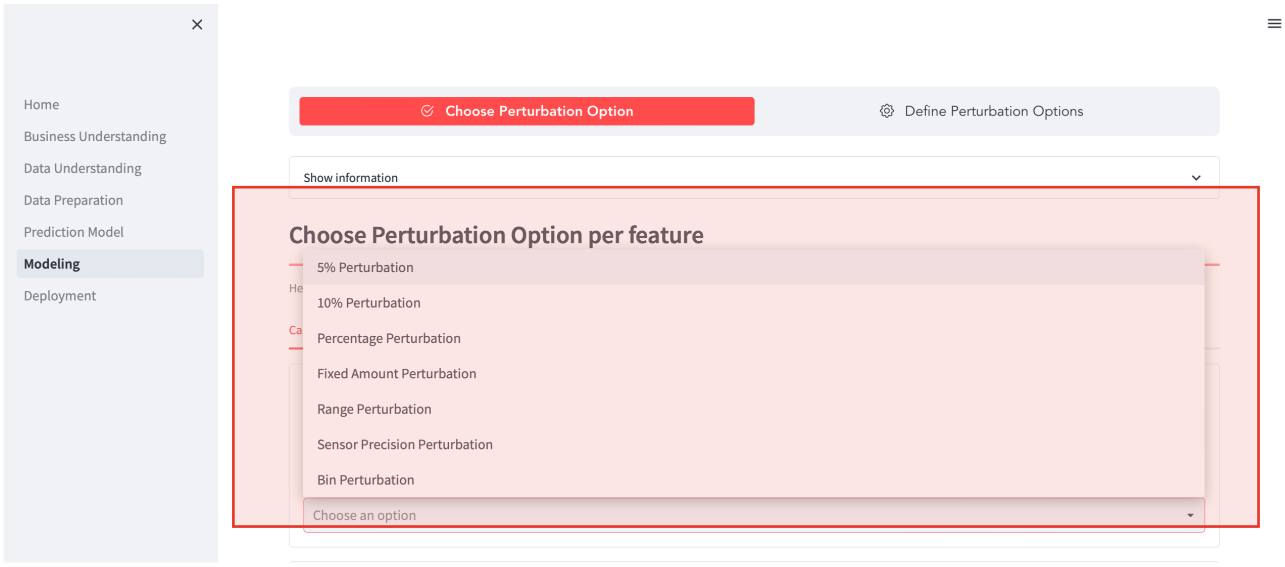
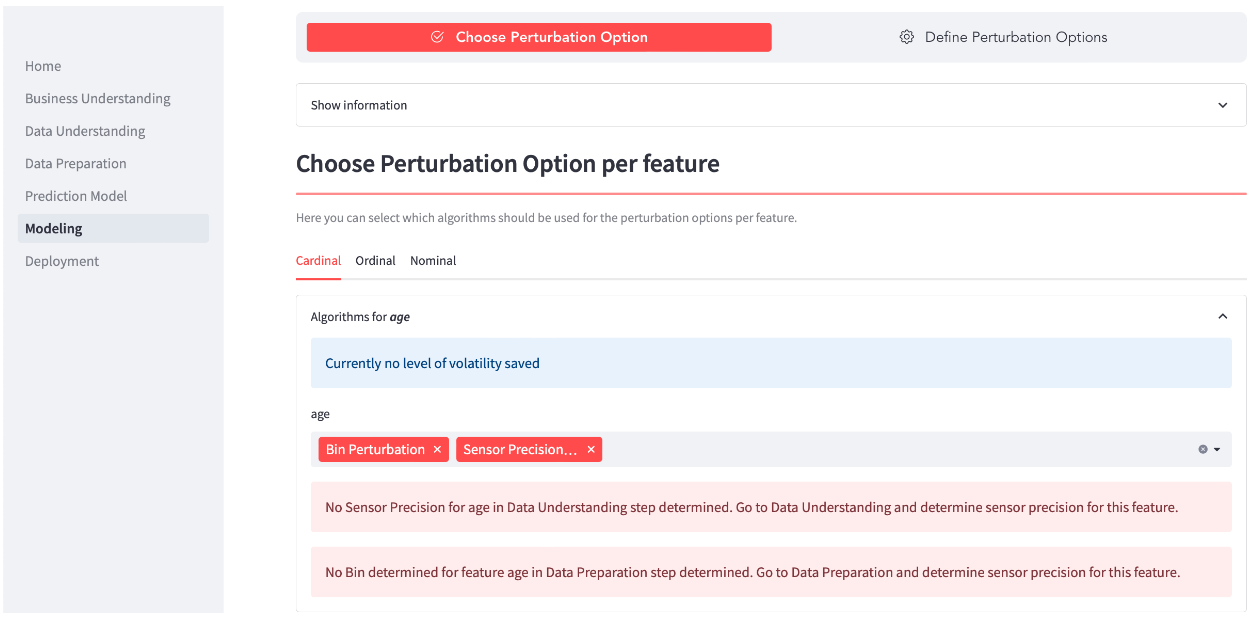


Figure 33 Modeling - choosing the perturbation option per feature

The *Percentage Perturbation* was created as a 5 % and 10 % variant. In addition, it is possible to create an individual *Percentage Perturbation* (Table 2). Furthermore, there is the option to select *Fixed Amount Perturbation* and *Range Perturbation*, which can be seen in Table 3 and Table 4 respectively.

Sensor Precision Perturbation (Table 5) and *Bin Perturbation* (Table 6) are available to the developer as a selection option but are only applied if this information has been entered during the Data Understanding or Data Preparation step. If not, the developer will see the error message in Figure 34. This design decision was made intentionally to indicate that this perturbation option can in principle be used for this feature as soon as the information is available.



Choose Perturbation Option per feature

Here you can select which algorithms should be used for the perturbation options per feature.

Algorithms for *age*

Currently no level of volatility saved

age

No Sensor Precision for age in Data Understanding step determined. Go to Data Understanding and determine sensor precision for this feature.

No Bin determined for feature age in Data Preparation step determined. Go to Data Preparation and determine sensor precision for this feature.

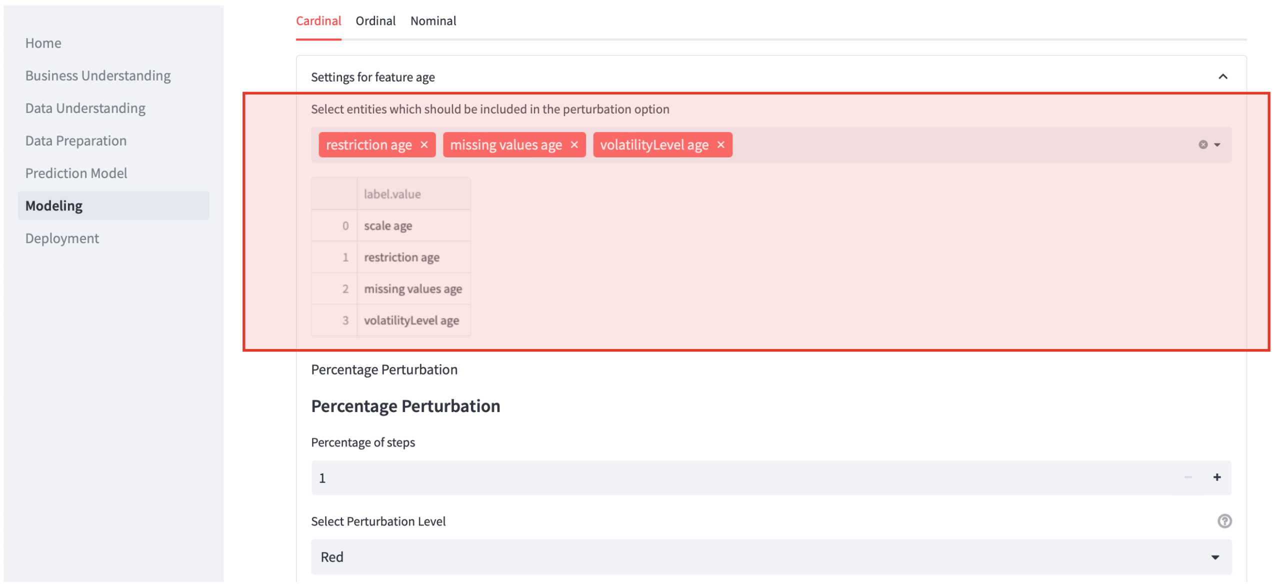
Figure 34 Modeling - error message Sensor Precision Perturbation and Bin Perturbation

For ordinal features, the perturbation options *Perturb In Order* (Table 7) and *Perturb All Values* (Table 8) can be selected. For nominal features, as already explained in chapter 3.5, only the option *Perturb All Values* is available. In principle, the number of perturbation options per feature is not limited and multiple selections are possible. Since the interface for the selection of ordinal and nominal features is structured identically to the selection of cardinal features, no further example illustrations are provided here.

5.6.2. Perturbation option

For the cardinal perturbation options, except for 5 % and 10 %, as well as for the ordinal perturbation option *Perturb In Order*, further parameters are required, which the developer can set in the next tab *Define Perturbation Options* on the *Modeling* page. The features are divided according to scale and only those features are displayed to the developer that were previously assigned to at least one perturbation option. If no perturbation option has been assigned to a certain level of scale, the developer is informed via an info box.

The volatility, data restriction and missing values information previously created during the Data Understanding and Data Preparation steps can optionally be added to the perturbation option. The scale, feature sensor precision and binning information are mandatory and cannot be deselected as these values have a direct influence on the selection of the perturbation options. This means that a developer can optionally provide the previously collected information in the Data Understanding and Data Preparation steps for the perturbation options, as exemplified in the highlighted area in Figure 35. By default, all information is preset and must be actively deselected by the developer.



Settings for feature age

Select entities which should be included in the perturbation option

restriction age × missing values age × volatilityLevel age ×

	label.value
0	scale age
1	restriction age
2	missing values age
3	volatilityLevel age

Percentage Perturbation

Percentage Perturbation

Percentage of steps

1

Select Perturbation Level

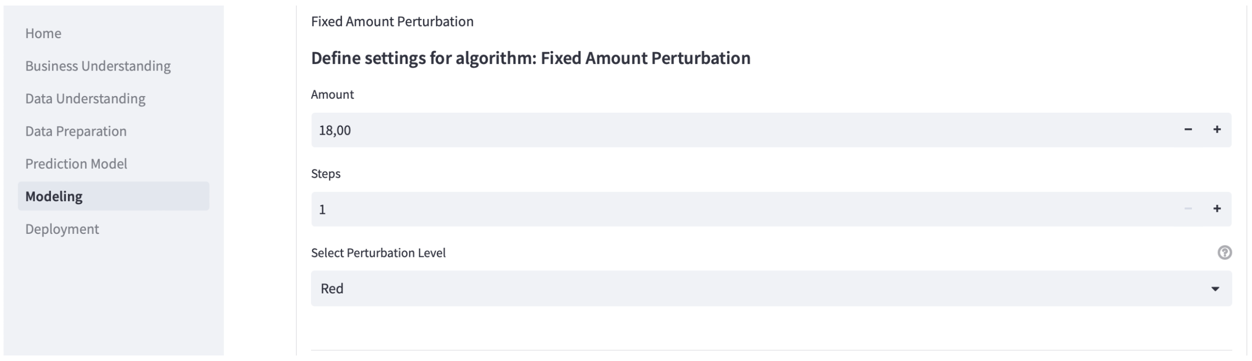
Red

Figure 35 Modeling - selection of entities for perturbation option

When the data restriction entity is selected, the range of values to be set changes according to the data restriction. This information is placed at the top of each settings expander for the perturbation options so that it cannot be overlooked. If several perturbation options are selected at the same time for a feature, they will receive the same Data Understanding and Data Preparation entities.

When *Percentage Perturbation* is selected for a cardinal feature, the developer has the option of specifying a percentage value between 1 and 100, as can be seen in the lower half of Figure 35. For each percentage value, a perturbed value is generated, with the values at the lower end generated first and then the perturbations above the original value. In this process, only values that are within the data restriction are generated. If a 20 % perturbation option with a value of 50 is selected, the perturbed values (40, 40.5, 41, ..., 48.5, 49, 49.5) are generated first, and then the values (50.5, 51, 51.5, ..., 59, 59.5, 60). In addition, the perturbation level can be selected for each perturbation option, which was explained in chapter 3.4. To give the developer full flexibility in creating the perturbation options, it is possible to choose between green, orange, and red perturbation levels for each perturbation option. It is not restricted to the recommended perturbation levels from chapter 3.4.

Fixed Amount Perturbation allows the developer to specify the value by which the initial value is to be changed as shown in Figure 36. The value can take a minimum value of 0.01 and a maximum value of the maximum unique values of this cardinal feature or if set, the data restriction. In addition, it can be set how many steps are to be perturbed. The perturbation option generates the number of defined steps of perturbed values in both directions or until the data restriction is reached. First, the fixed amount is subtracted from the output value until the restriction, or the number of steps is reached. Then the fixed amount is added to the output value until the data restriction, or the number of steps is reached. For example, if a feature value of 10 is changed by a fixed amount of 1 in 2 steps, the values (8, 9) and afterwards the values (11, 12) are created.



Fixed Amount Perturbation

Define settings for algorithm: Fixed Amount Perturbation

Amount

18,00

Steps

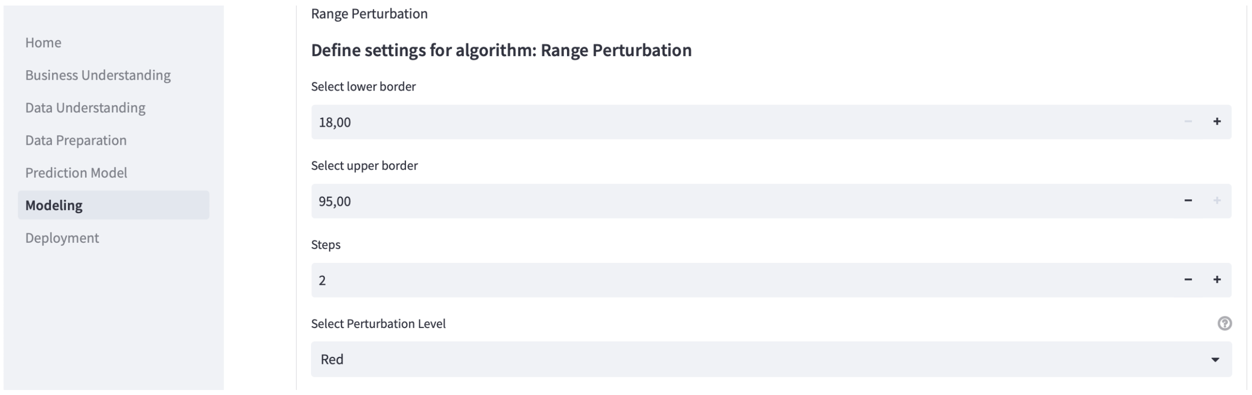
1

Select Perturbation Level

Red

Figure 36 Modeling - Fixed Amount Perturbation option

Figure 37 shows the interface for the *Range Perturbation* option*,* which allows the developer to enter the range using two input fields, where the lower limit must have at least the value of the minimum unique value or if set, the data restriction . The upper limit must not exceed the value of the maximum unique value or if set, the data restrictions of the feature. Additionally, the steps can be determined. In this example, the lower limit would be 18, the upper limit 95 and the number of steps 2, then the following values would be perturbed: (18, 56.5, 95).



Range Perturbation

Define settings for algorithm: Range Perturbation

Select lower border

18,00

Select upper border

95,00

Steps

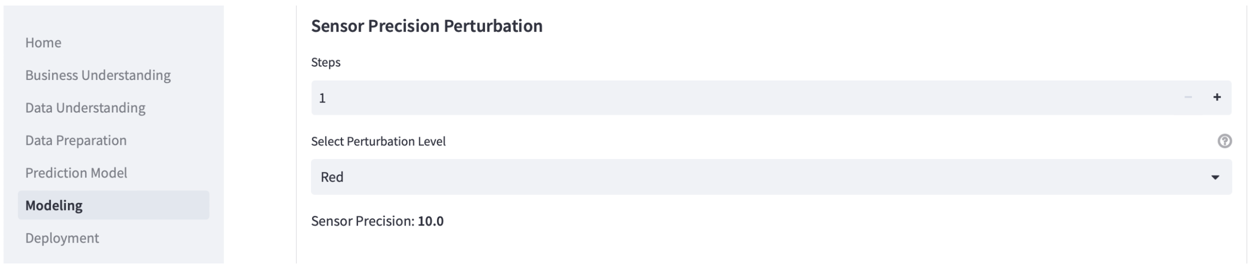
2

Select Perturbation Level

Red

Figure 37 Modeling - Range Perturbation option

Once a feature sensor precision has been defined for the feature in the Data Understanding step, the developer can create a *Sensor Precision Perturbation*, which can be seen in Figure 38. It can be specified how many steps are to be created within the sensor precision. In addition, it can be seen which sensor precision is currently set for the feature. In this example, the feature has a sensor precision of 10% and steps are set to 1. With a feature value of 50 and 1 step, the following new values are created (45, 55).



Sensor Precision Perturbation

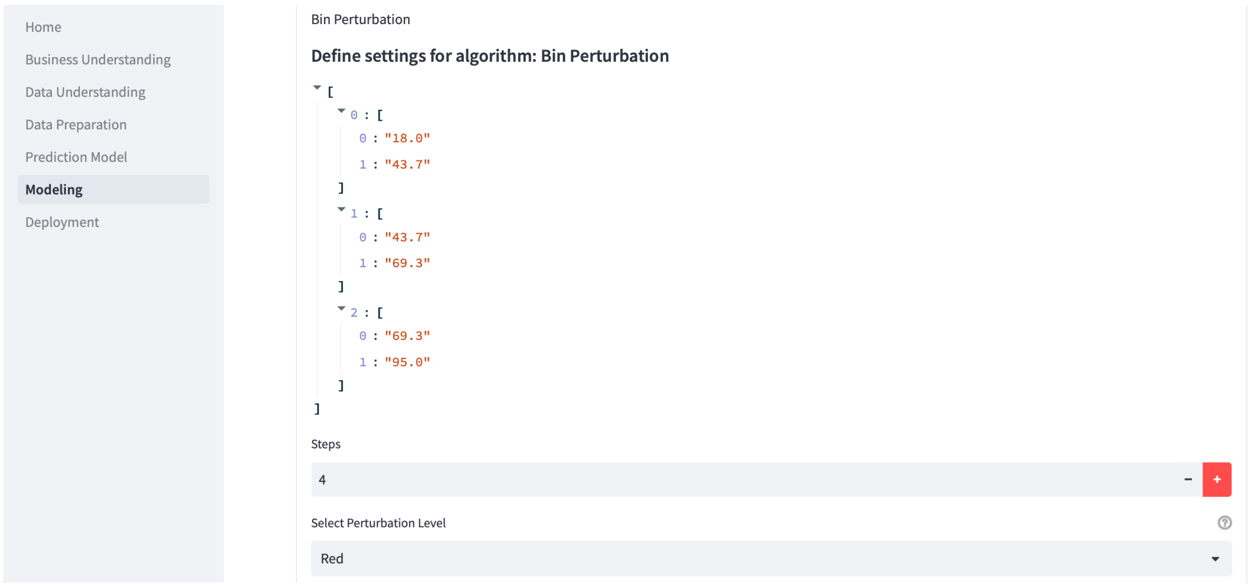
Steps: 1

Select Perturbation Level: Red

Sensor Precision: **10.0**

Figure 38 Modeling - Sensor Precision Perturbation option

If a binning of a cardinal feature was documented in the Data Preparation step, the *Bin Perturbation* can be determined for this feature. The developer is shown which binning values are saved on the top of the *Bin Perturbation* settings, as can be seen in Figure 39. In addition to this information, the developer can define the number of steps how many perturbed values are to be created within the bin. Assuming that the value to be perturbed is 20 and that 4 steps have been set, the following new values will be generated (18, 24.43, 37.28, 43.7).



Bin Perturbation

Define settings for algorithm: Bin Perturbation

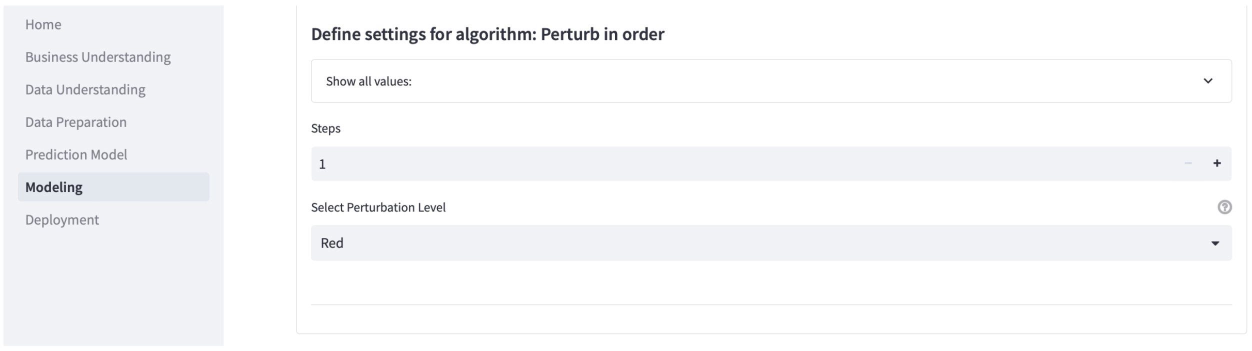
```
[
  0 : [
    0 : "18.0"
    1 : "43.7"
  ]
  1 : [
    0 : "43.7"
    1 : "69.3"
  ]
  2 : [
    0 : "69.3"
    1 : "95.0"
  ]
]
```

Steps: 4

Select Perturbation Level: Red

Figure 39 Modeling - Bin Perturbation option

For ordinal features, the developer can use the *Perturb In Order* perturbation option. Using an input field, the developer can set the number of steps by which the order of ordinal data is to be perturbed, as can be seen in Figure 40. The minimum amount of steps is one and the maximum amount of steps is the number of unique values or the number of data restriction values for the feature minus one. In the *Show all values* expander, the unique values or if set, the data restriction values are displayed.



Define settings for algorithm: Perturb in order

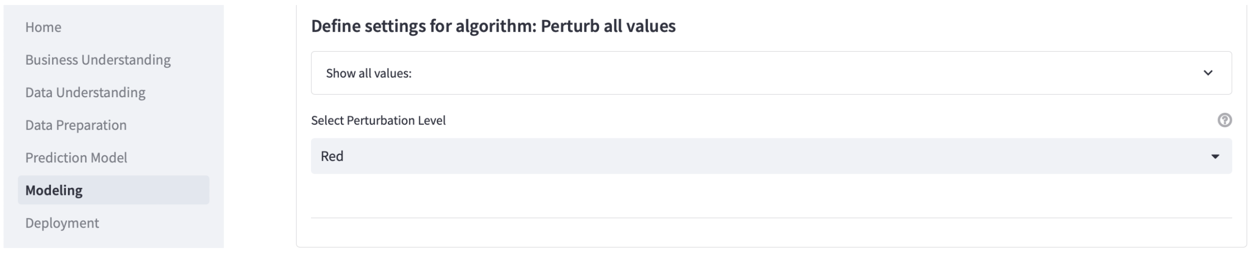
Show all values:

Steps: 1

Select Perturbation Level: Red

Figure 40 Modeling - Perturb In Order perturbation option

For *Perturb All Values* perturbation option, the developer can only set the perturbation level. As shown in Figure 41. The *Show all values* expander displays the unique values or if set, the data restriction values of the feature.



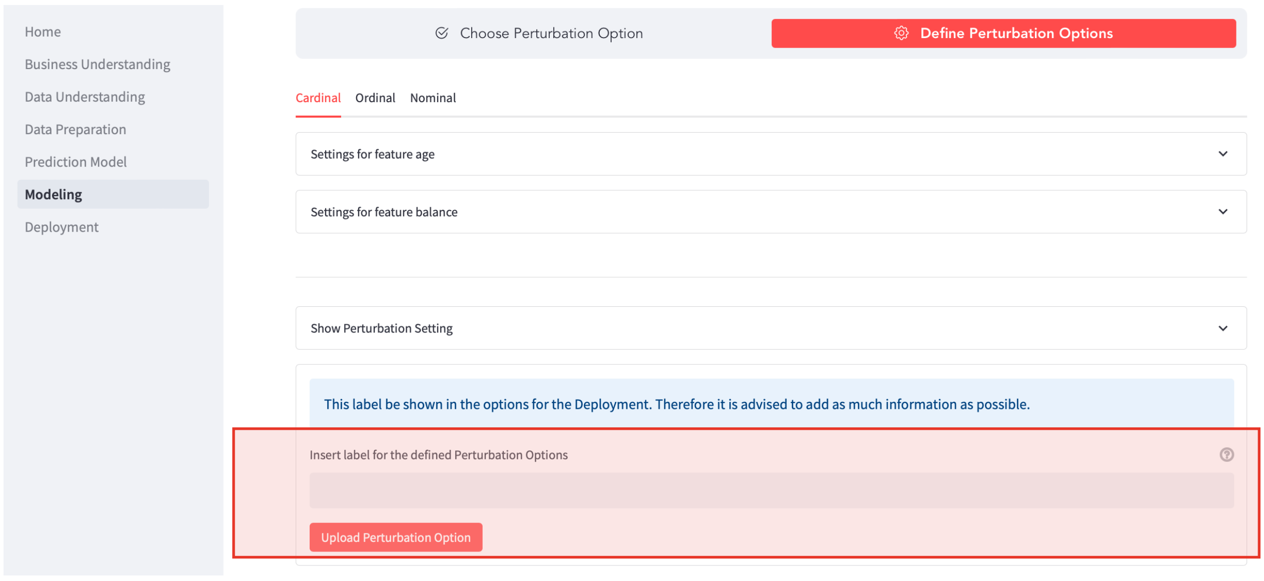
Define settings for algorithm: Perturb all values

Show all values:

Select Perturbation Level: Red

Figure 41 Modeling - Perturb All Values perturbation option

Since it is possible to generate perturbation options for different features and different algorithms at the same time, jointly created perturbation options can be added to a group and assigned a label, as shown in the highlighted are of Figure 42.


Home
Business Understanding
Data Understanding
Data Preparation
Prediction Model
Modeling
Deployment
Choose Perturbation Option
Define Perturbation Options
Cardinal
Ordinal
Nominal
Settings for feature age
Settings for feature balance
Show Perturbation Setting
Insert label for the defined Perturbation Options
Upload Perturbation Option

This label be shown in the options for the Deployment. Therefore it is advised to add as much information as possible.

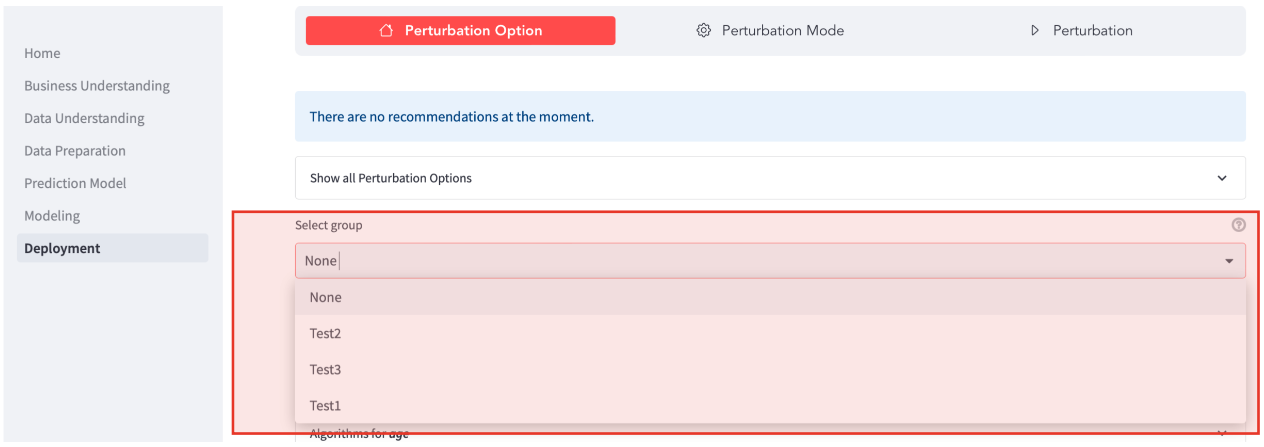
Figure 42 Modeling - labeling defined perturbation options

5.7. Deployment

This chapter describes how the Deployment step is implemented in the web-based user interface. Compared to the previous pages, the developer and the analyst have the same view of this page, which is divided into three tabs. The first tab contains the selection of the previously created perturbation options for the respective features, this tab is explained in more detail in chapter 5.7.1. Subsequently, the perturbation mode is presented in chapter 5.7.2. Chapter 5.7.3 shows the implementation of the perturbation. Developers and analysts will be referred to as users from now on.

5.7.1. Perturbation option

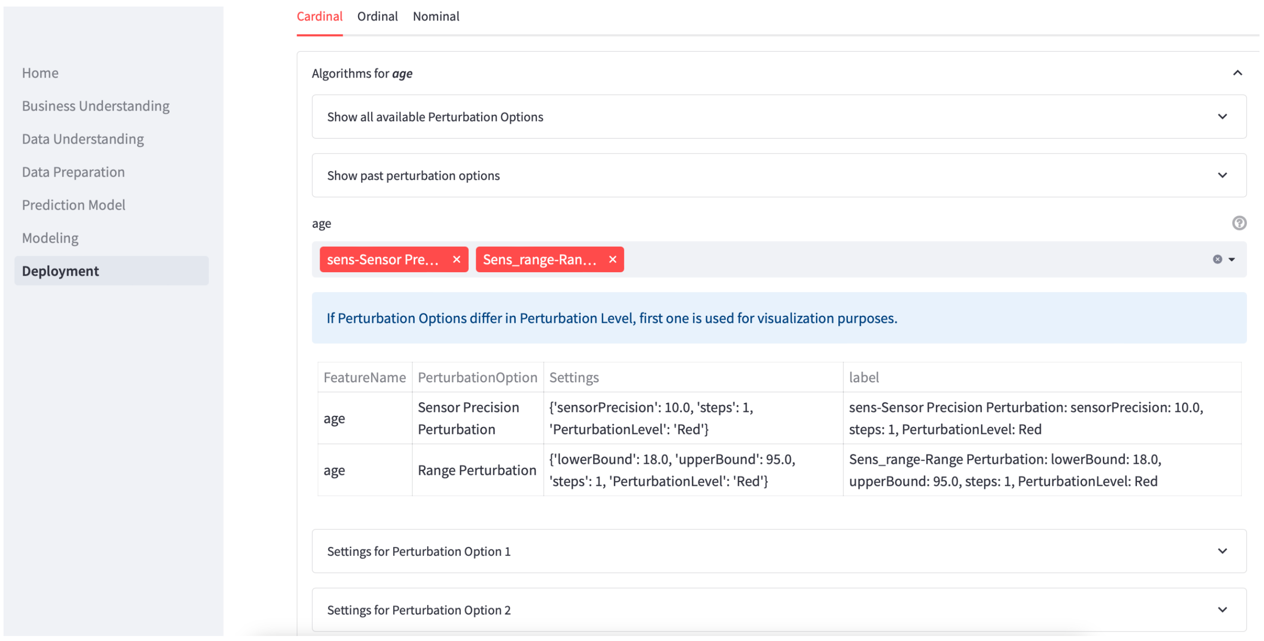
On the first tab of the *Deployment* page, all available perturbation options for all features are displayed in the first expander with the label *Show all Perturbation Options*. Two different approaches can be used to select the perturbation options. The first approach is to select a group of perturbation options created during a Modeling activity (highlighted area Figure 42), illustrated in the highlighted area in Figure 43.



There are no recommendations at the moment.

Figure 43 Deployment - choosing perturbation option per feature

The second approach is to select individual perturbation options. To select a perturbation option, the user must select a perturbation option in the multiselect field as shown in Figure 44. Once a perturbation option has been selected, the user can only select perturbation options with the same data restriction entity. The reason for this limitation is that the data restriction determines on the one hand which data can be entered as a classification case and on the other hand which values the perturbed values are allowed to take.



If Perturbation Options differ in Perturbation Level, first one is used for visualization purposes.

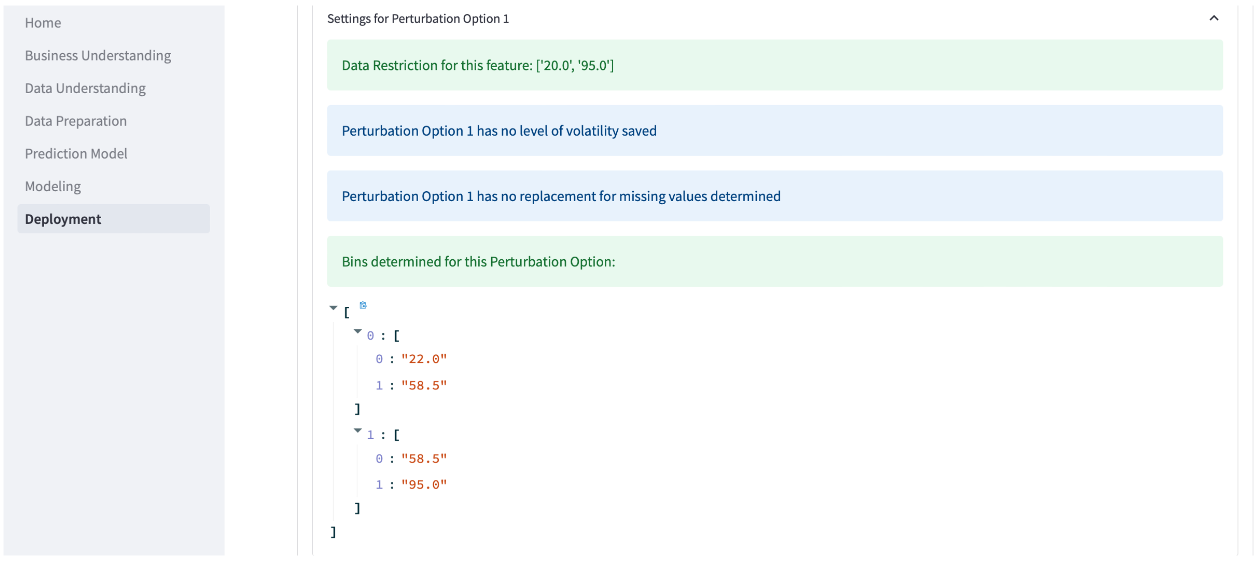
FeatureName	PerturbationOption	Settings	label
age	Sensor Precision Perturbation	{'sensorPrecision': 10.0, 'steps': 1, 'PerturbationLevel': 'Red'}	sens-Sensor Precision Perturbation: sensorPrecision: 10.0, steps: 1, PerturbationLevel: Red
age	Range Perturbation	{'lowerBound': 18.0, 'upperBound': 95.0, 'steps': 1, 'PerturbationLevel': 'Red'}	Sens_range-Range Perturbation: lowerBound: 18.0, upperBound: 95.0, steps: 1, PerturbationLevel: Red

Settings for Perturbation Option 1
Settings for Perturbation Option 2

Figure 44 Deployment - selection of perturbation for feature age

It is possible to select two identical perturbation options, but only the first selected is applied. This design decision was made because the newly created rows grow exponentially, and the same perturbation option produces the same values. However, the selection is not forbidden to inform the user and allow him to choose between the perturbation options.

If multiple perturbation options are selected, an expander is generated for each selected perturbation option, containing information about the perturbation option, as shown in Figure 45. In this expander, the user is informed which and whether a data restriction is valid for the perturbation options. It is displayed whether or which level of volatility has been selected for the perturbation option and whether information has been stored for how missing values were replaced. For Bin Perturbation, the determined bins are displayed.



Data Restriction for this feature: ['20.0', '95.0']

Perturbation Option 1 has no level of volatility saved

Perturbation Option 1 has no replacement for missing values determined

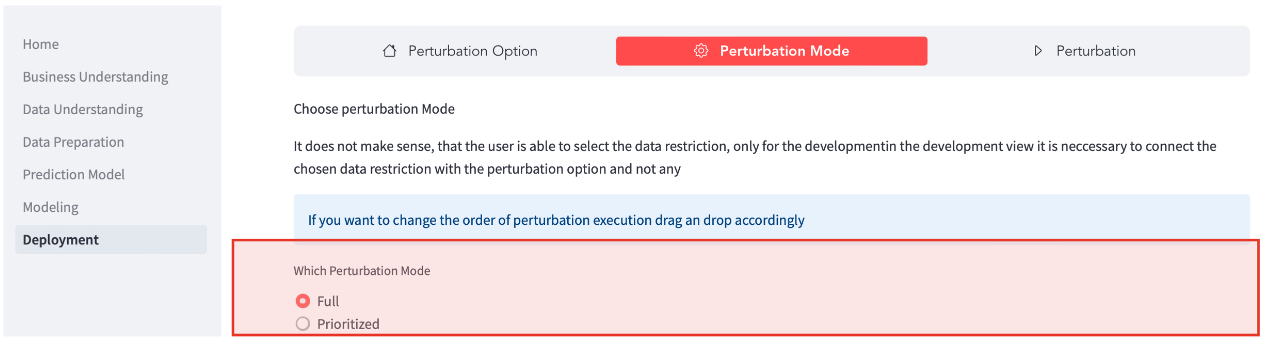
Bins determined for this Perturbation Option:

```
[
  0 : [
    0 : "22.0"
    1 : "58.5"
  ]
  1 : [
    0 : "58.5"
    1 : "95.0"
  ]
]
```

Figure 45 Deployment - information for selected perturbation option

5.7.2. Perturbation mode

In the *Perturbation Mode* tab, the user can set which perturbation mode is to be used, illustrated in the marked area in Figure 46. *Full* or *Prioritized Mode* can be selected. *Full Mode* is selected by default. If the user selects *Prioritized Mode*, he can do so via a list that determines the order of perturbations. The Selection Mode was not implemented since the selection of perturbation options is already done in the *Perturbation Option* Tab.



Choose perturbation Mode

It does not make sense, that the user is able to select the data restriction, only for the developmentin the development view it is neccessary to connect the chosen data restriction with the perturbation option and not any

If you want to change the order of perturbation execution drag an drop accordingly

Which Perturbation Mode

[✓ Full]
Prioritized

Figure 46 Deployment - perturbation

5.7.3. Perturbation

In the *Perturbation* tab, the user can perturb his desired use cases with the selected perturbation options. Figure 47 shows two expanders for the different upload methods for case data.

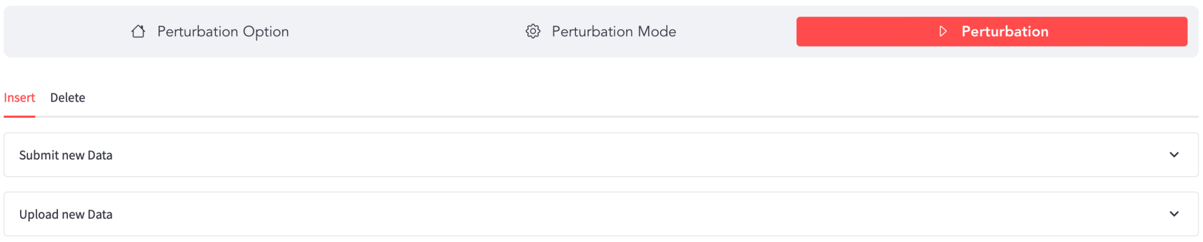
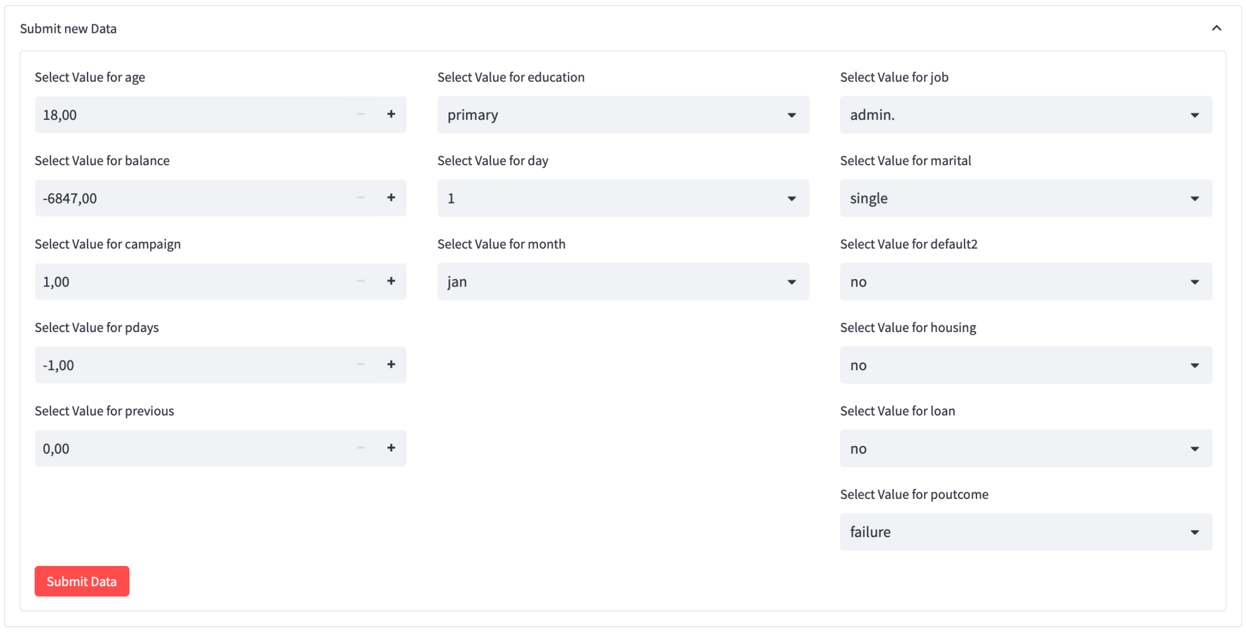


Figure 47 Deployment – uploading methods for case data

On the one hand, there is the option to *Submit new Data* via an input mask that accepts individual values, which can be seen in detail in Figure 48. The cardinal features are implemented as a number input field in the left column, in the middle column the user can set ordinal data via a select box and in the right column the nominal features can be set via a select box.

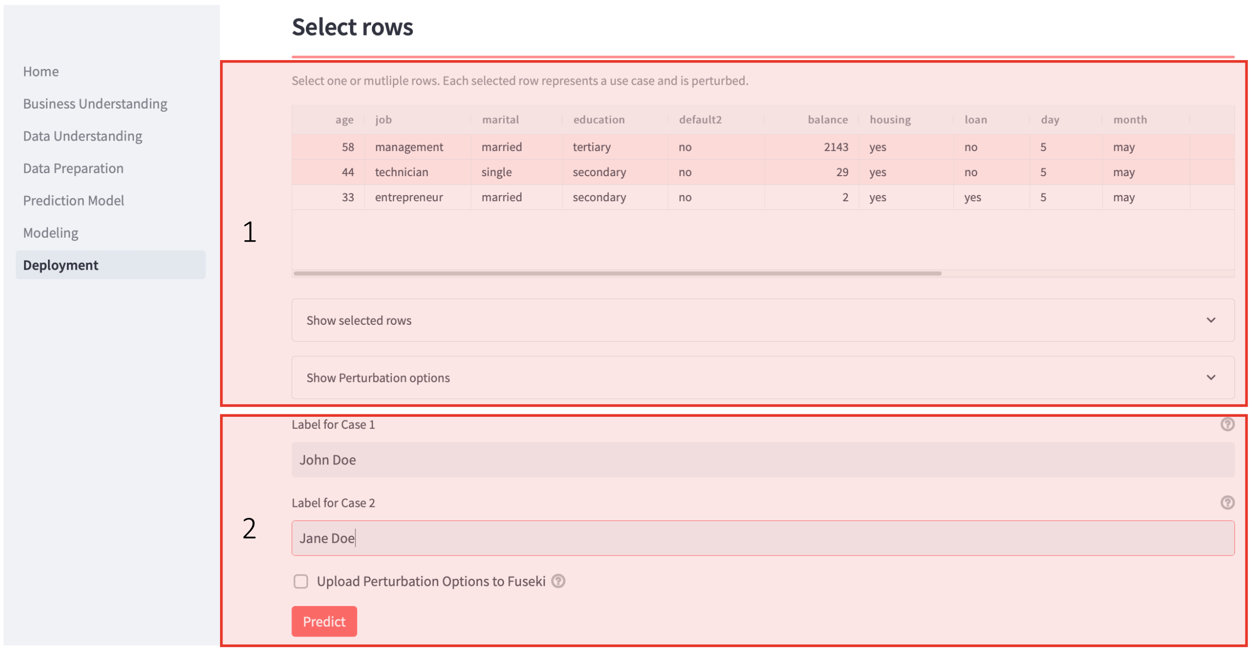


Submit new Data

Select Value for age	Select Value for education	Select Value for job
18,00	primary	admin.
Select Value for balance	Select Value for day	Select Value for marital
-6847,00	1	single
Select Value for campaign	Select Value for month	Select Value for default2
1,00	jan	no
Select Value for pdays		Select Value for housing
-1,00		no
Select Value for previous		Select Value for loan
0,00		no
		Select Value for poutcome
		failure

Figure 48 Deployment - individual interface for uploading case data

On the other hand, several use cases can be provided via a CSV upload via the *Upload new data* expander which can be seen in Figure 47. The columns must have the same label as the provided prediction model. For both approaches the user must select the perturbation use cases from a table visible in the marked area 1 in Figure 49. Figure 49 shows two additional expanders (highlighted area 1) that contain the selected rows on the one hand and the perturbation settings option on the other. To apply the selected perturbation options, a label must be created for each of the selected rows (highlighted area 2). As soon as this has been entered, the user can apply the perturbation options. The user can choose whether the assessment should be uploaded to the Apache Jena Fuseki server or not.



Select rows

Select one or mutliple rows. Each selected row represents a use case and is perturbed.

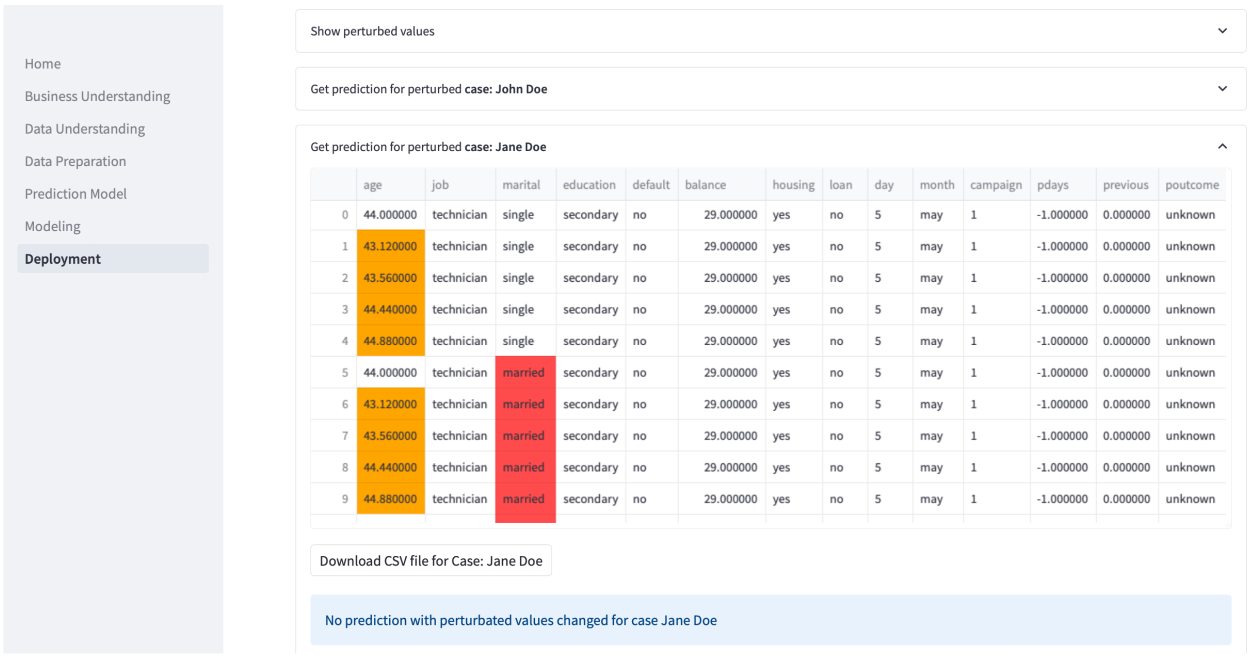
age	job	marital	education	default2	balance	housing	loan	day	month
58	management	married	tertiary	no	2143	yes	no	5	may
44	technician	single	secondary	no	29	yes	no	5	may
33	entrepreneur	married	secondary	no	2	yes	yes	5	may

Label for Case 1: John Doe

Label for Case 2: Jane Doe

☐ Upload Perturbation Options to Fuseki

Figure 49 Deployment - selection of rows for perturbation


Home
Business Understanding
Data Understanding
Data Preparation
Prediction Model
Modeling
Deployment
Show perturbed values
Get prediction for perturbed **case: John Doe**
Get prediction for perturbed **case: Jane Doe**
Download CSV file for Case: Jane Doe

	age	job	marital	education	default	balance	housing	loan	day	month	campaign	pdays	previous	poutcome
0	44.000000	technician	single	secondary	no	29.000000	yes	no	5	may	1	-1.000000	0.000000	unknown
1	43.120000	technician	single	secondary	no	29.000000	yes	no	5	may	1	-1.000000	0.000000	unknown
2	43.560000	technician	single	secondary	no	29.000000	yes	no	5	may	1	-1.000000	0.000000	unknown
3	44.440000	technician	single	secondary	no	29.000000	yes	no	5	may	1	-1.000000	0.000000	unknown
4	44.880000	technician	single	secondary	no	29.000000	yes	no	5	may	1	-1.000000	0.000000	unknown
5	44.000000	technician	married	secondary	no	29.000000	yes	no	5	may	1	-1.000000	0.000000	unknown
6	43.120000	technician	married	secondary	no	29.000000	yes	no	5	may	1	-1.000000	0.000000	unknown
7	43.560000	technician	married	secondary	no	29.000000	yes	no	5	may	1	-1.000000	0.000000	unknown
8	44.440000	technician	married	secondary	no	29.000000	yes	no	5	may	1	-1.000000	0.000000	unknown
9	44.880000	technician	married	secondary	no	29.000000	yes	no	5	may	1	-1.000000	0.000000	unknown

No prediction with perturbated values changed for case Jane Doe

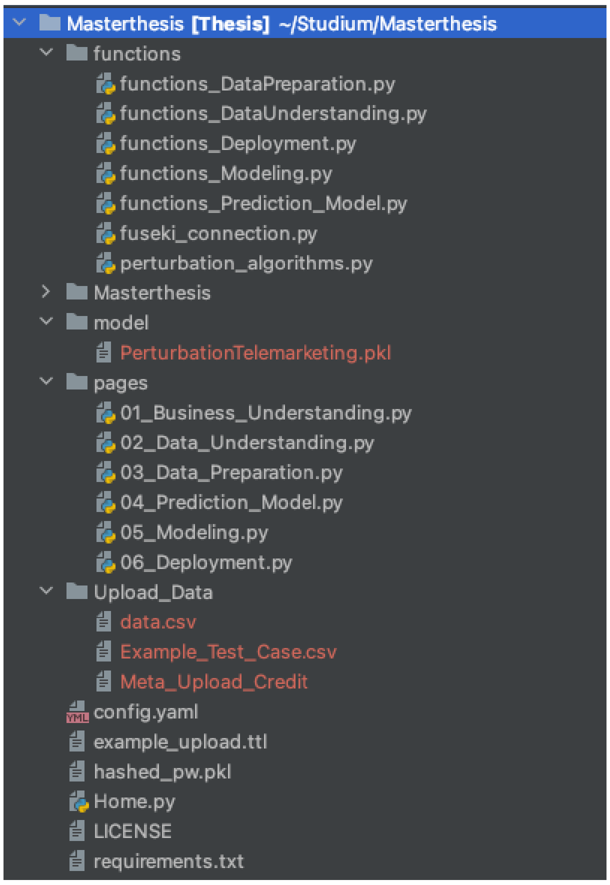
Figure 50 Deployment - perturbation cases

If the *Predict* (highlighted area 2) button in Figure 49 is pressed, the user interface looks like Figure 50. The first expander informs the user about the perturbed values of the individual features. The expanders of the selected use cases appear directly below. Within these expanders, a table with all newly created rows is displayed. The color of the columns informs the user about the perturbation level. This table can be downloaded via the *Download CSV file for Case: Jane Doe* button provided. If at least one prediction in a perturbed test case changes compared to the original use case, another expander appears containing the perturbed cases that lead to a change in the individual prediction.

6. Implementation perspective

This chapter deals with aspects of the implementation of the web-based user interface. Chapter 6.1 covers the general implementation structure of the web-based user interface. Chapter 6.2 describes the knowledge graph in detail.

6.1. General implementation structure



```
Masterthesis [Thesis] ~/Studium/Masterthesis
  functions
    functions_DataPreparation.py
    functions_DataUnderstanding.py
    functions_Deployment.py
    functions_Modeling.py
    functions_Prediction_Model.py
    fuseki_connection.py
    perturbation_algorithms.py
  Masterthesis
  model
    PerturbationTelemarketing.pkl
  pages
    01_Business_Understanding.py
    02_Data_Understanding.py
    03_Data_Preparation.py
    04_Prediction_Model.py
    05_Modeling.py
    06_Deployment.py
  Upload_Data
    data.csv
    Example_Test_Case.csv
    Meta_Upload_Credit
  config.yaml
  example_upload.ttl
  hashed_pw.pkl
  Home.py
  LICENSE
  requirements.txt
```

Figure 51 Master thesis implementation structure

Figure 51 shows the structure of the implementation of the master thesis. The start page as well as the start script of the web-based user interface is the file Home.py, which is visualized and described in chapter 5.1. This script must be started to start the web-based user interface via the command streamlit run Home.py.

The other pages are in the pages folder and work independently of each other. In addition, a separate folder was created for the helper functions of the respective pages, which support the general functionality of the respective page and are in the functions folder. In addition, there is a separate Python script called fuseki_connection.py. In this file, all functions that access the graph database have been placed. One last script contains the implementation of the underlying algorithms of the perturbation options namely perturbation_algorithms.py.

In the model folder, the predictive models of the use cases are stored in a separate folder named after the selected dataset. In the config.yaml file, the registered users with the hashed passwords are stored, as well as the e-mail addresses that are allowed to log in, furthermore, the expiry date of the cookies can be set here. The folder Upload_Data, contains sample data for the telemarketing use case namely the training data, the metadata, and an example of use case data. The last file Example_upload.ttl is used to ensure that when a new Fuseki dataset is created via the web-based user interface's interface, all required RDF triples are loaded into the newly created dataset.

6.2. Knowledge graph

The knowledge graph is stored using the RDF format, described in chapter 2.3, in the Fuseki triple store. In the following, the RDF triples for the individual steps of the reference process are explained and provided with examples. To keep the explanations understandable, the order of the reference process is followed, although the RDF triples in the web-based user interface can be created in a different order. First, however, the activities and entities of the individual reference process steps are explained since they are structured identically. Second, the other required RDF triples of Business Understanding, Data Understanding, Data Preparation, Modeling, and Deployment are explained.

Table 11 summarizes the activities of the CRISP-DM steps in RDF format. The namespace rprov used in the reference process was adopted. This prefix is used in the following for all provenance information required for the reference process that is not already contained in the PROV-O.

Table 11 RDF activities of the reference process steps

Subject	Predicate	Object
rprov: BusinessUnderstandingActivity DataUnderstandingActivity DataPreparationActivity ModelingActivity EvaluationActivity DeploymentActivity	rdf:type	owl:Class
	rdfs:subClassOf	prov:Activity

The activities each represent an instance of an owl:Class and the respective activities are each an instance of the *prov:Activity* via *rdfs:subClassOf*. Table 12 looks similar for the entities of the respective steps. Here, too, the entities are an instance of an *owl:Class* and an instance of the *prov:Entity* via *rdfs:subClassOf*. The *rdf:type owl:class* is no longer mentioned for further examples. Rdfs is an abbreviation for RDF Schema and is an extension of the basic RDF vocabulary.

Table 12 RDF entities of the reference process steps

Subject	Predicate	Object
rprov: BusinessUnderstandingEntity DataUnderstandingEntity DataPreparationEntity ModelingEntity DeploymentEntity	rdf:type	owl:Class
	rdfs:subClassOf	prov:Entity

To link the entities with the activities of the reference process, the individual object properties *rprov:wasGeneratedBy* defined in Table 13 are used. This way, each activity can be assigned to an entity and vice versa. The entity instance is specified via *rdfs:domain* and *rdfs:range* determines the instance of the activity. These object Properties are a *rdfs:subPropertyOf* the *prov:wasGeneratedBy.*

Table 13 RDF connection between activities and entities of the reference process steps

Subject	Predicate	Object
rprov:wasGeneratedBy BusinessUnderstandingActivity DataUnderstandingActivity DataPreparationActivity ModelingActivity DeploymentActivity	rdf:type	owl:ObjectProperty
	rdfs:domain	rprov:Entity
	rdfs:range	rprov:Activity
	rdfs:subPropertyOf	prov:wasGeneratedBy

6.2.1. Business Understanding

This chapter shows how the information of the Business Understanding step is stored. For this purpose, Table 14 shows *rprov:ChoiceOfAssessmentApproach* which is a sub class of the *rprov:BusinessUnderstandingActivity* and is required once the user has selected an assessment approach. The predicate *prov:endedAtTime* documents when the activity ended and connects it to a *xsd:dateTime*. Additionally, the *rdfs:label* is stored in *xsd:string*. Xsd is an abbreviation for XML Schema Definition.

Table 14 RDF Business Understanding ChoiceOfAssessmentApproach activity

Subject	Predicate	Object
rprov:ChoiceOfAssessment Approach	rdf:type	owl:Class
	rdfs:subClassOf	rprov:BusinessUnderstanding Activity
	rdfs:label	xsd:string
	prov:endedAtTime	xsd:dateTime

Table 15 shows the class *rprov:AssessmentApproach* created for this purpose, which is a *rdfs:subClassOf rprov:BusinessUnderstandingEntity*. This class is the superclass of *rprov:perturbationApproach*, which is relevant for this master thesis and is summarized in Table 16. The predicate *rprov:wasGeneratedByBUA* connects *rprov:perturbationApproach* with *rprov:ChoiceOfAssessmentApproach*. The time of generation is stored via *prov:generatedAtTime* in *xsd:dateTime*.

Table 15 RDF Business Understanding AssessmentApproach entity

Subject	Predicate	Object
rprov:AssessmentApproach	rdf:type	owl:Class
	rdfs:subClassOf	rprov:BusinessUnderstandingEntity

Table 16 RDF Business Understanding perturbationApproach entity

Subject	Predicate	Object
rprov:perturbationApproach	rdf:type	owl:Class
	rdfs:subClassOf	rprov:AssessmentApproach
	rprov:wasGeneratedByBUA	rprov:ChoiceOfAssessmentApproach
	prov:generatedAtTime	xsd:dateTime

Code 2 shows an example of the Business Understanding activity and the Business Understanding entity. To improve the readability of this example, a placeholder has been used for the unique Universal Resource Identifiers described in chapter 2.3. The @en after the label stands for the language of the label, multiple languages can be defined directly to allow to change the language easily. The label is provided to all the instances to make it human readable. As can be seen the Business Understanding entity instance *<URI-perturbationApproach>* was generated by <URI-ChoiceOfAssessmentApproach>. The predicate *prov:generatedAtTime* indicates when the entity was created, this predicate is found in all subsequent entities and will not be mentioned further. The same applies to the generated labels of the activities and the entities.

Code 2 RDF Business Understanding assessment approach example

```
<URI-ChoiceOfAssessmentApproach>
  rdf:type                    rprov:ChoiceOfAssessmentApproach , owl:NamedIndividual ;
  rdfs:label                  "choiceOfAssessment"@en ;
  prov:endedAtTime            xsd:dateTime .

<URI-perturbationApproach>
  rdf:type                    rprov:perturbationApproach , owl:NamedIndividual ;
  rprov:wasGeneratedByBUA     <URI-ChoiceOfAssessmentApproach> ;
  prov:generatedAtTime        xsd:dateTime .
```

6.2.2. Data Understanding

Table 17 presents the RDF triples for the Data Understanding activities together since they have the same structure and only change in subject. The following activities are distinguished: *Determination-*

OfFeature, *DeterminationOfScaleOfFeature*, *DeterminationOfUniqueValuesOfFeature*, *DeterminationOfVolatilityOfFeature*, *DeterminationOfDataRestriction* and *DeterminationOfSensorPrecisionOfFeature*. Each of these activities creates a Data Understanding entity, which is shown and explained below.

Table 17 RDF Data Understanding activities

Subject	Predicate	Object
rprov:DeterminationOf **Feature** **ScaleOfFeature** **UniqueValuesOfFeature** **VolatilityOfFeature** **DataRestriction** **SensorPrecisionOfFeature**	rdf:type	owl:Class
	rdfs:subClassOf	rprov:DataUnderstandingActivity
	rdfs:label	xsd:string
	prov:endedAtTime	xsd:dateTime

Feature

Table 18 shows the RDF triples for the feature entity, which is an instance of the Data Understanding entity and is linked to and created by the *rprov:DeterminationOfFeature* activity via *rprov:wasGeneratedByDUA*.

Table 18 RDF Data Understanding Feature entity

Subject	Predicate	Object
rprov:Feature	rdf:type	owl:Class
	rdfs:subClassOf	rprov:DataUnderstandingEntity , prov:Entity
	rdfs:label	xsd:string
	rprov:wasGeneratedByDUA	rprov:DeterminationOfFeature

The object property *rprov:toFeature* shown in Table 19 is used for the instances of the Data Understanding, Data Preparation and Modeling entities to link them to the features which is determined by *rdfs:domain* and *rdfs:range*.

Table 19 RDF Data Understanding toFeature object property

Subject	Predicate	Object
rprov:toFeature	rdf:type	owl:ObjectProperty
	rdfs:domain	rprov:HandlingOfMissingValues , rprov:DataRestriction , rprov:VolatilityOfFeature , rprov:RangeOfBinnedFeature , rprov:ScaleOfFeature , rprov:SensorPrecisionOfFeature
	rdfs:range	rprov:Feature

Code 3 shows the creation of a *rprov:DeterminationOfFeature* and a feature instance. *<URI-DeterminationOfFeature>* is an instance of the Data Understanding activity that creates the *<URI-Feature>* Data Understanding entity with the feature name as label.

Code 3 Data Understanding feature example

```
<URI-DeterminationOfFeature>
  rdf:type                      rprov:DeterminationOfFeature , owl:NamedIndividual ;
  rdfs:label                    "detOfFeature"@en ;
  prov:endedAtTime              xsd:dateTime .

<URI-Feature>
  rdf:type                      rprov:Feature , owl:NamedIndividual ;
  rdfs:label                    "FeatureName"@en ;
  rprov:wasGeneratedByDUA       <URI-DeterminationOfFeature> .
```

Scale

Table 20 shows how the level of scales for the features are stored in RDF syntax. The scale entity is instantiated by the level of scales classes *rprov:Cardinal*, *rprov:Nominal* and *rprov:Ordinal*, shown in Table 21.

Table 20 RDF Data Understanding Scale entity

Subject	Predicate	Object
rprov:Scale	rdf:type	owl:Class
	rdfs:subClassOf	prov:Entity

Table 21 RDF Data Understanding level of scale entity

Subject	Predicate	Object
rprov: **Cardinal** **Nominal** **Ordinal**	rdf:type	owl:Class
	rdfs:subClassOf	rprov:Scale

The *rprov:scale* shown in Table 22 indicates an object property that is required to pass the scale property to a feature. The instance of the Data Understanding entity *rprov:ScaleOfFeature* is assigned *rprov:Cardinal*, *rprov:Ordinal*, or *rprov:Nominal* via the predicate *rprov:scale*, which is an instance of *rprov:Scale*.

Table 22 RDF Data Understanding scale object property

Subject	Predicate	Object
rprov:scale	rdf:type	owl:ObjectProperty
	rdfs:domain	rprov:ScaleOfFeature
	rdfs:range	rprov:Scale
	rdfs:subPropertyOf	prov:wasDerivedFrom

Table 23 shows the Data Understanding entity *rprov:ScaleOfFeature*, which is created by *rprov:DeterminationOfScaleOfFeature* and assigned via *rprov:toFeature* to a *rprov:Feature*, which represents the instantiated Data Understanding entities *<URI-Feature>* in example Code 4. Furthermore, the object property *rprov:scale* contains the information to the respective scale level.

Table 23 RDF Data Understanding ScaleOfFeature entity

Subject	Predicate	Object
rprov:ScaleOfFeature	rdf:type	owl:Class
	rdfs:subClassOf	rprov:DataUnderstandingEntity
	rprov:wasGeneratedByDUA	rprov:DeterminationOfScaleOf Feature
	rdfs:label	xsd:string
	rprov:toFeature	rprov:Feature

Code 4 Data Understanding scale example

```
<URI-DeterminationOfScaleOfFeature>
  rdf:type                  rprov:DeterminationOfScaleOfFeature, owl:NamedIndividual ;
  rdfs:label                "detScaleOfFeature"@en ;
  prov:endedAtTime          xsd:dateTime .

<URI-ScaleOfFeature>
  rdf:type                  rprov:ScaleOfFeature , owl:NamedIndividual ;
  rdfs:label                "scaleFeatureName"@en ;
  rprov:scale               rprov:Cardinal ;
  rprov:toFeature           <URI-Feature> ;
  rprov:wasGeneratedByDUA   <URI-DeterminationOfScaleOfFeature> ;
  prov:generatedAtTime      ^^xsd:dateTime .

<URI-ScaleOfFeature>
  rdf:type                  rprov:ScaleOfFeature , owl:NamedIndividual ;
  rdfs:label                "scaleFeatureName"@en ;
  rprov:scale               rprov:Ordinal ;
  rprov:toFeature           <URI-Feature> ;
  rprov:wasGeneratedByDUA   <URI-DeterminationOfScaleOfFeature> ;
  prov:generatedAtTime      ^^xsd:dateTime .

<URI-ScaleOfFeature>
  rdf:type                  rprov:ScaleOfFeature , owl:NamedIndividual ;
  rdfs:label                "scaleFeatureName"@en ;
  rprov:scale               rprov:Nominal ;
  rprov:toFeature           <URI-Feature> ;
  rprov:wasGeneratedByDUA   <URI-DeterminationOfScaleOfFeature> ;
  prov:generatedAtTime      ^^xsd:dateTime .
```

Unique values

In the following, the storage of the unique values of the individual features in RDF syntax is explained. Table 24 shows the Data Understanding entity *rprov:UniqueValuesOfFeature*, which contains the information about the unique values of a feature. This is illustrated by the datatype property *rprov:uniqueValues*, which is summarized in Table 25 and shown in Code 5.

All *rprov:UniqueValuesOfFeature* entities are created by the same Data Understanding activity instance *<URI-DeterminationOfUniqueValuesOfFeature>*. The storage of cardinal and ordinal features differs from the storage of nominal features. The class *rdf:Seq* is used for the storage of cardinal and ordinal unique values. This does not differ formally from *rdf:Bag*, which is used for the nominal unique values. However, it is usually used to indicate to the human reader that the numerical order of the container's membership properties should be significant (W3C, 2014a). The assignment of the entity to the feature is done via the predicate *rprov:toFeature*. The data type property *rprov:uniqueValues* associates the entity with the content of the class sequence stored as *rdf:Seq* or *rdf:Bag*.

As can be seen in the example of Code 5, the unique values for the cardinal and ordinal features are stored as *rdf:Seq* and are thus assigned a sequence. For the cardinal features, these are always *rdf:_0 minimumValue* and *rdf:_1 maximimumValue*. To know which is the minimum and which is the maximum value, these must be ordered. For ordinal features all unique values are saved in order. For the nominal features the class *rdf:Bag* is used and therefore contains no order.

Table 24 RDF Data Understanding UniqueValuesOfFeature entity

Subject	Predicate	Object
rprov:UniqueValuesOfFeature	rdf:type	owl:Class
	rdfs:subClassOf	rprov:DataUnderstandingEntity
	rprov:wasGeneratedByDUA	rprov:DeterminationOfUniqueValuesOfFeature

Table 25 RDF Data Understanding uniqueValues data property

Subject	Predicate	Object
rprov:uniqueValues	rdf:type	owl:DatatypeProperty
	rdfs:domain	rprov:UniqueValuesOfFeature
	rdfs:range	rdf:Seq, rdf:Bag

Code 5 Data Understanding unique values example

```
<URI-DeterminationOfUniqueValuesOfFeature>
  rdf:type                    rprov:DeterminationOfUniqueValuesOfFeature, owl:NamedIndividual;
   rdfs:label                 "detUniqueValuesOfFeature"@en ;
   prov:endedAtTime           xsd:dateTime .
```

```
<URI-UniqueValuesOfCardinalFeature>
  rdf:type                      rprov:UniqueValuesOfFeature , owl:NamedIndividual ;
  rdfs:label                    "uniqueValues age"@eng ;
  rprov:toFeature               <URI-Feature> ;
  rprov:uniqueValues            <URI-UniqueValuesCardinalFeature> ;
  rprov:wasGeneratedByDUA       <URI-DeterminationOfUniqueValuesOfFeature> ;
  prov:generatedAtTime          ^^xsd:dateTime .

<URI-UniqueValuesCardinalFeature>
  rdf:type                      rdf:Seq , owl:NamedIndividual ;
  rdf:_0                        "18" ;
  rdf:_1                        "95" .

<URI-UniqueValuesOfOrdinalFeature>
  rdf:type                      rprov:UniqueValuesOfFeature , owl:NamedIndividual ;
  rdfs:label                    "uniqueValues education"@eng ;
  rprov:toFeature               <URI-Feature> ;
  rprov:uniqueValues            <URI-UniqueValuesOrdinalFeature> ;
  rprov:wasGeneratedByDUA       <URI-DeterminationOfUniqueValuesOfFeature> ;
  prov:generatedAtTime          ^^xsd:dateTime .

<URI-UniqueValuesOrdinalFeature>
  rdf:type                      rdf:Seq , owl:NamedIndividual ;
  rdf:_0                        "primary." ;
  rdf:_1                        "secondary" ;
  rdf:_2                        "tertiary" .

<URI-UniqueValuesOfNominalFeature>
  rdf:type                      rprov:UniqueValuesOfFeature , owl:NamedIndividual ;
  rdfs:label                    "uniqueValues job"@eng ;
  rprov:toFeature               <URI-Feature> ;
  rprov:uniqueValues            <URI-UniqueValuesNominalFeature> ;
  rprov:wasGeneratedByDUA       <URI-DeterminationOfUniqueValuesOfFeature> ;
  prov:generatedAtTime          ^^xsd:dateTime .

<URI-UniqueValuesNominalFeature>
  rdf:type                      rdf:Bag , owl:NamedIndividual ;
  rdf:_0                        "admin." ;
  rdf:_1                        "blue-collar" ;
  rdf:_2                        "technician" ;
  …                             …
```

Volatility

Table 26 contains the RDF triples generated by the activity *rprov:DeterminationOfVolatilityOfFeature*, which are assigned to the defined volatility levels via the datatype property *rprov:volatilitylevel* contained in Table 27.

Table 26 RDF Data Understanding VolatilityOfFeature entity

Subject	Predicate	Object
rprov:VolatilityOfFeature	rdf:type	owl:Class
	rdfs:subClassOf	rprov:DataUnderstandingEntity
	rprov:wasGeneratedByDUA	rprov:DeterminationOfVolatili-tyOfFeature

Table 27 RDF Data Understanding volatilityLevel datatype property

Subject	Predicate	Object
rprov:volatilitylevel	rdf:type	owl:DatatypeProperty
	rdfs:domain	rprov:VolatilityOfFeature
	rdfs:range	xsd:string

Code 6 illustrates an example for a *rprov:DeterminationOfVolatilityOfFeature* instance *<URI-DeterminationOfVolatilityOfFeature>* that creates a *rprov:VolatilityOfFeature* instance *<URI-VolatilityOfFeature>*. This contains the information of the *rprov:volatilitylevel* in form of *xsd:string* and is assigned to a feature via *rprov:toFeature*.

Furthermore, the *<URI-VolatilityOfFeature>* instance contains information about the *prov:invalidatedAtTime*, which is created as soon as a user deletes any entity via the frontend. This predicate is then assigned to the instance and remains in the RDF triple-store and thus in the perturbation options already created. For new perturbation options it is no longer available, and a new entity can be created. This ensures that the generated information is not lost when creating new entities. Since this procedure is identical in the following steps of the Data Understanding as well as the Data Preparation step, it will not be mentioned in the following examples.

Code 6 Data Understanding volatility example

```
<URI-DeterminationOfVolatilityOfFeature>
  rdf:type                       rprov:DeterminationOfVolatilityOfFeature,
                                 owl:NamedIndividual ;
  rdfs:label                     "detVolatilityOfFeature"@en ;
  prov:endedAtTime               xsd:dateTime .

<URI-VolatilityOfFeature>
  rdf:type                       rprov:VolatilityOfFeature, owl:NamedIndividual ;
  rdfs:label                     "volatility of feature"@en ;
  rprov:volatilitylevel          "Medium volatility" ;
  rprov:toFeature                <URI-Feature> ;
  rprov:wasGeneratedByDUA        <URI-DeterminationOfVolatilityOfFeature> ;
  prov:generatedAtTime           ^^xsd:dateTime ;
  prov: invalidatedAtTime        ^^xsd:dateTime .
```

Data restriction

The RDF triples needed to store the data restriction in the Data Understanding step are illustrated below. Table 28 shows the data restriction entity *rprov:DataRestriction* that contains the information about the data restriction via the datatype property *rprov:restriction* shown in Table 29. The link between the data restriction entities to the RDF containers containing the data restrictions are realized via *rdf:Seq* or *rdf:Bag* for the respective level of scale.

Table 28 RDF Data Understanding DataRestriction entity

Subject	Predicate	Object
rprov:DataRestriction	rdf:type	owl:Class
	rdfs:subClassOf	rprov:DataUnderstandingEntity
	rprov:wasGeneratedByDUA	rprov:DeterminationOf DataRestriction

Table 29 RDF Data Understanding restriction datatype property

Subject	Predicate	Object
rprov:restriction	rdf:type	owl:DatatypeProperty
	rdfs:domain	rprov: DataRestriction
	rdfs:range	rdf:Seq, rdf:Bag

Code 7 shows an example of a data restriction. A Data Understanding activity *<URI-Determination-OfDataRestriction>* creates a Data Understanding entity *<URI-DataRestrictionOfFeature>* that has a predicate *rprov:restriction* to an instance of a container. This container *rdf:Seq* for cardinal and ordinal features and *rdf:Bag* for nominal features contains the information about the restrictions of the feature in *xsd:string* format.

Code 7 Data Understanding data restriction example

```
<URI-DeterminationOfDataRestriction>
 rdf:type                           rprov:DeterminationOfDataRestriction,
                                    owl:NamedIndividual ;
  rdfs:label                        "detDataRestriction"@en ;
  prov:endedAtTime                  xsd:dateTime .

<URI-DataRestrictionOfFeature>
  rdf:type                          rprov:DataRestriction, owl:NamedIndividual ;
  rdfs:label                        "restriction age"@eng ;
  rprov:restriction                 <URI-DataRestrictionSeq> ;
  rprov:toFeature                   <URI-Feature> ;
  rprov:wasGeneratedByDUA           <URI-DeterminationOfDataRestriction> ;
  prov:generatedAtTime              ^^xsd:dateTime ;
  prov: invalidatedAtTime           ^^xsd:dateTime .


<URI-DataRestrictionSeq>
  rdf:type                          rdf:Seq , owl:NamedIndividual ;
  rdf:_0                            "20" ;
  rdf:_1                            "95" .

<URI-DataRestrictionOfNominalFeature>
  rdf:type                          rprov:UniqueValuesOfFeature , owl:NamedIndividual ;
  rdfs:label                        "restriction job"@eng ;
  rprov:toFeature                   <URI-Feature> ;
  rprov: restriction                <URI-DataRestrictionBag> ;
  rprov:wasGeneratedByDUA           <URI-DeterminationOfDataRestriction> ;
  prov:generatedAtTime              ^^xsd:dateTime ;
  prov: invalidatedAtTime           ^^xsd:dateTime .

<URI-DataRestrictionBag>
  rdf:type                          rdf:Bag , owl:NamedIndividual ;
  rdf:_0                            "admin." ;
  rdf:_1                            "blue-collar" .
```

Sensor precision

To store the sensor precision of a feature, the RDF triples listed in Table 30 and Table 31 are needed. The first table represents the Data Understanding entity, which together with the datatype property in the second table contains the information about the sensor precision.

Table 30 RDF Data Understanding SensorPrecisionOfFeature entity

Subject	Predicate	Object
rprov:SensorPrecisionOfFeature	rdf:type	owl:Class
	rdfs:subClassOf	rprov:DataUnderstandingEntity
	rprov:wasGeneratedByDUA	rprov:DeterminationOfSensorPreci-sionlevel

Table 31 RDF Data Understanding sensor datatype property

Subject	Predicate	Object
rprov:sensor	rdf:type	owl:DatatypeProperty
	rdfs:domain	rprov:SensorPrecisionOfFeature
	rdfs:range	xsd:float

Code 8 shows an example of an activity *rprov:DeterminationOfSensorPrecisionOfFeature*, which creates a *rprov:SensorPrecisionOfFeature* instance *<URI-SensorPrecisionOfFeature>*. This instance contains the information of the sensor precision via *rprov:sensor* and is assigned to a feature via *rprov:toFeature*.

Code 8 Data Understanding sensor precision example

```
<URI-DeterminationOfSensorPrecisionOfFeature>
  rdf:type                      rprov:DeterminationOfSensorPrecisionOfFeature,
                                owl:NamedIndividual ;
  rdfs:label                    "detSensorPrecisionOfFeature"@en ;
  prov:endedAtTime              xsd:dateTime .

<URI-SensorPrecisionOfFeature>
  rdf:type                      rprov:SensorPrecisionOfFeature, owl:NamedIndividual ;
  rdfs:label                    "SensorPrecisionOfFeature"@en ;
  rprov:sensor                  0.05 ;
  rprov:toFeature               <URI-Feature> ;
  rprov:wasGeneratedByDUA       <URI-DeterminationOfSensorPrecisionOfFeature> ;
  prov:generatedAtTime          ^^xsd:dateTime ;
  prov: invalidatedAtTime       ^^xsd:dateTime .
```

6.2.3. Data Preparation

This chapter stores the information of the Data Preparation step. For this purpose, Table 32 lists the RDF triples of the Data Preparation activity that are needed as soon as the user carries out a documentation on the binning of features or a documentation on the handling of missing values.

Table 32 RDF Data Preparation activities

Subject	Predicate	Object
rprov:DocumentationOf RangeOfBinnedFeatures HandlingOfMissingValues	rdf:type	owl:Class
	rdfs:subClassOf	rprov:DataPreparationActivity
	rdfs:label	xsd:string
	prov:endedAtTime	xsd:dateTime

Binned Feature

To store the bins of a feature, the RDF triples listed in Table 33 and Table 34 are needed. The first table represents the Data Preparation entity, which together with the datatype property in the second table contains the information about the bins.

Table 33 RDF Data Preparation RangeOfBinnedFeature entity

Subject	Predicate	Object
rprov:RangeOfBinnedFeature	rdf:type	owl:Class
	rdfs:subClassOf	rprov:DataPreparationEntity
	rprov:wasGeneratedByDPA	rprov:DocumentationOfRangeOf-BinnedFeatures

Table 34 RDF Data Preparation range datatype property

Subject	Predicate	Object
rprov:range	rdf:type	owl:DatatypeProperty
	rdfs:domain	rprov:RangeOfBinnedFeature
	rdfs:range	rdf:Seq

Code 9 shows an example of a *rprov:DocumentationOfRangeOfBinnedFeatures* activity that creates an instance of a *rprov:RangeOfBinnedFeature* entity. This contains the information of the bins via *rprov:range*, which is an ordered *rdf:Seq*. Via *rprov:toFeature*, the *<URI-RangeOfBinnedFeature>* instance is assigned to a feature.

Code 9 Data Preparation binning example

```
<URI-DocumentationOfRangeOfBinnedFeatures>
  rdf:type                   rprov:DocumentationOfRangeOfBinnedFeatures,
                             owl:NamedIndividual ;
  rdfs:label                 "DocuOfRangeOfBinnedFeature"@en ;
  prov:endedAtTime           xsd:dateTime .

<URI- RangeOfBinnedFeature >
  rdf:type                   rprov:RangeOfBinnedFeature, owl:NamedIndividual ;
  rdfs:label                 "RangeOfBinnedFeature"@en ;
  rprov:range                <URI-RangeSeq> ;
  rprov:toFeature            <URI-Feature> ;
  rprov:wasGeneratedByDUA    <URI-DocumentationOfRangeOfBinnedFeatures> ;
  prov:generatedAtTime       ^^xsd:dateTime ;
  prov: invalidatedAtTime    ^^xsd:dateTime .

<URI-RangeSeq>
  rdf:type                   rdf:Seq , owl:NamedIndividual ;
  rdf:_0                     "20.0" ;
  rdf:_1                     "57.5" ;
  rdf:_2                     "95.0" .
```

Missing values

To store the information about the missing values of a feature, the RDF triples shown in Table 35 and Table 36 are needed. The first table represents the Data Preparation entity, which together with the datatype property in the second table contains the information about the missing values.

Table 35 RDF Data Preparation HandlingOfMissingValues entity

Subject	Predicate	Object
rprov:HandlingOfMissingValues	rdf:type	owl:Class
	rdfs:subClassOf	rprov:DataPreparationEntity
	rprov:wasGeneratedByDPA	rprov:DocumentationOfHan-dlingOfMissingValues

Table 36 RDF Data Preparation missingValues datatype property

Subject	Predicate	Object
rprov:missingvalues	rdf:type	owl:DatatypeProperty
	rdfs:domain	rprov:HandlingOfMissingValues
	rdfs:range	xsd:string

Code 10 shows an example of a *rprov:DocumentationOfHandlingOfMissingValues* activity that creates an instance of a *rprov:HandlingOfMissingValues* entity. This contains the information about the missing values stored in an *xsd:string* format. Via *rprov:toFeature* the instance *<URI-HandlingOfMissingValues>* is assigned to a feature.

Code 10 Data Preparation missing values example

```
<URI- DocumentationOfHandlingOfMissingValues >
  rdf:type                      rprov:DocumentationOfHandlingOfMissingValues,
                                owl:NamedIndividual ;
  rdfs:label                    "DocuOfHandlingOfMissingValues"@en ;
  prov:endedAtTime              xsd:dateTime .

<URI-HandlingOfMissingValues>
  rdf:type                      rprov:HandlingOfMissingValues, owl:NamedIndividual ;
  rdfs:label                    "MissingValuesOfFeature"@en ;
  rprov:missingvalues           "Missing values were replaced with Median value" ;
  rprov:toFeature               <URI-Feature> ;
  rprov:wasGeneratedByDUA       <URI-DocumentationOfHandlingOfMissingValues> ;
  prov:generatedAtTime          ^^xsd:dateTime ;
  prov: invalidatedAtTime       ^^xsd:dateTime .
```

6.2.4. Modeling

This chapter shows how the information of the Modeling step is stored. For this purpose, Table 37 lists the RDF triples of the Modeling activity that are required as soon as the user defines a perturbation option or a perturbation mode.

Table 37 RDF Modeling activities

Subject	Predicate	Object
rprov:DefinitionOf perturbationMode perturbationOption	rdf:type	owl:Class
	rdfs:subClassOf	rprov:ModelingActivity
	rdfs:label	xsd:string
	prov:endedAtTime	xsd:dateTime

The RDF triples for the perturbation option generated in the Modeling step are shown in Table 38. A subclass of the Modeling entity *rprov:PerturbationOption* was generated by the Modeling activity *rprov:DefinitionOfPerturbationOption*, the entity is stored with the generated knowledge from the Business Understanding, Data Understanding and Data Preparation steps. The generated entities are passed via the predicate *rprov:modelingEntityWasDerivedFrom*.

Table 38 RDF Modeling PerturbationOption entity

Subject	Predicate	Object
rprov:PerturbationOption	rdf:type	owl:Class
	rdfs:subClassOf	rprov:ModelingEntity
	rprov:modelin-gEntityWasDerivedFrom	rprov:PerturbationApproach , rprov:ScaleOfFeature , rprov:VolatilityOfFeature , rprov:DataRestriction , rprov:SensorPrecisionOfFeature , prov:RangeOfBinnedFeature , rprov:HandlingOfMissingValue
	rprov:wasGeneratedByMA	rprov:DefinitionOfPerturbation-Option

Table 39 shows the RDF triples that are needed to store the perturbation mode. The *rprov:PerturbationMode* is a subclass of the Modeling entity and is instantiated via the object property *rprov:pertModeValue*, which has an *xsd:string* as *rdfs:range* for selecting between Full and Prioritized perturbation mode shown in Table 40.

Table 39 RDF Modeling PerturbationMode entity

Subject	Predicate	Object
rprov:PerturbationMode	rdf:type	owl:Class
	rdfs:subClassOf	rprov:ModelingEntity
	rprov:wasGeneratedByMA	rprov:DefinitionOfPerturbation-Option

Table 40 RDF Modeling pertModeValue object property

Subject	Predicate	Object
rprov:pertModeValue	rdf:type	owl:ObjectProperty
	rdfs:domain	rprov:PerturbationMode
	rprov:range	rprov:string

To assign the perturbation level to a perturbation option, the RDF triples shown in Table 41 are needed. These represent an object property with a *rprov:PerturbationOption* as *rdfs:domain* and a *xsd:string* as *rprov:range*. This can be used to assign the perturbation level red, orange or green to a perturbation option.

Table 41 RDF Modeling assignedPerturbationlevel object property

Subject	Predicate	Object
rprov:assignedPerturbationlevel	rdf:type	owl:ObjectProperty
	rdfs:domain	rprov:PerturbationOption
	rprov:range	xsd:string

Table 42 contains the object property that provides the perturbation options with the additional values, named *rprov:assignedPerturbationSetting*. For example, the settings for the percentage for a Percentage perturbation are stored here in *xsd:string*.

Table 42 RDF Modeling assignedPerturbationSetting object property

Subject	Predicate	Object
rprov:assignedPerturbationSetting	rdf:type	owl:ObjectProperty
	rdfs:domain	rprov:PerturbationOption
	rprov:range	xsd:string

Table 43 shows an object property named *rprov:modelingEntityWasDerivedFrom* which is needed to assign the created *rprov:DataPreparationEntity*, *rprov:DataUnderstandingEntity*, *rprov:BusinessUnderstandingEntity* to a perturbation option.

Table 43 RDF Modeling modelingEntityWasDerivedFrom object property

Subject	Predicate	Object
rprov:modelingEntityWasDerivedFrom	rdf:type	owl:ObjectProperty
	rdfs:domain	rprov:ModelingEntity
	rprov:range	rprov:DataPreparationEntity , rprov:DataUnderstandingEntity , rprov:BusinessUnderstandingEntity
	rdfs:subPropertyOf	prov:wasDerivedFrom

The RDF triples shown in Table 44 store the algorithm underlying a perturbation option. For this purpose, they are stored as *xsd:string* and assigned to the perturbation option with the datatype property *rprov:generationAlgorithm*.

Table 44 RDF Modeling generationAlgorithm datatype property

Subject	Predicate	Object
rprov:generationAlgorithm	rdf:type	owl:DatatypeProperty
	rdfs:domain	rprov:PerturbationOption
	rprov:range	xsd:string

Code 11 shows an example of the creation of a perturbation option. An instance of the Modeling entity is created by the *<URI-DefinitionOfPerturbationOption>* instance activity. This *<URI-PerturbationOption>* contains the information about *rprov:assignedPerturbationlevel*, *rprov:assignedPerturbationSetting* and *rprov:generationAlgorithm*, which are stored via *xsd:string*. In addition, the feature to be perturbed is stored via *rprov:toFeature*. Via *rprov:modelingEntityWasDerivedFrom* all selected instances of *<URI-PerturbationApproach>*, *<URI-ScaleOfFeature>*, *<URI-VolatilityOfFeature>*, *<URI-DataRestrictionOfFeature>*, *<URI-SensorPrecisionOfFeature>*, *<URI-RangeOfBinnedFeature>* and *<URI-HandlingOfMissingValues>* are stored.

Code 11 Modeling perturbation option example

```
<URI-DefinitionOfPerturbationOption>
  rdf:type                        rprov:DefinitionOfPerturbationOption,
                                  owl:NamedIndividual ;
  rdfs:label                      “Definition of perturbation option “@en ;
  prov:endedAtTime                xsd:dateTime .

<URI-PerturbationOption>
  rdf:type                        rprov:PerturbationOption, owl:NamedIndividual ;
  rdfs:label                      “perturbation option with settings”@eng ;
  rprov:assignedPerturbation      “Red” ;
        level
  rprov:assignedPerturbation      “{‘steps’: 5}” ;
        Settings
  rprov:toFeature                 <URI-Feature> ;
  rprov: wasGeneratedByMA         <URI-DeterminationOfDataRestriction> ;
  rprov:generationAlgorithm       “5% Perturbation” ;
  rprov:modelingEntityWas         <URI-PerturbationApproach>, <URI-ScaleOfFeature>,
        DerivedFrom               <URI-VolatilityOfFeature>, <URI-DataRestrictionOfFeature>, <URI-Sen-
                                  sorPrecisionOfFeature>, <URI-RangeOfBinnedFeature>, <URI-Han-
                                  dlingOfMissingValues> ;
  prov:generatedAtTime            ^^xsd:dateTime .
```

6.2.5. Deployment

In this chapter, the information of the Deployment step is stored. Table 45 lists the RDF triples of the Deployment activity that are required as soon as the user wants to perturb a classification case.

Table 45 RDF Deployment PerturbtationOfClassificationCase activity

Subject	Predicate	Object
rprov:PerturbationOfClassifica-tionCase	rdf:type	owl:Class
	rdfs:subClassOf	rprov:DeploymentActivity

Table 46 shows the Deployment entities *rprov:Assessment* and *rprov:Case*. These are used as super-classes for *rprov:PerturbationAssessment* and *rprov:ClassificationCase* shown in Table 47 and Table 48. The entity *rprov:PerturbationAssessment* is used as an instance of the actual perturbation, as can be seen in the example in Code 12. The instance *rprov:ClassificationCase* stores the information about the original classification case.

Table 46 RDF Deployment Assessment and Case entity

Subject	Predicate	Object
rprov:Assessment	rdf:type	owl:Class
rprov:Case	rdfs:subClassOf	rprov:DeploymentEntity

Table 47 RDF Deployment PerturbationAssessment entity

Subject	Predicate	Object
rprov:PerturbationAssessment	rdf:type	owl:Class
	rdfs:subClassOf	rprov:Assessment
	rprov:deploymentEntity WasDerievedFrom	rprov:PerturbationOption , rprov:PerturbationMode
	rprov:wasGeneratedByDA	rprov:PerturbationOfClassifica-tionCase

Table 48 RDF Deployment ClassificationCase entity

Subject	Predicate	Object
rprov:ClassificationCase	rdf:type	owl:Class
	rdfs:subClassOf	rprov:Case
	rprov:wasAssignedTo DeploymentEntity	rprov:PerturbationAssessment

The RDF triples shown in Table 49 are needed to assign the perturbation options to a perturbation assessment.

Table 49 RDF Deployment deploymentEntityWasDerivedFrom object property

Subject	Predicate	Object
rprov:deploymentEnti-tyWasDerievedFrom	rdf:type	owl:ObjectProperty
	rdfs:domain	rprov:DeploymentEntity
	rprov:range	rprov:ModelingEntity
	rdfs:subPropertyOf	prov:wasDerivedFrom

For the assignment of the instance *rprov:ClassificationCase* to a perturbation assessment, the RDF triples described in Table 50 are required.

Table 50 RDF Deployment wasAssignedToDeploymentEntity object property

Subject	Predicate	Object
rprov:wasAssignedToDeploymen-tEntity	rdf:type	owl:ObjectProperty
	rdfs:domain	rprov:Case
	rprov:range	rprov:DeploymentEntity
	rdfs:subPropertyOf	prov:wasDerivedFrom

Table 51 contains the perturbed testcases, which are stored using rprov:range *xsd:string and rdfs:domain rprov:PerturbationAssessment.*

Table 51 RDF Deployment perturbedTestcase datatype property

Subject	Predicate	Object
rprov:perturbedTestcase	rdf:type	owl:DatatypeProperty
	rdfs:domain	rprov:PerturbationAssessment
	rprov:range	xsd:string

The RDF triples shown in Table 52 are needed to store the input values of a classification case and to link them to *rprov:ClassificationCase* to document the input values.

Table 52 RDF Deployment values datatype property

Subject	Predicate	Object
rprov:values	rdf:type	owl:DatatypeProperty
	rdfs:domain	rprov:ClassificationCase
	rprov:range	rprov:string

Code 12 shows a perturbation example. As in the other steps, a Deployment activity <*URI-PerturbationOfClassificationCase*> creates a Deployment entity <*URI-PerturbationAssessment*>. This contains all the necessary information about the perturbation. Via *rprov:deploymentEntityWasDerivedFrom*, all instances of the perturbation options used for the perturbation assessment are stored. The object property *rprov:pertModeValue*, shown in Table 40, is used to store which perturbation the user has selected. Since the number of perturbed cases is growing exponentially, it was decided to include a reference in *rprov:perturbedTestCase* under which CSV file name the perturbation assessment was saved.

The entered values of the classification cases are stored in the <*URI-ClassificationCase*> instance in *rprov:values* and linked to the <*URI-PerturbationAssessment*> instance via *rprov:wasAssignedToDeploymentEntity*.

Code 12 Deyployment perturbation example

```
<URI-PerturbationOfClassificationCase>
 rdf:type                          rprov:DefinitionOfPerturbationOption,
                                   owl:NamedIndividual ;
  rdfs:label                       "PerturbationOfClassificationCaseLabel"@en ;
  prov:endedAtTime                 xsd:dateTime .

<URI-PerturbationAssessment>
  rdf:type                         rprov:PerturbationAssessment, owl:NamedIndividual ;
  rdfs:label                       "PerturbationAssessment"@eng ;
  rprov:deploymentEntity           <URI-PerturbationOption> ;
         WasDerivedFrom
  rprov:pertModeValue              "Full" ;
  rprov:perturbedTestCase          "Saved as csv with name: PerturbationAssessment" ;
  rprov:wasGeneratedByDA           <URI-PerturbationOfClassificationCase> ;
  prov:generatedAtTime             ^^xsd:dateTime .

<URI-ClassificationCase>
  rdf:type                         rprov:ClassificationCase, owl:NamedIndividual ;
  rdfs:label                       "ClassificationCaseLabel"@en ;
  rprov:values                     "{'age': 18.0, 'job': 'blue-collar', 'marital': 'divorced', 'education': 'primary',
                                   'default': 'no', 'balance': 0.0, 'housing': 'no', 'loan': 'no', 'day': '1', 'month':
                                   'jan', 'campaign': '1', 'pdays': -1.0, 'previous': 0.0, 'poutcome': 'failure',
                                   'prediction': 0}"@en ;
  rprov:wasAssignedTo              <URI-PerturbationAssessment> ;
         DeploymentEntity
  prov:generatedAtTime             ^^xsd:dateTime .
```

7. User study

To test the usability of the web-based user interface, an evaluation was carried out. A qualitative study in the form of a laboratory study was chosen for this purpose (Rosenzweig, 2015). A qualitative study was deliberately chosen because the evaluation aims to analyze the behavior of the test persons during the use of the web-based user interface.

The number of test subjects was six. In addition, three of these test subjects were used a second time for an evaluation after their feedback had been incorporated. For this user study, the test subjects were divided into the two target groups developers and analysts. The subjects testing the developer viewpoint all had knowledge of data mining and classification models as well as CRISP-DM.

The second group was selected on the basis that they did not have any advanced knowledge on the topics data mining, classification models and CRISP-DM. However, both groups received the same basic introduction to CRISP-DM, the classification models, and the reference process. The ratio of the groups was four developers to two analysts.

At the beginning of the evaluation with the test subjects, the basic idea of the CRISP-DM was presented to the test subjects. Followed by a rudimentary introduction to the classification to give the test subjects, who had no previous knowledge, a basic understanding. During the introduction of the reference process, the individual steps and activities were discussed. Special attention was paid to explaining the perturbation methodology in detail to all test subjects.

The evaluation always followed the same pattern. At first, the test subjects were not given a task and were supposed to try to go through the reference process without help. The aim was to check the basic structure of the web-based user interface and the additional information. In a second round, the test subjects were given tasks that they were to implement in a targeted manner. This was done to check whether the functions offered were user-friendly and understood. In the process, the test subjects were asked to express their thoughts on each individual step and to describe their approach to improve usability.

The test subjects which tested the web-based user interface as developers went through the entire life cycle. The first difficulty was noticed by all test subjects already at the first step, the creation of a new record on the Apache Jena Fuseki server. The main reason was that the page was divided into two tabs, database, and upload. This division confused the subjects, as it seemed unclear where a new record could be created. To counteract this confusion, the naming was standardized, the expander for creating a record was moved to the top and a description was added.

It should be noted that all respondents were of the opinion that info boxes would be advantageous for the usability of the web-based user interface. In these, the basic steps were recorded, what is to be done on the respective page and what the individual values mean. This step already helped the other subjects after implementation. Other information that did not help the test subjects or was confusing was removed. For example, the subjects found it confusing that the perturbed test cases were displayed in three different tables. First the prediction of the original use case was shown, then the perturbed test cases with the old prediction and finally the perturbed test cases with old and new prediction. This has been shortened in the final version so that only the last table with all information is displayed.

When setting the level of measurements of the individual features, it was criticized that the default value is nominal. This can lead to confusion for many features as to which feature has already been set.

At the time of the tests, a slider was available for setting the data restriction. As it turned out that the handling of the slider was not helpful for some test subjects to obtain accurate input values, it was removed. Furthermore, it was criticized that a description was missing for what a label for a data restriction is needed.

Lastly, the inconsistent coloring of the upload buttons was criticized in some cases. In addition to the aspects criticized by the test subjects, other processes on individual pages have been improved without causing any difficulties. For example, it is now obligatory that a deploy entity receives a label.
It should be noted that no qualitative study was conducted that calculates usability metrics. In addition, there is a quick learning effect with this web-based user interface, which already occurred after the first application for some test subjects.

8. Conclusions

The design and implementation of a web-based user interface for the guided assessment of reliability of classification results using the perturbation approach, as proposed by Staudinger et al. (2023), are described in this master’s thesis and answers the research question: *"How can a tool be developed to assist users in evaluating the reliability of predictive analytics results?"*.

Following the steps of the Cross-Industry Standard Process for Data Mining, the web-based user interface provides users the possibility to perform reliability assessment-related activities during each of the steps. Regarding Data Understanding, the web-based user interface offers users the opportunity to document the scale, volatility, data restrictions, and feature sensor precision for each feature. Additionally, the order of ordinal data can be documented, and unique values can be uploaded. For the steps of Data Preparation, the ability to document Equal Width binning and the handling of missing values has been implemented. Furthermore, users are empowered to upload, select, or create and save a classification prediction model. In the Modeling step, users are presented with exemplarily implemented perturbation options for selection, which can be configured and saved. Information which is gathered during these activities is used during the Deployment step to assess the reliability of an individual classification prediction result by utilizing the perturbation approach.

In summary, this master thesis represents a significant advancement for assessing the reliability of classification results by using the perturbation approach. By exemplifying a part of the reliability process for classification models, organizations could enhance the utilization of their models and make well-informed decisions based. Future research should concentrate on incorporating additional features and evaluating the process in a real-world context to assess and enhance its efficacy. For example, the next steps for this web-based user interface may involve implementing additional perturbation options or enhancing the existing ones with further configuration capabilities. Furthermore, the incorporation of the local quality measures approach could provide users with additional means to assess the reliability of their individual predictions.

References

Bargmeyer, B. E., & Gillman, D. W. (2000). Metadata standards and metadata registries: An overview. International Conference on Establishment Surveys II, Buffalo, New York,

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web: A New Form of Web Content That is Meaningful to Computers Will Unleash a Revolution of New Possibilities. ScientificAmerican.com.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T. P., Shearer, C., & Wirth, R. (2000). CRISP-DM 1.0: Step-by-step data mining guide.

ECMA. (2017). The JSON data interchange syntax. Retrieved 12.03 from https://www.ecma-international.org/publications-and-standards/standards/ecma-404/

Fisher, M. J., & Marshall, A. P. (2009). Understanding descriptive statistics. Australian Critical Care, 22(2), 93-97. https://doi.org/https://doi.org/10.1016/j.aucc.2008.11.003

Foundation, A. S. (2023). Apache Jena Fuseki. Retrieved 12.03 from https://jena.apache.org/documentation/fuseki2/

Kang, H. (2013). The prevention and handling of the missing data. Korean J Anesthesiol, 64(5), 402-406. https://doi.org/10.4097/kjae.2013.64.5.402

Kejriwal, M. (2019). What Is a Knowledge Graph? In M. Kejriwal (Ed.), Domain-Specific Knowledge Graph Construction (pp. 1-7). Springer International Publishing. https://doi.org/10.1007/978-3-030-12375-8_1

Kordon, A. K. (2020a). Artificial Intelligence-Based Data Science Solutions. In A. K. Kordon (Ed.), Applying Data Science: How to Create Value with Artificial Intelligence (pp. 69-122). Springer International Publishing. https://doi.org/10.1007/978-3-030-36375-8_3

Kordon, A. K. (2020b). Data Preparation. In A. K. Kordon (Ed.), Applying Data Science: How to Create Value with Artificial Intelligence (pp. 221-249). Springer International Publishing. https://doi.org/10.1007/978-3-030-36375-8_8

Krcmar, H. (2015). Informationsmanagement. https://doi.org/10.1007/978-3-662-45863-1

Mohammedali, N. (2019). Recommendation System Based on Graph Database Techniques. 6, 754-763.

Morgenthaler, S. (2009). Exploratory data analysis. WIREs Computational Statistics, 1(1), 33-44. https://doi.org/https://doi.org/10.1002/wics.2

Moro, S., Cortez, P., & Rita, P. (2014). A data-driven approach to predict the success of bank telemarketing. Decision Support Systems, 62, 22-31. https://doi.org/https://doi.org/10.1016/j.dss.2014.03.001

Plotnikova, V., Dumas, M., & Milani, F. P. (2022). Applying the CRISP-DM data mining process in the financial services industry: Elicitation of adaptation requirements. Data & Knowledge Engineering, 139, 102013. https://doi.org/https://doi.org/10.1016/j.datak.2022.102013

Rosenzweig, E. (2015). Chapter 7 - Usability Testing. In E. Rosenzweig (Ed.), Successful User Experience: Strategies and Roadmaps (pp. 131-154). Morgan Kaufmann. https://doi.org/https://doi.org/10.1016/B978-0-12-800985-7.00007-7

Schröer, C., Kruse, F., & Gómez, J. M. (2021). A Systematic Literature Review on Applying CRISP-DM Process Model. Procedia Computer Science, 181, 526-534. https://doi.org/https://doi.org/10.1016/j.procs.2021.01.199

scikit. (2023a). Documentation scikit OneHotEncoder. Retrieved 13.03 from

scikit. (2023b). Documentation scikit Ordinal Encoder. Retrieved 13.03 from https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OrdinalEncoder.html#sklearn.preprocessing.OrdinalEncoder

Staudinger, S., Schütz, C. G., & Schofel, M. (2023). A Reference Process for Assessing the Reliability of Predictive Analytics Results

Streamlit. (2023). Streamlit documentation. https://docs.streamlit.io/

Tukey, J. W. (1977). Exploratory data analysis. Addison-Wesley.

W3C. Universal Resource Identifiers. Retrieved 12.03 from https://www.w3.org/Addressing/URL/URI_Overview.html

W3C. (2004). RDF Test Cases. Retrieved 12.03 from https://www.w3.org/TR/rdf-testcases/#ntriples

W3C. (2008a). Extensible Markup Language (XML) 1.0 (Fifth Edition). Retrieved 12.03 from https://www.w3.org/TR/xml/

W3C. (2008b). SPARQL Query Language for RDF. Retrieved 12.03 from https://www.w3.org/TR/rdf-sparql-query/

W3C. (2011a). Notation3 (N3): A readable RDF syntax. Retrieved 12.03 from https://www.w3.org/TeamSubmission/n3/

W3C. (2011b). Turtle - Terse RDF Triple Language. Retrieved 12.03 from https://www.w3.org/TeamSubmission/turtle/

W3C. (2012). Web Ontology Language (OWL). Retrieved 12.03 from https://www.w3.org/OWL/

W3C. (2013). PROV-O: The PROV Ontology. Retrieved 12.03 from https://www.w3.org/TR/prov-o/

W3C. (2014a). RDF Schema 1.1. Retrieved 13.03 from https://www.w3.org/TR/rdf-schema/

W3C. (2014b). Resource Description Framework (RDF). Retrieved 12.03 from https://www.w3.org/RDF/

Wu, C., Kao, S.-C., & Okuhara, K. (2013). Examination and comparison of conflicting data in granulated datasets: Equal width interval vs. equal frequency interval. Information Sciences, 239, 154-164. https://doi.org/https://doi.org/10.1016/j.ins.2013.03.014

Zaki, M. J., & Jr, W. M. (2014). Data Mining and Analysis: Fundamental Concepts and Algorithms. Cambridge University Press.