Application-grounded evaluation of predictive model explanation methods

Lin, C.

Technische Universiteit **Eindhoven** University of Technology

Department of Mathematics and Computer Science
Data Mining Research Group

Application-grounded evaluation of predictive model explanation methods

Master's Thesis

Chin-Fang Lin

Final version

Eindhoven, September 2018

Abstract

The current fraud detection system at Rabobank suffers from high false alarm rate. Therefore, a Machine Learning (ML) model has been introduced to learn the behaviour of fraud and reduce the false alarm rates. In addition to the model accuracy, the reason why a model makes a specific decision is also crucial to understand. Hence, the explanation model is applied to enhance the interpretability of the ML model. We evaluate explanation interpretability by providing useful information to domain experts and, further, by performing the reduce false alarms task automatically. The evaluation is performed on two model-agnostic explanation methods: Local Interpretable Model-agnostic Explanations (LIME) and SHapley Additive exPlanations (SHAP).

In applying explanations to provide useful information, two experiments were conducted. The first experiment was done to investigate the relation between model accuracy and interpretability. The second experiment was done to discover the utility of the explanation on aggregating local explanation into group and global levels.

To investigate the relation between model accuracy and interpretability, six ML models with different accuracy scores and biases were built. The soundness between each explanation decreasing with the accuracy score was observed. However, the behaviour of the explanations became diverse when we encountered different model complexity, class distribution and dimension of data. These results make it challenging to define the ML model accuracy threshold regarding interpretability. Hence, we concluded the experiment with an unknown threshold and moved to the utility on aggregating explanation at different levels.

To discover the utility of the explanation at different levels, the utility of two state-of-art model-agnostic explanation models, SHAP and LIME, were investigated. Based on their strengths and weakness, we aggregated the explanation at different levels and applied visualization techniques to derive useful explanations in a case study. The desired useful explanations are fulfilled and summarized. The possible utility for the domain experts is also demonstrated.

In the false positive elimination task, two approaches are proposed to filter out the false alarms by explanation feature values and extracted rules, respectively. In the first approach, the ML filter was built by substituting the explanation feature values for the original feature values. The performance of the ML filter can outperform the ML model on different types of fraud. With the same true positive rate, the false positive rate of the ML filter was 40% less and 32% less than the ML model in Mo1 and Mo2 respectively. Hence, the explanation values are recommended for future use to automatically post-process the fraud alerts. In the second approach, although the ML filter cannot achieve better performance, the extracted rules can still provide insight on describing false positives.

Preface

This thesis is the result of the master graduation project for Data Science Engineering at Eindhoven University of Technology. The research of this project is performed within the Data Mining Group of the TU/e in collaboration with Rabobank located in the Netherlands.

First, I would like to thank my supervisor prof. dr. Mykola Pechenizkiy, he provided valuable motivation and guidance to clean my thoughts. No matter how busy he was, he always scheduled a meeting to help me with doubts. Secondly, I would like to thank PhDs who collaborate with Rabobank for their research, Simon and Tita, for kindly sharing their experience and having relax coffee break. Especially Simon always invests his time in formalizing my questions and inspiring me new ideas. I would also like to thank the committee, dr. Odysseas Papapetrou, who provided valuable comments on improving the thesis. From Rabobank, I would also like to thank Werner van Ipenburg and Jan W. Veldsink for their valuable feedback from both machine learning and business points of view. Furthermore, I would also like to thank my friends and colleagues for their constant support. Finally, I want to thank my family for their support throughout these two years of study.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

From self-driving cars to email spam filtering, machine learning (ML) systems are becoming increasingly ubiquitous. They outperform humans in specific tasks, and they are gradually being used to guide human decisions. The ML systems deployed are expected not only to perform tasks but also to meet other criteria, such as safety, non-discrimination, faithfulness, and the ability to provide the right to explanation. However, unlike the measurement of accuracy, these criteria usually cannot be quantified completely. This situation introduces the need for model *interpretability*: if a model decision can be explained in understandable terms to humans, then we can verify whether the reasoning is sound.

The evaluation of interpretability can be grouped into two types [4]. The first type evaluates interpretability in the context of application-based evaluation. The assumption is that if an ML system is useful in performing a task, then it must be interpretable on a certain level. The second type evaluates interpretability in the context of function-based evaluation. A common approach is to restrict the ML model to some interpretable class, such as decision trees and decision lists, and then claim that the model can reach accuracy as well as more complicated models can.

In this thesis, we investigate application-based evaluation on the model-agnostic local explanation model. The model-agnostic local explanation model is one approach of post-hoc interpretation. Post-hoc interpretations explain predictions without elucidating the mechanisms by which models work; they may confer the useful information for the end user of machine learning [7]. Hence, it can be used to explain black-box models without sacrificing predictive performance. The model-agnostic local explanation model can therefore be introduced to solve the banks fraud detection problem.

The popularity of online banking systems has led to an increase in the cases of fraud each year. Online banking fraud is difficult to detect because the fraudulent behaviour is very dynamic, buried in different customers' profiles, dispersed across highly imbalanced large data sets and occurs within a very limited time, thus requiring real-time detection [18]. Moreover, most customers rarely check their online banking history regularly, so they often cannot report fraudulent transactions immediately.

1.1 Motivation

In the current fraud detection system at Rabobank, the rule-based detector is used to identify fraudulent online transfer behaviours. The detector continuously generates large amounts of fraud alerts every day. However, most of these alerts are false alarms that can result in block the transaction and frozen online banking functions for customers. Once the rule-based detector generates an alert, domain experts need to analyze the alert within a small timeframe and manually scan through different attributes for each type of fraud. The description of rules also require regularly update in order to reduce the number of these false alarms. This process demands much effort and is time-consuming. Therefore, an ML model can be introduced to learn the behaviour of fraud and reduce the false alarm rates.

However, the existing issue is the trade-off between model accuracy and interpretability, where more accurate ML algorithms are usually harder to interpret by humans. The accurate ML model is a black box and there is no extra information can be used to assist domain experts in post processing the alerts. As a result, the explanation model is applied to enhance the interpretability of the ML model while retaining the prediction performance. In this thesis, we investigate the utility of the explanations for domain experts and perform false alarms elimination task automatically. The pros and cons of each system is summarized in Table 1.1.

	Pros	**Cons**
Rule-based detector	Easy to understand	High FPR
Accurate ML model	Better TPR/FPR	No help in processing positives
Accurate ML model + explanation	Better TPR/FPR Extra information for domain experts	Utility of explanation not studied

Table 1.1: The pros and cons of each fraud detection system.

1.2 Goals and Challenges

The main goal of this thesis is to evaluate model-agnostic explanation methods Local Interpretable Model-agnostic Explanations (LIME) and SHapley Additive exPlanations (SHAP) on providing useful information to a domain expert and further assisting in the false alarm elimination task. The first goal is to evaluate the interpretability of explanation by soundness of the explanation. Once the relation between model accuracy and the soundness of the explanation is established, we can provide the information to domain experts. The explicit accuracy threshold for interpretable ML model may also be defined. The second goal is to discover the utility of the explanation on aggregating local explanation into group and global levels. A local explanation is only helpful when we have an interest in a specific case. Hence, we aggregate local explanation and discover the useful information that can be provided. The third goal is applying explanation to assist domain experts in automatically eliminating false alarms while retaining most of the true positives. The evaluation of usefulness is compared to the performance with other baselines.

The main challenges in achieving the research goal can be grouped into three. To achieve the first goal, the link between quantitative accuracy score and qualitative soundness need to be established. A useful explanation needs to be defined in order to achieve the second goal. A way to present the model explanation in terms that are understandable to humans needs to be developed as well. For the third goal, a framework will be established based on the explanation values. To the best of our knowledge, no previous research has applied explanation values to eliminate false alarms. The explanation values can be considered as the attributed importance of each alert. However, how the attributed importance can relate to false alarms and true positives is not obvious. The relationship between attributed values and corresponding importance is also unclear. Furthermore, the imbalanced class distribution in false alarms and true positives increases the difficulty of learning the behaviour of alerts.

1.3 Approaches

The first experiment was conducted on six ML models with different accuracy scores and biases. State-of-the-art model-agnostic explanation methods provided a local explanation for each instance to ensure local accuracy. The explanation can be visualized in a single instance or aggregately for all instances. From an explanation of a single instance, it is difficult to judge the relationship between the local explanation and the ML models accuracy. Since it only reflects the local behaviour of the model and the result is highly dependent on the chosen instance, we therefore grouped the local explanation into the model level and visualized the result.

In the second experiment, we investigated the utility of explanation on model, group and local levels. Since the connection between ML explanations and humans desired explanations remains unclear, we addressed the problem by starting from a human perspective and then applying visualization techniques to derive useful explanations. The following types of explanations will be discussed: general, abnormal, contrastive and selected.

In the third experiment, we investigated the utility of processing alerts and reducing false alarm rates in a more efficient way to assist domain experts. We proposed two approaches to solve this problem through the application of the explanation values of each alert. In the first approach, we built an ML filter by substituting the explanation feature values for their original values. We expected that some of the characteristics of false positives and true positives are reflected by the explanations. In the second approach, we transferred the information from the explanation values to the original feature values and built a false positives filtering layer. To do so, we clustered the local explanations into several groups and observed the dominant features in each group. After gaining an understanding of the ML model, we found rules that map these clusters from the explanation values back to the original feature domains. We did so by applying the subgroup discovery technique in each cluster to find subgroups that were as large as possible and had the most unusual distributional characteristics. The generated rules can help domain experts gain insights into the combination of features that may induce false alarms. Finally, we filtered out rules that have a high concentration of false positives and measure the output performance. Two other baselines were also designed to be compared with the proposed framework, which were filtering rules discovered without clustering and the original ML model.

1.4 Results

The result of the first experiment shows that the soundness between explanations decreases with the accuracy score in the simple models. However, the behaviour of the explanations became diverse when we encountered different model complexities, class distributions and dimensions of data. These results made it challenging to define the ML model accuracy threshold regarding interpretability. Hence, we concluded the experiment with an unknown threshold and moved to the utility of explanations.

Explanation utility is discovered from the model, group to local level. The properties of useful explanations attained on each level are summarized in Table 5.5. On the local level explanation, we showed that the output probability can be boosted by prior knowledge obtained from the explanations. Moreover, we present the possible utility of the explanations for domain experts.

In the application-grounded evaluation, we proposed two approaches to eliminate FPs automatically. In the first approach, we built an ML filter by substituting the explanation feature values for their original values. The performance of the ML filter are better than the ML model. Hence, the goal is achieved. The explanation values are recommended for future use to automatically post-process the fraud alerts. In the second approach, the visualization result illustrated the structure of dominant features in each cluster. The similar patterns occurred in both explanation values and the original features. The extracted rules from each cluster can represent the general rule that describes false positives of some clusters. However, those rules contained many true positives. As a result, the filter cannot outperform any baseline.

1.5 Thesis outline

This thesis is structured as follows. In Chapter 2, we discuss the problems that emerge when evaluating explanations on domain experts from different aspects, then we summarize the problems in the research questions. In Chapter 3, we describe state-of-the-art model-agnostic explanation methods and summarize their advantages and drawbacks. In Chapter 4, we establish the framework used to eliminate false alarms and describe each component in detail. In Chapter 5, we first investigate the relation between model accuracy and interpretability. Then we present the explanation utility at the global, group and local levels. In Chapter 6, we show the application-grounded evaluation results on the performance of the false alarm elimination framework. In Chapter 7, we draw conclusions based on the research questions, and then we discuss the limitations of our research and future work directions.

Chapter 2

Problem Statement

The main objective of this thesis is to evaluate model-agnostic explanation methods on providing useful information to domain experts and further assisting domain experts in the false alarm elimination task. The process of evaluation can be illustrated in Figure 2.1.

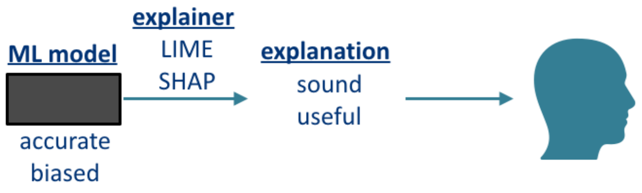


Figure 2.1: The process of evaluating interpretability of explanation methods.

From a black box ML model, the explanation model LIME or SHAP can be applied to explain the prediction result. There are many criteria can be used to evaluate interpretability, e.g., soundness, usefulness, completeness, compactness and comprehensibility. Concerning the utility for domain experts, the information regarding the relation between ML model accuracy and soundness of the explanation is helpful. *Soundness* means that everything an explanation says is true [6]. With this information, domain experts can have an expectation on the explanation with corresponding ML model accuracy. Moreover, if an accuracy threshold can be defined such that the given model is interpretable, the domain experts can trust ML model based on the accuracy. This problem is addressed in Section 5.1, where we discuss the soundness of explanation with reference ML models.

In addition to soundness, the usefulness can also be adopted to evaluate interpretability. The good explanation for humans can be considered as useful. Miller [12] explored various research fields and summarized what a good explanation is from different perspectives, such as philosophy, cognitive psychology and social psychology. A good explanation is general, abnormal, contrastive and selected. *General* is a cause that can explain most of the events is very general and can be considered a good explanation. In contrast to general causes, *abnormal* causes that rarely occur but influence the outcome are also good explanations. *contrastive* explanation that highlight the difference between the case of interest and the reference case are good to humans. Moreover, humans prefer to select one or two causes from a huge number of causes is the *selected* explanation. Molnar [13] combined two perspectives and described the meaning of good explanations in interpretable ML. We assume the useful explanation should also satisfy the types of good explanations. We discuss which useful information can be provided to domain experts by aggregating local explanations at different levels in Section 5.4.

To assist domain experts in the false alarm elimination task, the post process of alerts need to be conducted automatically, otherwise domain experts cannot analyze the alert within a small timeframe. In addition, it is also crucial to bridge the knowledge gap between rule-based detector and ML model. This could be done by discovering the rules captured by ML model and the rules induced more false positives. In Chapter 6, we propose a framework that combines an explanation model with different data mining techniques to eliminate false alarms. Note that the terms false alarm and false positives (FPs), as well as real alarm and true positives (TPs), are used interchangeably.

We address the above-mentioned problems and summarize the research questions as follows:

1. Will interpretability drop as the reference model become less accurate / biased?

   We investigate the following models with diverse accuracy and biased: *a*) perfect *b*) moderately accurate *c*) random *d*) imbalance *e*) high-dimensional *f*) low-dimensional. From observing relation between model accuracy and the soundness of the explanation, we might able to define an accuracy threshold such that the corresponding ML model is interpretable.

2. Which type of useful information can be obtained at the model-, group- or local-level explanations?

   We consider that good explanations are useful to humans. Hence, we started by defining the types of good explanations. Then, we investigated the utility of two state-of-art model-agnostic explanation models, SHAP and LIME. Based on their strengths and weakness, we aggregated the explanation at different levels and applied visualization techniques to derive useful explanations in a case study.

3. Can the explanations help a domain expert to post process fraud alerts?

   The explanation values of each alert can be considered as the importance of each feature. We expect that these explanation values can identify groups of obvious false alarms or actual fraud. Therefore, the explanation values can be adopted to automatically eliminate the false alarms.

4. Can the explanations indicate the structure of different fraud detector rules?

   Each rule consists of a number of mathematical and logical operations on specific features. In order to discover the dominant features in the rules, we cluster alerts based on their similarity in the explanation values. Then the features with a higher impact in each cluster can be explored. Moreover, we apply the rule mining technique to discover the pattern of alerts and extract relevant rules from each cluster.

Chapter 3

Background and related work

In this chapter, we introduce the state-of-art model-agnostic explanation methods LIME and SHAP. Afterwards, we compare the pros and cons of both methods and connect their strengths with the useful explanations.

3.1 Explanation methods

It is vital to ensure that ML models make decisions for the right reasons. Understanding why a model makes such a prediction can help the user to debug, gain trust in and further improve the existing model. This process falls into a research area called interpretability, which is defined as the ability to explain or present in understandable terms for humans [4].

Simple models, such as linear models and decision trees, usually cannot handle difficult tasks but can be interpreted by their weights and decision nodes. Complex models, such as ensemble methods or neural networks, typically have better predictive ability; however, they are difficult for humans to interpret. Hence, the explanation model uses a simpler model to locally approximate the original complex model and to return the explanation.

Recently, many current methods for interpreting machine learning predictions fall into the class of additive feature attribution methods. The explanation methods LIME [16] and SHAP [9][8] used in the following experiments also belong to this class.

3.1.1 Additive feature attribution methods

An additive feature attribution method explains the model output as the sum of the values attributed to each input feature. The explanation model can be written as a linear function of binary variables:

Definition 3.1.1 (Additive feature attribution methods)**.** The explanation model g can approximate output z' with the attribution value of each features ϕ_i.

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i \tag{3.1}$$

where $\phi_i \in R$, $z' \in \{0,1\}^M$ and M is the number of interpretable input features. The z'_i represent the appearance of the features. If the feature is observed then $z'_i = 1$, otherwise $z'_i = 0$.

3.1.2 LIME

The local explanation method LIME interprets an individual prediction by learning an interpretable model locally. The intuition behind LIME is that it samples instances both in the vicinity and far away from the interpretable representation of the original input. Then LIME takes the interpretable representation of these sample points, determines their predictions and builds a weighted linear model by minimizing the loss and complexity. The samples weighting is based on their distances from the original point. The points weights decrease as the points get farther away. The explanation is locally faithful, which means it represents the model prediction of vicinity instances. This is illustrated in Figure 3.1

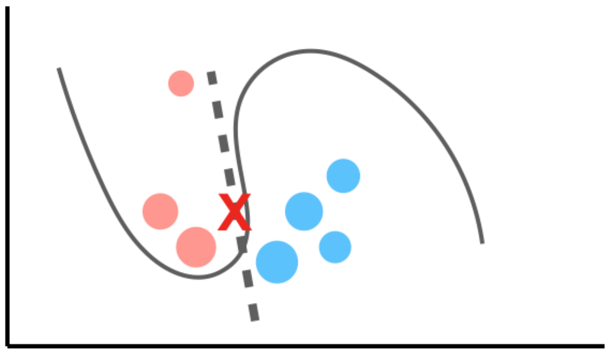


Figure 3.1: The black curve is the decision boundary of a complex model f. The red cross is the instance being explained. LIME samples instances (red and blue dots), weights them based on their distances from the original instance (represents by size) and determines predictions using f. The dashed grey line is the learned explanation model g that is locally faithful [16].

The interpretable representation can be the features themselves or the representation of the actually used features. For example, an interpretable representation for numerical classification is a binary vector indicating a presence or absence of the discretized feature. For text and image classification, an interpretable representation can be an appearance of a word or a superpixel.

We now formalize this intuition. Let f denote the complex model being explained. In classification $f(x)$ is the probability that an instance x belongs to the certain class. Let g be defined as explanation model $g \in G$, where G is a class of interpretable models, such as linear models and decision trees. In the current version of LIME, G is the class of linear models, such that $g(z') = w_g \cdot z'$.

As previously mentioned, LIME explains an instance by interpretable representation and draws the samples around it. Let x denotes the original instance, where $x \in R^N$ and N is the number of features in the original input. A mapping function is defined to convert the binary pattern of missing features represented by x' to its original feature value $x = h_x(x')$. The interpretable representation of input and the samples are denoted by binary vectors x' and z', where $x', z' \in \{0,1\}^M$. LIME draws samples around x' uniformly at random. Given a perturbed sample z', LIME can recover the sample to its original input z and obtain $f(z)$, which is used as a label for the explanation model.

To ensure the interpretability and local fidelity, LIME produces the explanation ξ by minimizing the loss $\mathcal{L}(f, g, \pi_x)$ and complexity $\Omega(g)$. $\mathcal{L}(f, g, \pi_x)$ measures how unfaithful g is in approximating f. $\Omega(g)$ measures the complexity of the explanation model. For example, $\Omega(g)$ can be the number of non-zero weights of linear models or the depth of the decision trees. The proximity measure between x and z is denoted as an exponential kernel π_x. Given the dataset $\mathcal{Z}$ of perturbed samples with associated labels, the explanation $\xi(x)$ can be obtained by optimizing the following equation:

$$\xi(x) = \underset{g \in G}{\arg\min}\, \mathcal{L}(f, g, \pi_x) + \Omega(g) \tag{3.2}$$

The loss $\mathcal{L}$ is computed by weighted square loss as Equation 3.3 , where $\pi_x(z)$ is an exponential kernel defined on some distance function with width.

$$\mathcal{L}(f, g, \pi_x) = \sum_{z, z' \in \mathcal{Z}} \pi_x(z)(f(h_x(z')) - g(z'))^2 \tag{3.3}$$

To solve the above equations and to obtain the explanation, LIME approximates $\Omega(g)$ by selecting K features with Lasso and then learning the weights via least squares.

The advantage of LIME is that the runtime does not depend on the size of the dataset but on the complexity of $f(x)$ and the number of samples. However, any choices of interpretable representation and G will have some drawbacks. Since the complex model can be treated as a black-box, the chosen interpretable representation might not fully explain specific behaviours. Additionally, a linear models G cannot provide faithful explanation if the input model is highly non-linear.

3.1.3 SHAP

SHAP assigns each feature importance based on Shapley values from game theory. Shapley values is a unique solution in the class of additive attribution methods that satisfies three desired properties: local accuracy, missingness and consistency. Local accuracy requires the sum of the feature contributions to be equal to the models prediction. Missingness requires the absent features in the original input to have no impact. Consistency ensures that when the model changes and a feature has a higher impact on the model, the importance of that feature will never decrease. Lundberg et al. [9] have proven the SHAP method is the only possible locally accurate and consistent method that satisfied the missingness property. With these properties, SHAP can provide theoretically justified improvements for existing explanation model such as LIME.

Shapley value is one way to distribute total gains of the contributors, assuming that all the contributors collaborate. Shapley value is calculated by considering all the possible orders of the contributors arrival and by computing each contributors marginal contribution. The following example depicts this.

Suppose there are two players A and B in the coalitional game, and their contribution values v are $v(A) = 6$, $v(B) = 10$ and $v(A, B) = 24$. The following table displays the marginal contributions of both players. Therefore, the expected marginal contribution and Shapley value for player$A$ is $0.5 * 6 + 0.5 * 14 = 10$ and for player$B$ is $0.5 * 18 + 0.5 * 10 = 14$.

order of arrivals	marginal contributions of A	marginal contributions of B
A, B	$v(\{A\}) = 6$	$v(\{A, B\}) - v(\{A\}) = 18$
B, A	$v(\{A, B\}) - v(\{B\}) = 14$	$v(\{B\}) = 10$

SHAP approximates the input value as a conditional expectation function of the original model $f_x(S) = f(h(z')) = E[f(x)|x_S]$, where S is a set of non-zero indexes in z', and $E[f(x)|x_S]$ is the expected value of the conditional function on a set of input features S. SHAP values combine these conditional expectations with the classic Shapley values from game theory to attribute ϕ_i values to each feature:

$$\phi_i = \sum_{S \subseteq M \setminus \{i\}} \frac{|S|!(M - |S| - 1)!}{M!} [f_x(S \cup i) - f_x(S)] \tag{3.4}$$

where M is the number of interpretable input features. This value is the marginal contribution when i feature adds to the set of feature S weighting according to the number of ways to devise this marginal calculation divided by the permutation of all features.

We describe how SHAP is adapted to combine with LIME and tree-based classification models to obtain full explanations for ML models as follows.

Kernel SHAP

LIME chooses the parameters heuristically and does not recover the Shapley values. Hence, the intuition of Kernel SHAP is to find the loss function $\mathcal{L}$, weighting kernel π_x, and the regularization term Ω that recover the Shapley values.

Lundberg et al. [9] have proven that the following solutions make Equation 3.2 recover the Shapley values and hold for the above three properties:

$$\Omega(g) = 0 \tag{3.5}$$

$$\pi_x(z') = \frac{M - 1}{(M choose |z'|)|z'|(M - |z'|)} \tag{3.6}$$

$$\mathcal{L}(f, g, \pi_x) = \sum_{z' \in \mathcal{Z}} [f(h_x(z')) - g(z')]^2 \pi_{x'}(z') \tag{3.7}$$

where $|z'|$ is the number of non-zero elements in z'.

Tree SHAP

Currently, popular feature attribution methods for tree ensembles are inconsistent. Thus, when a model changes, such as a feature having a higher impact on the models output, the current methods can actually lower the impact of that feature [8]. A tree SHAP model is proposed to provide a theoretical improvement by solving the inconsistency problem.

As a tree model, $E[f(x)|x_S]$ can be estimated by recursively computing the proportion of the training samples matching the conditioning set S multiplied with the node values. However, running the algorithm for all 2^M subsets for all the M features will cost $O(TL2^M)$ time. Recall T is the number of trees, L is the maximum number of leaves in any tree and M is the number of features. Hence, the polynomial time tree SHAP is developed to reduce the runtime to $O(TLD^2)$. The intuition behind this is to recursively track the proportion of all possible subsets flow down into each of the leaves of the tree. The current version Tree SHAP has been merged into XGBoost, LightGBM, CatBoost and scikit-learn tree models. The computation performance is more efficient than Kernel SHAP, since it estimates $E[f(x)|x_S]$ without building an extra linear model.

3.1.4 Summary

We summarize the pros and cons of the LIME and SHAP methods and connect their advantages to seek useful explanations. LIME provides selected explanations that can explain instances with fewer important features. This meets the selected property in a good explanation, since humans usually prefer sparse explanations. LIME is also capable of predicting the probability change of a label based on the changing of an input feature, which provides a good contrast explanation.

On the other hand, SHAP provides a full explanation that is crucial for decisions that must consider all effects, such as assisting doctors in clinical diagnosis and helping judges make careful decisions regarding defendants. SHAP also provides contrast explanations that allow the user to compare explanations on the group level. In addition, SHAP value cannot be considered as the probability decreased after removing the feature as in LIME. The value should be interpreted as the feature contribution to the difference between the actual prediction and the mean prediction. We will show an example of the misinterpretation in Section 5.5.3.

Chapter 4

FPs elimination framework

In this Chapter, we introduce the framework used in the FPs elimination experiment. The framework contains two approaches: FPs elimination by SHAP values and FPs elimination from extracted rules. Then, we describe each component in detail.

As discussed in Chapter 2, fraud detection is a streaming process where alerts are generated consecutively. A more reasonable approach is building an ML model on training data in a small interval and eliminating upcoming false alerts. Then, new ML models can be built for the next time intervals. The performance of the new ML model should increase due to the increasing size of the training set. However, training ML models and explanation models are time-consuming. To reduce the problem size, in this thesis we built an off-line static ML model on one-year transaction data from October 2016 to September 2017.

4.1 Framework

The framework for FPs elimination is illustrated in Figure 4.1. The proposed novel approaches are shown in the grey frame. First, we start with pre-processing the transaction data. The transaction dataset has a large number of features and is highly imbalanced. Under-sampling and feature selection techniques are applied to deal with these problems. Secondly, we build an ML model on the training set and predict on a test set. The cases predicted as a positive class are considered as alerts, while the cases predicted as negative are non-alerts. Thirdly, we use the Tree SHAP explanation model to obtain the explanations for the alerts. Each explanation of the alert consists of the SHAP value for all features. After the SHAP values are obtained, we propose the following two approaches to filter out FPs automatically.

In the first approach, SHAP values are applied to distinguish between FPs and TPs. Since the SHAP value of each feature is the decomposition of output probability, the sum of SHAP values should reflect that TPs have a higher probability of being an alert than FPs. Furthermore, SHAP values decompose this confidence; thus, we expect some features in this decomposition will dominate the true positive classifications and other features of the false positive classifications. Therefore, the SHAP values are considered as features to build another ML model for predicting FPs and TPs.

In the second approach, SHAP values are applied to discriminate FPs and TPs on the original feature space. Therefore, the elimination process does not rely on the SHAP values and can filter out FPs by the original features. We relate SHAP values with the original features by using clustering and subgroup discovery techniques. Clustering is applied to find the structure in SHAP values. After forming the clusters, the instances in each cluster are dominated by similar features. Then,

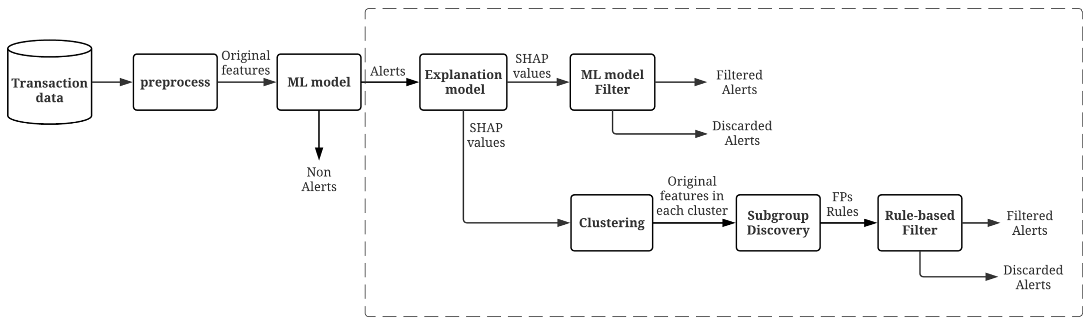


Figure 4.1: The framework of FPs elimination.

the problem remains how to extract the pattern to distinguish FPs and TPs. Thus, subgroup discovery is implemented to discover the rules to distinguish FPs and TPs in each cluster. To ensure the extracted rules also fulfil a certain confidence in all the alerts, the support and confidence of each rule are computed. Finally, the rules with higher FPs confidence and lower TPs support are used in the filter.

The two approaches are evaluated on the performance metric ROC-curve with the ML model and respective baseline (Section 6.2, Section 6.4). The extracted rules in the second approach are presented to a domain expert to validate their genuineness.

4.2 Pre-processing

The transaction dataset has numerous features and imbalanced classes. A useful explanation requires the ML model to have high accuracy; therefore, the pre-processing step is necessary.

The imbalanced class problem can be addressed on different levels: data level, algorithm level and combining methods [5]. At the algorithm level, it adjusts the costs of various classes to balance the classes. For example, different weights can be assigned to the respective classes in kNN, and the bias can be developed in SVM such that the learned hyperplane is further away from the major class. At the data level, there are many forms of re-sampling, such as random under-sampling, random over-sampling and the over-sampling technique SMOTE [3]. Combining methods combine the results of many classifiers, and each of them is built on under/over-sampling data with different sampling ratios. To avoid building an extra model on the massive transaction dataset, we focus on a data level algorithm. Over-sampling can induce an additional computational task since the size of our fraud dataset is already large. We apply Random under-sampling in the experiments, as it aims to balance class distribution through the random replication of minor class samples. One major drawback is that this method can discard examples that could be important for the performance metric.

The main feature selection methods are filter, wrapper and embedded methods [2]. Filter methods examine each feature individually to determine the strength of correlation between the input feature and class labels. These methods do not rely on a learning algorithm and are computationally light. However, finding a suitable algorithm can be relatively difficult since filter methods do not take into account any classifier; ignoring the specific heuristics and biases of the classifier might lower the classification accuracy. Additionally, important features are more informative when combined with others that could be discarded and less important on their own. Wrapper methods use classifiers and evaluate the variable subset by the predicted performance. The main drawback is the number of computations required to obtain the feature subset can become computationally expensive. Embedded methods incorporate feature selection in the training process. Decision trees, such as CART, have a built-in mechanism to perform feature selection. The coefficients of regularized regression models are useful in selecting features. After the under-sampling, the size of the transaction dataset is relatively small. Therefore, we apply wrapper feature selection using a Random Forest classifier with 500 estimators.

4.3 ML model and explanation model

We select the ML model and the explanation model by their computation efficiency and model accuracy. The kernel-based explanation models, e.g. LIME and Kernel SHAP, must build a linear model for each instance; thus, they are unsuitable for our massive transaction dataset. Therefore, we apply Tree SHAP to obtain the explanations, as its fast C++ implementation is supported for eXtreme Gradient Boosting (XGBoost), LightGBM, CatBoost and scikit-learn tree model. We chose Tree SHAP XGBoost implementation which is the same as the original paper published by the author [8].

The XGBoost model is an implementation of the gradient boosting framework with the a highly efficient design. Boosting is an algorithm that creates multiple weak learners instead of one single powerful model. New models are added sequentially to correct the errors made by the existing models. At each iteration, the misclassified instances are assigned a higher weight, while the instances that were correctly predicted receive a lower weight. The process is repeated until no improvement can be made.

Gradient boost creates a new model based on the difference between the current prediction approximation and the target value and then adds them together to make a final prediction. This difference is usually called the residual or the residual vector. The name gradient boost refers to the gradient descent algorithm that trains on the residual vector to minimize the loss when adding new models.

4.4 Clustering

The clustering algorithm is mostly divided into the partitioning method and the hierarchical method. The partitioning method usually assumes a certain number of clusters in advance, while the hierarchical method does not have a priori assumption. The hierarchical method is again divided into two types: agglomerative and divisive. The agglomerative type uses a bottom-up approach, starting with individual data points. Then, the two most similar data points are merged into one cluster. On the contrary, the divisive type uses a top-down approach, treating all the data points as one big cluster and then divides it into small clusters.

We chose a hierarchical agglomerative algorithm since the number of clusters was unknown. The intuition behind the algorithm is to repeatedly merge the two most similar groups, as measured by linkage. We use Wards method as the linkage function, which is defined to be the squared Euclidean distance between points. There is little theory regarding how to find the correct number of clusters; it is even unclear what the correct number of clusters means. Hence, we visualize each merge step and decided the cut off distance in a hierarchical clustering dendrogram. The dendrogram visualization also helped us in capturing the structure of features.

4.5 Subgroup discovery

In the FPs elimination task, we try to filter out false alarm cases while retaining most true positives. This can be done by using descriptions to distinguish between the TPs and FPs groups. The main objective of the following three techniques is extracting descriptive knowledge from the data of a property of interest: contrast set mining, emerging pattern mining and subgroup discovery. Contrast set mining is a data mining technique for finding differences between contrasting groups. Emerging pattern mining is defined as item sets whose support increases significantly from one dataset to another. Subgroup discovery is aimed at finding descriptions of interest subgroups. The terminology, task definitions and rule-learning heuristics of those three mining techniques have been proven to be compatible [15]. Contrast set mining and emerging pattern mining communities have primarily addressed only categorical data, whereas the subgroup discovery community also considers numeric and relational data. Hence, we apply subgroup discovery techniques to discovery description for FPs.

Subgroup discovery is a data mining technique aimed at discovering interesting relationships between a set of instances with respect to a target variable. The patterns extracted are represented in the form of rules called subgroups. This is suitable for our framework since we aim to find small numbers of descriptive rules with high coverage on the FPs. The subgroup discovery can be divided into three categories: extensions of classification algorithms, extension of association rule and fuzzy evolutionary systems. We apply the extensions of classification algorithms CN2-SD, which is implemented in the open source tool Orange [17]. CN2-SD is obtained by adopting a rule-learning classifier CN2 to subgroup discovery. A standard CN2 tends to generate highly specific rules due to using an accuracy heuristic. CN2-SD uses weighted relative accuracy (WRAcc) to optimize a trade-off between rule accuracy and coverage.

WRAcc optimizes two factors of rule $X \Rightarrow Y$: rule coverage $P(X)$ and distribution unusualness $p(Y|X) - p(Y)$. Rule coverage is the size of the subgroup. The distribution unusualness is the difference between the proportion of positive examples in the describing rule and the entire example set. The algorithm starts by selecting the best rule with the highest WRAcc. In each iteration, after the best rule is selected, the weights of positive examples are decreased according to the number of rules covering each example $c(e)$, where $w(e) = 1/c(e)$. Let p denotes positive examples correctly classified as positive by rule, n denotes negative examples wrongly classified as positive by rule, and P and N are the numbers of all positive and negative examples in the dataset. $p' = \sum_{TP(X,Y)} w$ is the sum of the weights of all covered positive examples, and P' is the sum of the weights of all positive examples. WRAcc is computed as

$$WRAcc(X,Y) = \frac{p'+n}{P'+N} \cdot \left(\frac{p'}{p'+n} - \frac{P}{P+N} \right)$$

Since our objective is to discover the rule with higher coverage on global FPs, we measure the rule quality by global support and confidence. The support is defined as the proportion of total positive examples covered by the rule. The confidence is defined as the proportion of positive examples in all examples covered by the rule. The coverage of the rule can be illustrated in Figure 4.2. The rectangular represents the global alerts dataset divided into FPs and TPs. The ellipse represents the subgroups defined by rule X. Due to the classes imbalanced in the generated alerts, the discovered rules might have relative high FP confidence than TP confidence. However, the lower TP confidence does not assure the lower coverage of TPs. Imagine the ellipse in Figure 4.2 enlarges twice, then the rule still has the same FP and TP confidence, but both of FP and TP support are increased and might describe too many TPs. Therefore, lower TP support is preferred in the rule quality measurement for an imbalanced dataset.

The support and confidence can be computed as follows:

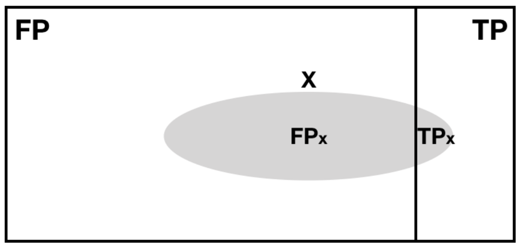


Figure 4.2: Quality measures of the rule [14].

$$Support(X \Rightarrow TP) = \frac{|TP \wedge X|}{|TP|} = \frac{|TP_X|}{|TP|}$$

$$Support(X \Rightarrow FP) = \frac{|FP \wedge X|}{|FP|} = \frac{|FP_X|}{|FP|}$$

$$confidence(X \Rightarrow TP) = \frac{|TP \wedge X|}{|X|} = \frac{|TP_X|}{|X|}$$

$$confidence(X \Rightarrow FP) = \frac{|FP \wedge X|}{|X|} = \frac{|FP_X|}{|X|}$$

4.6 Filter

Once the rules are extracted from subgroup discovery for each cluster, the FP confidence in the whole alerts population is computed. The rules are firstly selected by FP confidence higher than TP confidence. Then, the rules are selected by TP support threshold to restrict the amount of TPs. The selected rules are used to filter out the FPs.

4.7 Evaluation metrics

Area Under the Receiver Operating Characteristic (AUC-ROC) is the most common metric and recommended in handling class imbalance problems. It does not emphasize one class over the other, thus is not biased against the minority class. The ROC-curve is a plot on a two-dimension graph in which the y-axis is the TP rate, and the x-axis is the FP rate. TP rate denotes the correctly classified positive cases, and FP rate denotes the percentage of misclassified negative cases. The point $(0, 1)$ is the optimal classifier with perfect prediction. The diagonal dash line corresponds to a random classifier with the same TP rate and FP rate. The performance of the ML model is measured by

$$TPR_{ML} = \frac{TP_{ML}}{(TP_{ML} + FN_{ML})}$$

$$FPR_{ML} = \frac{FP_{ML}}{(TN_{ML} + FP_{ML})}$$

In our two approaches, the performance of ML model and filters are considered as a single model. The filters only perform on threshold 0.5, which corresponding to a single point on the ROC-curve. We denote the performance from ML model as TP_{ML} and FP_{ML}, and the remain TPs and FPs after filtering are TP_F and FP_F. The performance of the filter is measured by

$$TPR_F = \frac{TP_F}{(TP_{ML} + FN_{ML})}$$
$$FPR_F = \frac{FP_F}{(TN_{ML} + FP_{ML})}$$

Chapter 5

Experiments on explanation utility

In this chapter, we investigate the useful information that an explanation can provide to domain experts. In the first experiment, we try to define an accuracy threshold such that the local explanation is interpretable. In the second experiment, we identify the utility of the explanation in aggregating the local explanation into the group and global levels on a public dataset. Finally, we summarize the results and discuss the possible explanation utility for domain experts.

5.1 Experiment on model accuracy

In this experiment, we investigate the utility of explanation on different ML model accuracies. The first goal is to evaluate the interpretability of explanation by soundness of the explanation. The relation between model accuracy and the soundness of the explanation can be considered as useful information to domain experts. The explicit accuracy threshold for interpretable ML model may also be defined and assist domain experts in making a decision on the interpretable ML model. We compare the following six models which are: *a*) perfect *b*) moderately accurate *c*) random *d*) imbalanced *e*) high-dimensional and *f*) low-dimensional. The imbalanced, high-dimensional and low-dimensional models have the same setting as the perfect model but have different pre-processing strategies. The imbalanced model is trained on the imbalanced training set. The high-dimensional model is built on the dataset without feature selection and contains all the 506 features, whereas the low-dimensional model is built with only two selected features a and b.

5.1.1 Dataset description

To investigate high dimension and class imbalance problems, we create a binary synthetic dataset by using the following strategy: the dataset has six attributes related to the class and 500 irrelevant ones. (Table 5.1). The instances are classified as positive if all the following conditions are satisfied: $(a * b > 55) \wedge (c > 7) \wedge (d < 7)$. The dataset contains $10,000$ instances with an imbalance ratio of 30. The imbalance ratio is defined as the ratio of the number of instances in the majority class to the number of examples in the minority class.

Feature	**Description**	**Values**
a, b, c, d, 1-500	random variables	[1, 10]
a*b		[1, 100]
a/b		[0.1, 10]

Table 5.1: The description of the synthetic dataset.

5.1.2 Pre-processing

Random under-sampling and wrapper feature selection are applied to deal with class imbalance and high-dimensional data. The under-sampling ratio is set as the default, which results in equal numbers of majority and minority instances. Feature selection is performed by a Random Forests classifier with 100 estimators.

5.1.3 The ML model and the explanation model

The data are stratified split into the training and test sets; 80% of the data is used as the training set, and 20% is used as the testing set. To achieve diverse accuracies, we chose Support Vector Machines (SVMs) and Random Forests (RFs) in the experiment.

Both LIME and Kernel SHAP are model agnostic and can provide a local explanation for the SVMs and RFs models. However, it is difficult to judge the relation between the local explanation and the ML model accuracy from an explanation of a single instance, as it only reflects the local behaviour of the model, and the result is highly depended on the chosen instance. LIME does not guarantee a perfect distribution of the effects. Therefore, Kernel SHAP is the only suitable explanation model for the experiment. We group the local explanation to model the level and visualize the result.

5.2 Experiment results on model accuracy

In this section, we first show the AUC-ROC of each model and then use iml [10] to generate the force plot for visualizing the SHAP explanation result.

The force plot for a single instance is shown in Figure 5.1, where the features pushing the output probability higher are shown in red and the features pushing the output probability lower are shown in blue. The length of each colour block represents the SHAP value and the text below is corresponding feature value. The sum of the SHAP values equals the difference between the expected model output (averaged prediction of the training set) and the current model output.

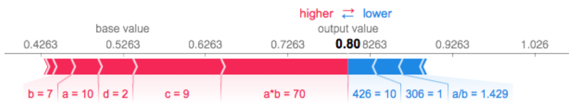
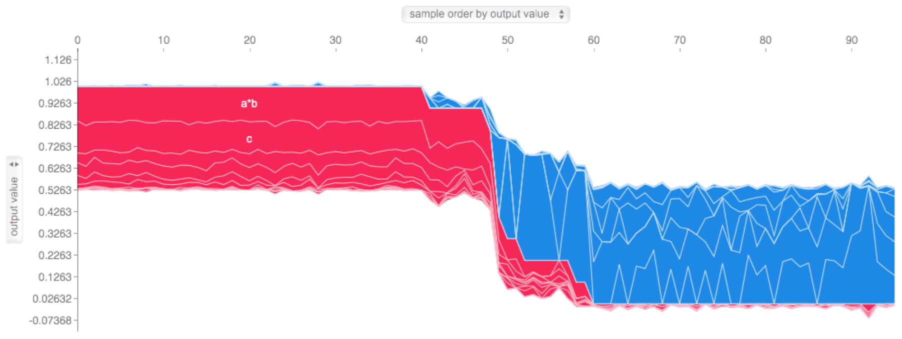


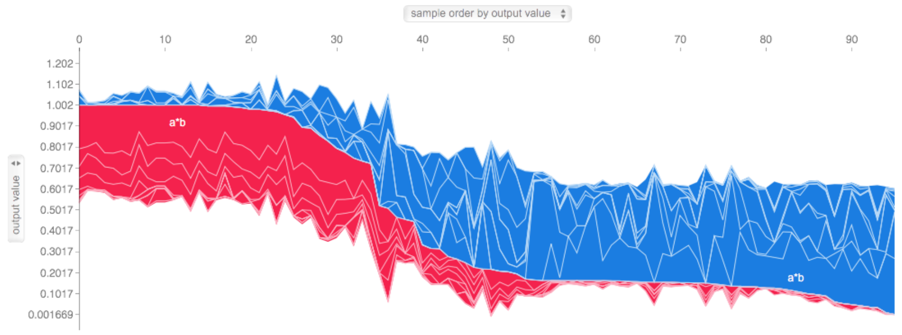
Figure 5.1: The SHAP explanation of a single instance.

To gain a balanced view of both classes, we sub-sample the instance in the negative class to the same number as the positive class in the force plot. The force plot rotates every single explanation 90 degrees in counter-clockwise direction and stacks all the explanations of the samples horizontally; the first half consists of the positive instances. The vertical length of the colour block separated by a white line indicates the importance of each feature. The middle point between two colours represents the prediction probability of the instance. The value of the original feature will be displayed when hovering over each sample.

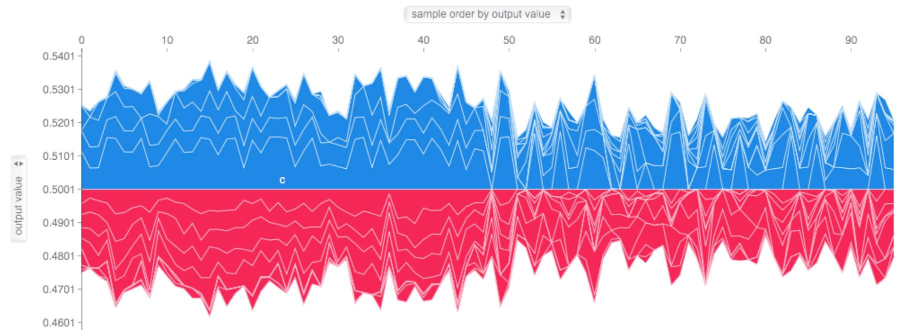
Table 5.2 and Figure 5.2 show the AUC-ROC of each model and the corresponding explanation. When the model has higher accuracy, the main decision of the model decision made by $a * b$ can be observed clearly. When the model has moderate accuracy, there are more instances that are close to the decision boundary and the model decision made by $a * b$ becomes less clear. When the model is close to random guessing, the model does not make the decision (the output probability is equal to 0.5), so the positive effects are equal to the negative effects.



(a) The SHAP explanation of the perfect model.



(b) The SHAP explanation of the moderately accurate model.



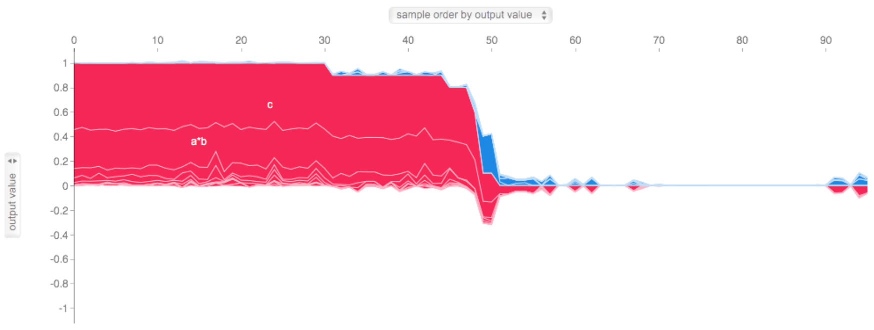
(c) The SHAP explanation of the random model.

Figure 5.2: The model-level explanation of the ML models with different AUC-ROC.

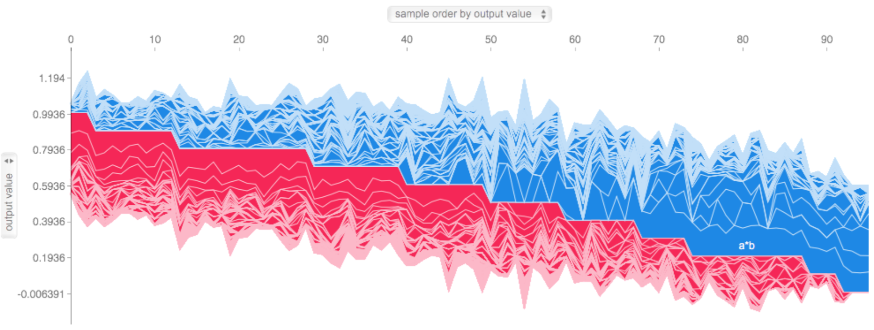
	Perfect	**Accurate**	**Random**	**Imbalance**	**High-dimensional**	**Low-dimensional**
AUC-ROC	1.000	0.789	0.510	1.000	0.955	0.929

Table 5.2: The AUC-ROC of the synthetic dataset.

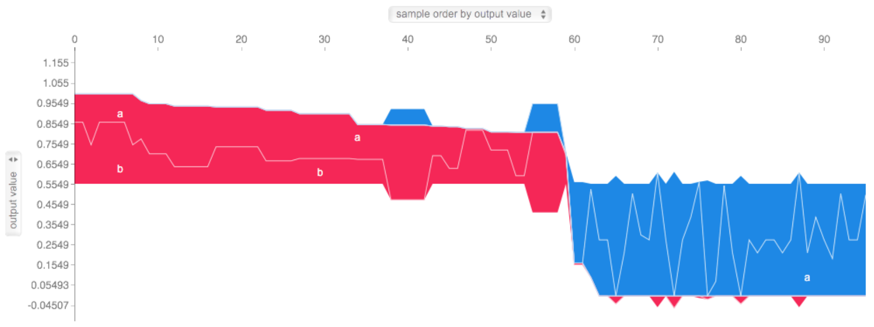
Figure 5.3 shows the models with different biases and the corresponding explanations. When the model is biased towards the negative class because of class imbalance, the model considers c more important than $a*b$ for the decision making and returns misleading information. The base value is moved from 0.5 to around 0, as it is the expected prediction from the training set. This increases the positive effects of positive predictions, as the SHAP values need to sum up to the output probability.



(a) The SHAP explanation of the imbalance model.

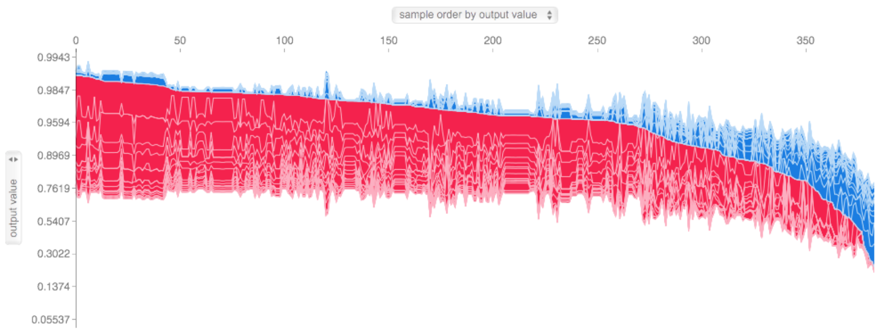


(b) The SHAP explanation of the high-dimensional model.



(c) The SHAP explanation of the low-dimensional model.

Figure 5.3: The model-level explanation of the models with different biased.



(a) The SHAP explanation of higher complexity model with AUC-ROC=0.996.

Figure 5.4: The model-level explanation of the models with higher complexity.

When the model is biased because of high-dimensional features (model contains both relevant and irrelevant features), the model output probability becomes less certain than the perfect model, as there are also many irrelevant features that contribute both positively and negatively to the prediction. When the model is biased because of low-dimensional features (model does not contain all the features of decision-making), the model makes a decision on features a and b because the decision of ground truth is made by $a * b$.

5.3 Conclusion of the experiment on model accuracy

From the above experiment results, we can observe that the soundness between each explanation decreases with the accuracy score in the simple models. The accuracy threshold of interpretable model seems possible to derive. However, when the model becomes more complex and biased, the threshold might also change, and the following questions emerge: How sound should the explanations be in order to be understandable to humans? If the accuracy scores are identical but model complexity arises (Figure 5.4), can humans still understand all the model decisions? When we encounter class imbalance and different dimensional data, can the accuracy threshold still hold? These questions make it challenging to define the ML model accuracy threshold on interpretability. Therefore, we conclude the experiment with an unknown threshold and move to the utility of different levels of explanations.

5.4 Experiment on explanation utility at different levels

The goal of this experiment is to determine the utility of the explanation for domain experts. The explanation derived from local explanation methods can be locally accurate. However, users need to have a specific instance that they are interested in. The model behaviour is also difficult to estimate from a local explanation. To help domain experts extract useful explanations, we show a process for understanding the model from aggregating the explanations to the model, group and local levels. The Rabobank transaction dataset is not intuitive in explaining informatively because of anonymous feature names and normalised feature values; therefore, we first conduct the experiment on the UCI bank marketing dataset. Then, we discuss the possible utility for domain experts.

To achieve this goal, we use the explanation models SHAP and LIME in order to identify explanations from the model, group and local levels. The data visualization tools Altair [1], iml [10] and Seaborn [11] are applied to combine the views from the original features, explanations and model predictions.

We consider the following sub-questions in the following experiment:

1. Which type of good explanations can be obtained from each level of explanation?
2. Can we change the output prediction with the knowledge extracted from the explanations?

5.4.1 Dataset description

The UCI bank marketing dataset consists of telemarketing phone calls to sell long-term deposits from May 2008 to November 2010. The data are related to the direct marketing campaigns of a Portuguese banking institution. In the campaign, a human agent makes phone calls to a list of clients to sell the deposits. If the clients call the contact centre for any other reason, they are asked to subscribe to the deposit. Thus, the result is a binary unsuccessful (“no“) or successful (“yes“) contact.

The original dataset contains 20 input features on four categories: bank client data, the last contact of the current campaign, other attributes and attributes of the social and economic contexts. The binary target label is if the client subscribed to a long-term deposit. The description of the variables is listed in Table 5.3. There are $41,188$ cases in total. The target labels are imbalanced because only 12.7% of the instances have “Yes“ responses.

Feature	**Description**	**Values**
bank client data		
age		[17, 98]
job	type of job	admin., blue-collar, entrepreneur, housemaid, management, retired, self-employed, services, student, technician, unemployed, unknown
marital	marital status	divorced, married, single, unknown
education	education level	basic.4y, basic.6y, basic.9y, high.school, illiterate, professional.course, university.degree, unknown
default	has credit in default?	no, yes, unknown
housing	has housing loan?	no, yes, unknown
loan	has personal loan?	no, yes, unknown
last contact of the current campaign		
contact	contact communication type	cellular, telephone
month	last contact month of year	jan, feb, mar, ..., nov, dec
day_of_week	last contact day of the week	mon, tue, wed, thu, fri
duration	last contact duration in seconds	[0, 4918]
other attributes		
campaign	number of contacts performed	[1, 56]
pdays	number of days passed by after previous campaign	[0, 999]
previous	number of contacts performed before this campaign and for this client	[0, 7]
poutcome	outcome of the previous campaign	failure, nonexisten, success
social and economic context		
emp.var.rate	employment variation rate	[-3.4, 1.4]
cons.price.idx	consumer price index	[92.201, 94.767]
cons.conf.idx	consumer confidence index	[-50.8, -26.9]
euribor3m	euribor three month rate	[0.634, 5.045]
nr.employed	number of employees	[4963.6, 5228.1]
output variable		
y	has the client subscribed a term deposit	yes, no

Table 5.3: The description of the UCI bank marketing dataset.

5.4.2 Pre-processing

The pre-processing steps for the bank marketing dataset are applied to deal with categorical attributes and unknown values. There are 10 categorical attributes, and 6 attributes contain unknown values. To prevent XGBoost from wrongly interpreting categorical features that have a numeric relationship, we simply convert each category into a dummy variable. There are several methods of handling the missing values, such as by deleting rows, replacing with the mean or median, and predicting the missing values by complete features. We do not discuss each method in detail because it is outside the scope of the study. The following are the pre-processing steps:

1. Convert the categorical attributes into dummy attributes except for "education".
2. Convert the "education" attribute to an integer by the education level in the range [1, 6].
3. Delete rows when the number of unknown values is less than 500. Predict unknown values with a Random Forests model when the number of unknown values is larger than 500.

The attribute "duration" is discarded because it is highly correlated to the output target. The author of the dataset suggests that it should only be included for benchmarking purposes. To focus more on general aspect of explanation, we select one social and economic context attribute, "euribor3m", which is the three-month Euribor rate computed by the European Central Bank (ECB); the rest of the 14 attributes serve as the features in this experiment.

5.4.3 The ML model and the explanation model

The data are stratified split into the training, validation and test sets; 60% is used as the training set, 20% is used for validating the parameters and 20% is used as the testing set for evaluating model performance.

To deal with class imbalance, we first apply Random under-sampling on the training set in order to balance two classes into an equal sample size. Then we build an XGBoost model on the training set and validate the parameters on the validation set. The AUC-ROC score on the test set is 0.803. Because of the characteristic of the explanation model, only the SHAP values are applied in global and group levels explanations. As the categorical attributes are converted into dummy attributes, the SHAP value of the categorical attributes is the sum of the SHAP values from all categories.

5.5 Experiment results on explanation utility at different levels

5.5.1 Explanation at the model level

We show that the model-level explanation can provide both general and abnormal explanations. In the scenario of selling the deposit, the first type of question might be "Why is the sell predicted as Yes?". We start by showing the importance of each feature. Figure 5.5 shows the feature importance computed by the mean of the absolute SHAP values. The most important features are "euribor3m", "month", "contact", "pdays" and "age". Then, we further explore how these important features make the outcome predicted as "Yes".

We first observe the global explanation of the four most important features in the model. Figure 5.6, 5.7 shows the scatter plot between the original feature value (x-axis) and its SHAP value (y-axis). The joint distribution of the two classes is shown on separate axes. This plot can be used to identify which feature values have a higher contribution to the target class and derive general explanations. For example, the value of "euribor3m" is lower than 2 has a higher probability of

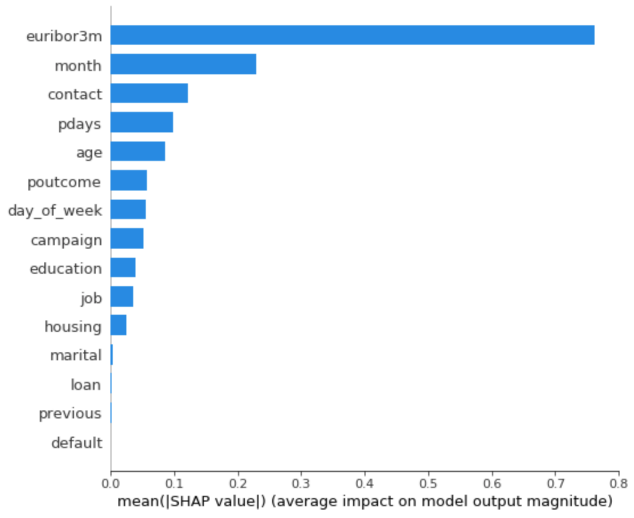


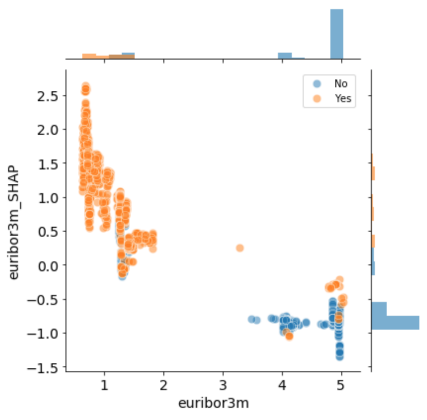
Figure 5.5: The feature importance for the XGBoost model.

being classified as “yes“. The generality can easily be measured by computing the support of the feature. We can measure generality by computing the support of the feature in each class, which is the number of instances covered by the rule in each class. The support in both classes are:

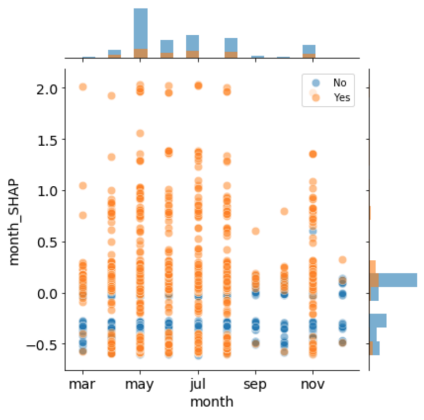
$$support((euribor3m < 2) \rightarrow \text{“}yes\text{“}) = \frac{|((euribor3m < 2) \bigcap \text{“}yes\text{“})|}{|\text{“}yes\text{“}|} = \frac{1720}{1747} = 0.98$$

$$support((euribor3m < 2) \rightarrow \text{“}no\text{“}) = \frac{|((euribor3m < 2) \bigcap \text{“}no\text{“})|}{|\text{“}no\text{“}|} = \frac{974}{6641} = 0.15$$

The support is relatively higher in the “yes“ class than in the “no“ class, so a lower euribor3m value has a higher probability of being a successful sell, which is a very general and simple explanation. This also explains why “euribor3m“ has such high feature importance —it separates most of the cases.

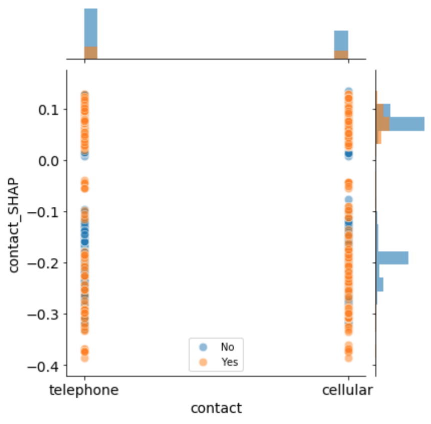


(a) The scatter plot of ”euribor3m”.

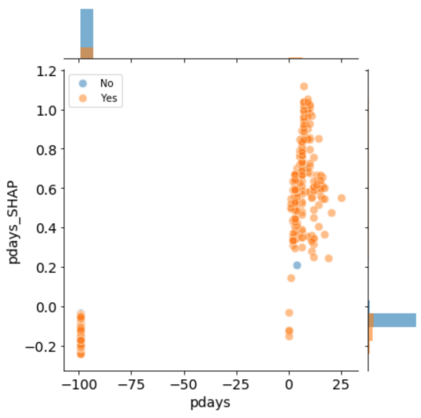


(b) The scatter plot of ”month”.

Figure 5.6: The scatter plot of ”euribor3m” and ”month”.



(a) The scatter plot of "contact".



(b) The scatter plot of "pdays".

Figure 5.7: The scatter plot of "contact" and "pdays".

In contrast to a general explanation, humans also prefer abnormal causes. Abnormal causes can be related to some features with infrequent values and have a strong influence on the prediction. The unusual values can be captured in the boxplot. Figure5.8 shows that “age“, “campaign“, “pdays“, “euribor3m“ and “month“ have some outliers with higher SHAP values. In particular, "campaign" does not have a high rank in the feature importance of the model and can easily be ignored.

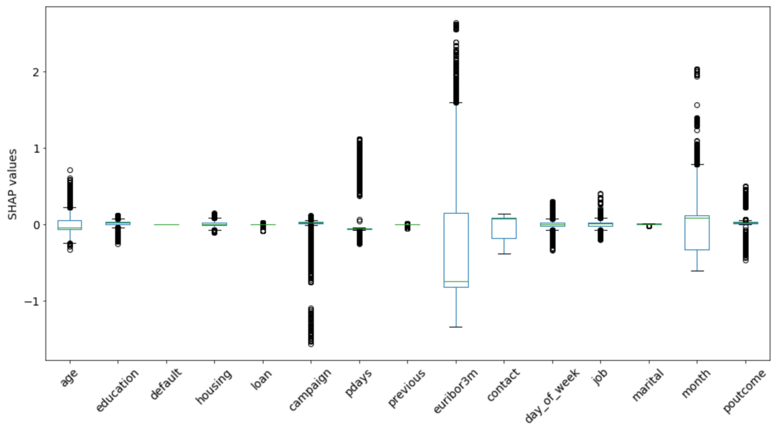
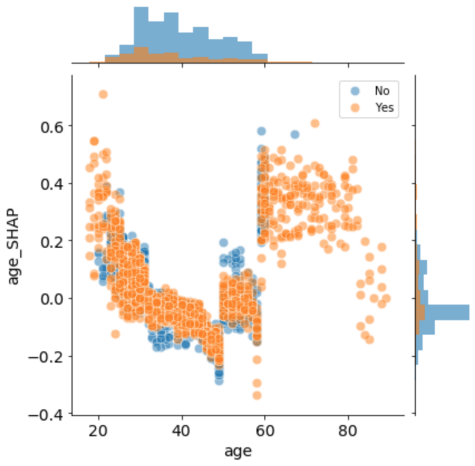


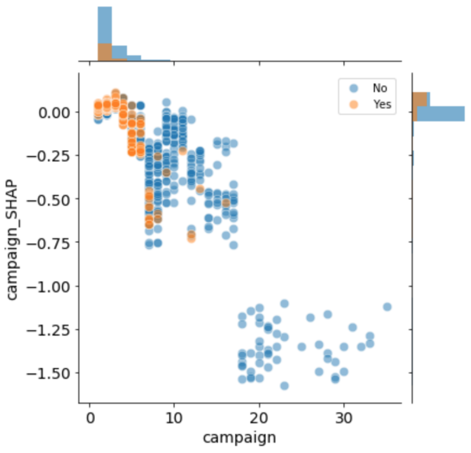
Figure 5.8: The boxplot of the SHAP values for the XGBoost model.

We again illustrate these features in the scatter plots (Figure 5.7). As shown in the figure, there is a small group of instances that have “pdays“ larger than 0, and most of the instances have a value -99 which means that there is no previous contact from the other campaigns. The value is the number of days that passed after the client was last contacted by the previous campaign. It shows that the previous contact has a high positive impact on the prediction, although such values rarely occur in the dataset. The same situation also happened in the two other features “age“ and “campaign“.

When the “age“ is over 60 or under 25, the feature has a higher positive impact on the prediction, whereas the age in between has some negative impact (Figure 5.9). In the feature “campaign“, the majority of positive cases occur when the feature value is less than 5. When“campaign“ is larger than 18, the feature has significant negative effects on the prediction. This is no surprise, as it is the number of contacts performed during this campaign and for this client. The aforementioned results show that the global explanation can also capture an abnormal value and derive a useful explanation.



(a) The scatter plot of “age“.

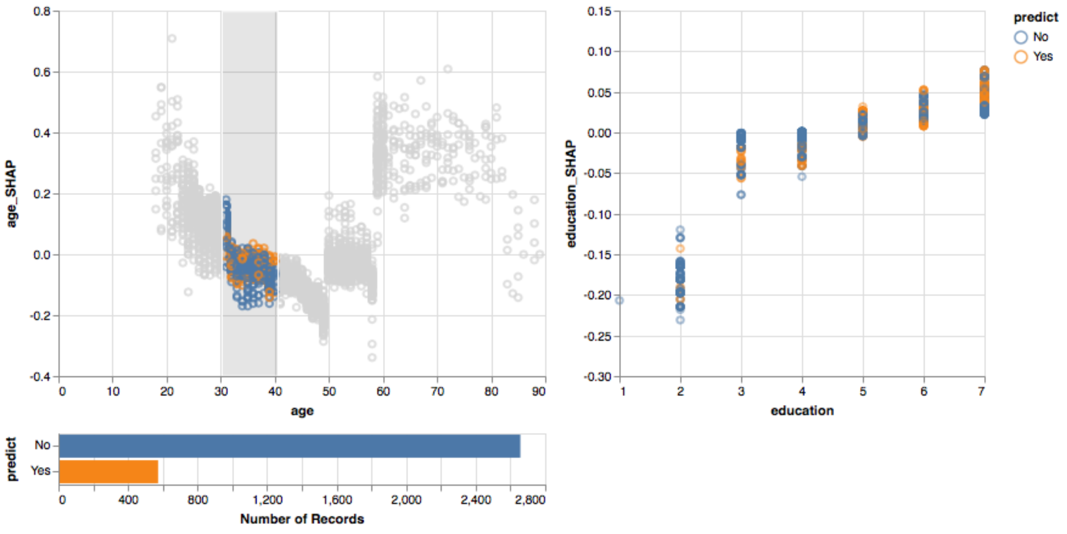


(b) The scatter plot of “campaign“.

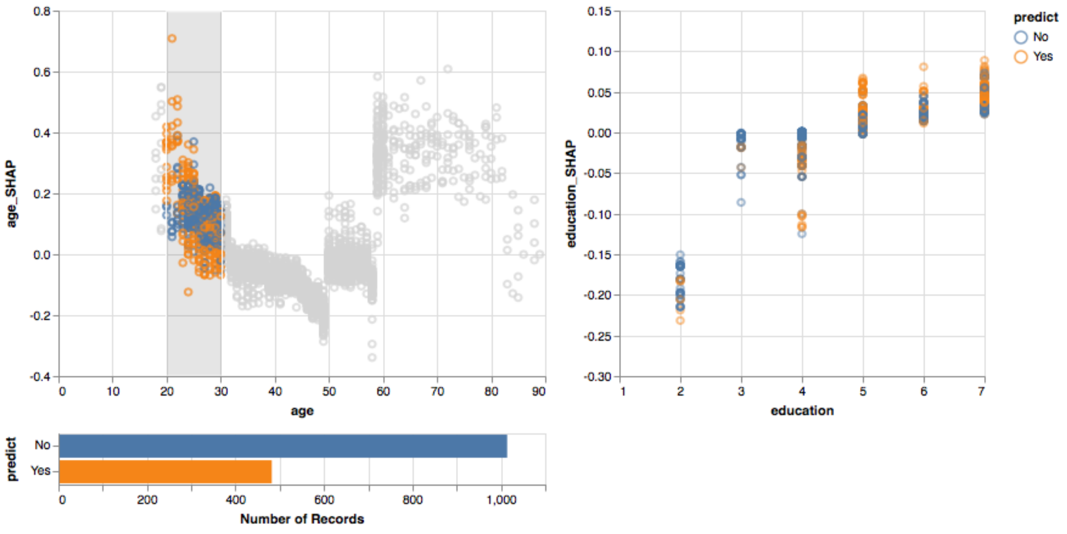
Figure 5.9: The scatter plot of “age“ and “campaign“.

5.5.2 Explanation at the group level

After gaining general insights from the model, we can further delve into some sub-groups. First, we show that group-level explanation can provide contrastive explanations between groups. For example, we need to provide a solution to increase the success probability of sales. Instead of one specific customer, we start by targeting a specific age group of customers. We may think “How would the prediction change if the client age is changed from range 30-40 to range 20-30?“. This question can be answered by using an interactive multi-facets scatter plot and histogram (Figure 5.10). We can select a different group of instances by moving the grey rectangular selector. After comparing the predicted distribution between two groups, we can immediately identify if the probability of predicting “yes“ is increased, and one of the explanations is the positive contribution by “age“. We can also focus on the relation between “age“ and other features that we are interested in, such as “education“, between two groups. The scatter plot of “education“ shows that an education level higher than 5 has a positive impact on the prediction. After comparing the scatter plot between the two groups, we find that the education level has a higher impact on customers’ age between 20 and 30 than the customers’ age between 30 and 40. The difference is not significant in this example; but the plot can be applied to identify different feature interactions.



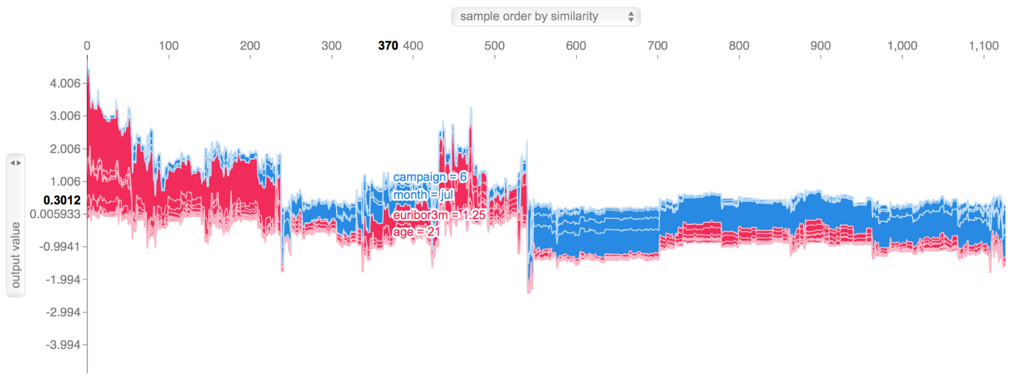
(a) The scatter plot between "age" and "education" for subgroup 20-30.



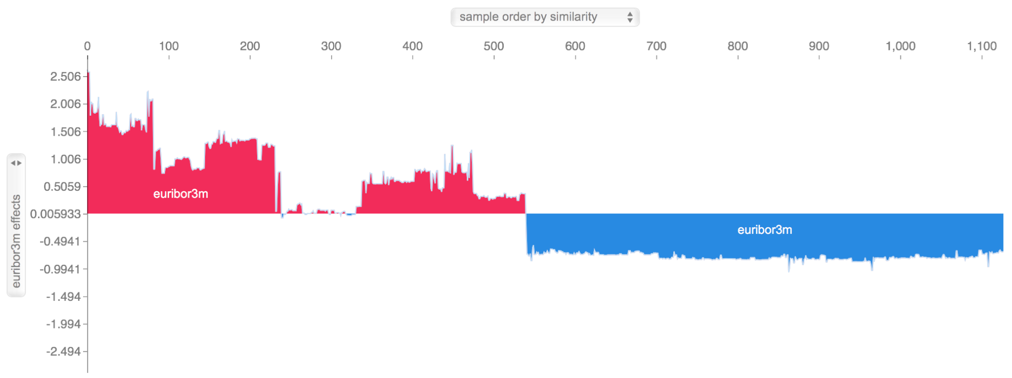
(b) The scatter plot between "age" and "education" for subgroup 30-40.

Figure 5.10: The interactive multi-facets scatter plot of feature "age" and "education".

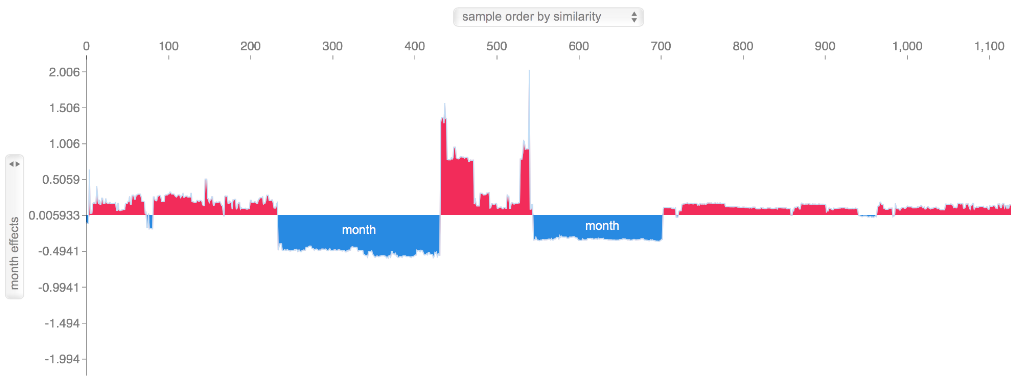
The group-level explanation can also provide contrastive explanations within the group. In the age group between 20 and 30, we can further ask, "Why are there some instances predicted as no, whereas others are predicted as yes?". The answer can be found in the full explanation within the subgroup. Figure 5.11 shows the explanation of all instances in the subgroup ordered by Hierarchical Clustering.



(a) The force plot of all the features in the subgroup.



(b) The force plot of "euribor3m" in the subgroup.



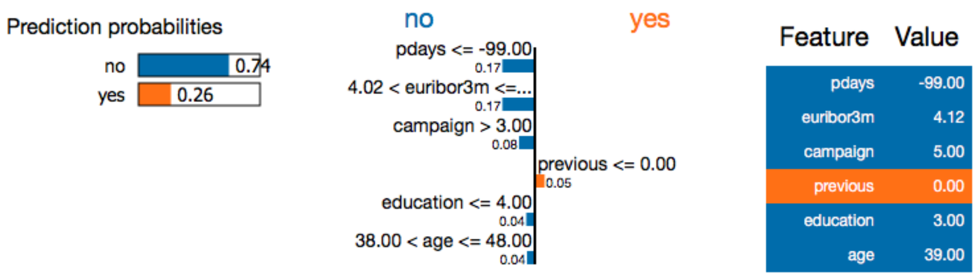
(c) The force plot of "month" in the subgroup.

Figure 5.11: The force plot of SHAP value in the subgroup age $20 - 30$.

We can see that there are roughly four clusters in the subgroup. The dominant features are "euribor3m" and "month". The first and fourth clusters are dominated by "euribor3m". The first cluster is predicted as "yes" because "$euribor3m < 2$", which is consistent with the previous global explanation. The second cluster is with "euribor3m", which is close to 2 and dominated by "month".

5.5.3 Explanation at the local level

After knowledge from the previous two levels is obtained, it is easier to derive a useful explanation at the local level. We show that LIME can provide contrastive and selected explanations at the local level. As mentioned in Chapter 3, LIME allows users to select the number of features to be perturbed and to predict the output probability for a different input value. We first target one customer who is predicted as “no“. Then, we modify the features to some counterfactual values in order to change the outcome. Figure 5.12 shows the explanation for a customer with age 39, education level 3 and has not been contacted by other campaigns before. The original output probability for the “yes“ label is 0.26. Basing on the previous knowledge from the global- and group-level explanations, we can boost the output probability by tuning the input features. Table 5.4 shows the change in input features and the corresponding output probability. After the input features are changed, the output probability is increased to 0.56.



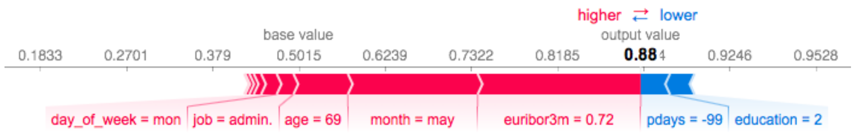
Feature	Value
pdays	-99.00
euribor3m	4.12
campaign	5.00
previous	0.00
education	3.00
age	39.00

Figure 5.12: The local explanation from LIME. The left figure is prediction probabilities of two classes. The middle figure is the LIME explanation of selected features. The representation of numerical features are discretized features. The right table shows the original feature values.

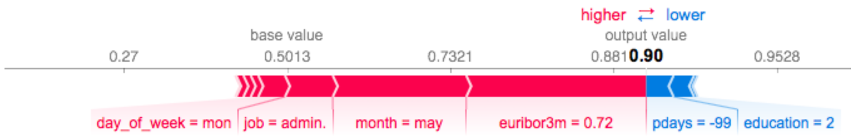
change of input feature	output probability
age = 20	0.39
age = 20 $\wedge$ pdays = 5	0.49
age = 20 $\wedge$ pdays = 5 $\wedge$ education = 7	0.56

Table 5.4: The relation between the change of input features and output probability

To show the reason why SHAP cannot provide a selected explanation, we present another example. Figure 5.13 shows that the output probability for the case is 0.88 and the SHAP value of “age“ is 0.31. When the explanation is wrongly interpreted, one might think that after the feature “age“ is removed, the output probability will decrease to 0.31. However, when we rebuild a model without the feature “age“, the output probability for the same instance is actually increased to 0.90. As discussed in Chapter 3, the SHAP value is the total contribution of the feature value to the difference in the actual prediction and the mean prediction. Therefore, when we rebuild a model, the actual prediction and the mean prediction might also change.



(a) The SHAP explanation with feature “age“.



(b) The SHAP explanation without feature “age“.

Figure 5.13: The local explanation from SHAP.

5.6 Conclusion of the experiment on explanation utility at different levels

We have shown the explanation utility from the model, group to local levels on a public dataset. To answer the sub-questions in this section, we summarize the properties of useful explanations obtained on each level in Table 5.5. At the local-level explanation, we also show that the output probability can be boosted by prior knowledge obtained from the explanations. The possible utility for domain experts when they deal with fraud detection will be discussed in Section 6.6.5.

	model	**group**	**local**
SHAP	general, abnormal	contrastive	contrastive
LIME	x	x	contrastive, counterfactual, selected

Table 5.5: The useful explanations on different explanation level

Chapter 6

Experiments on FPs elimination

We conduct an application-grounded evaluation experiment based on the Rabobank transaction dataset. The primary goal is to develop strategies to assist domain experts in fast processing alerts and reducing the FPs. We proposed two approaches for the automatic elimination of FPs. In the first approach, we build an additional ML model filter in which we substituted the SHAP values for the original values. We expect that some characteristics of FPs and TPs are reflected in the explanations. In the second approach, we clustered the SHAP values into several groups and observed the dominant features in each group. Then, we apply the subgroup discovery technique in each cluster to find subgroups that are as large as possible and have the most unusual distributional characteristics. Afterwards, the extracted rules that have a high concentration of FPs are used to filter out FPs.

6.1 Experiment setup

6.1.1 Dataset description

The Rabobank transaction dataset is composed of the transactions occurring from October 2016 to September 2017. The dataset contains 585 features, which including 394 categorical features, 174 numerical features, and 17 textual features. All the feature values are normalized between 0 and 1. The feature and label names are anonymous. The labels cover five classes with four types of fraud (Mo1, Mo2, Mo3, Mo9) and one class of non-fraud (Mo0). The class Mo9 fraud cover unknown fraud, and alerts for this class sometimes belongs to the other three types of fraud. In this thesis, we focus primarily on fraud types Mo1 and Mo2, since the majority of cases belongs to those two classes. This experiment is considered a multiclass classification problem since some of the features are shared among different types of fraud.

The dataset contains $1,796,979$ cases in total and equally partition into training, validation and test set. The distribution of fraud cases to training, validation and test set is 60%, 20% and 20% respectively. To deal with class imbalance, the training set contains higher percentage of fraud cases in comparison with the validation and test set. In addition, the sampling method also applied in the pre-processing to further balance the training set. The number alerts in each class are listed in Table 6.1.

6.1.2 Pre-processing

In order to deal with class imbalance and the large number of features, random under-sampling and a wrapper method of feature selection are applied. The under-sampling ratio(ratio between majority class and minority class) is set to 15. Wrapper feature selection is performed using a

Fraud type	**Mo0**	**Mo1**	**Mo2**	**Mo3**	**Mo9**
training	597,593	1,236	1,149	62	51
validation	597,573	420	379	20	18
test	597,645	409	384	18	22
Total	1,792,811	2,065	1,912	100	91

Table 6.1: Number of alerts in each class in the transaction dataset

random forests classifier with 100 estimators. We use 5-fold stratified cross-validation on the training set to select the number of features. The accuracy gradually decrease as more features are removed. To ensure adequate visualization for each feature, the number of features is limited to 20.

6.1.3 ML model and explanation model

A grid search is applied with 5-fold cross-validation to tune the hyperparameters for the XGBoost model. The objective function is chosen as ”multi:softprob” to train five classes of labels. The result contains the predicted probability of each alert belonging to a particular class. The prediction is the class with the maximum probability. As the classes are still imbalanced in the dataset after under-sampling, the samples of the minor classes are assigned different weights. The weight of the minor class is the ratio between the sum of the major class and sum of the corresponding minor class. The chosen parameters for the model are shown in Table 6.2. The training time of both ML model and explanation model are in the order of minutes. The AUC-ROC scores of each class and the confusion matrix are shown in Table 6.3 and Figure 6.1. Afterwards, the output predicted probability from the ML model is used as input for the Tree SHAP explanation model. The SHAP values can be considered as the feature importance of each class.

n_estimators	max_depth	min_child_weight	subsample	colsample_bytree	learning_rate
100	3	3	0.7	0.5	0.1

Table 6.2: The chosen parameters of the XGBoost model.

Fraud type	**Mo0**	**Mo1**	**Mo2**	**Mo3**	**Mo9**
AUC-ROC	0.992	0.984	0.996	0.998	0.924

Table 6.3: The AUC-ROC of each class in the XGBoost model.

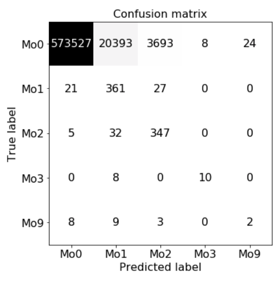


Figure 6.1: The confusion matrix of the XGBoost model.

6.1.4 ML model filter

The parameter settings used for the ML filters are the same as for the ML models. The ML filter is trained on the SHAP values of the output alerts from the ML model. In this experiment, we only consider the instances with output probability over 0.5 as alerts. Since the number of alerts and the SHAP values from each class are different, the ML filter is trained separately for each class. The features are the SHAP values of each type of fraud. When the prediction is consistent with the specific fraud type, it is labeled as TP; when the prediction wrongly classified a non-fraud (Mo0) class as a specific type of fraud, it is labeled as FP. We train the ML filter using the same training set that was used for the ML model and predict on the test set for fraud types Mo1 and Mo2 respectively. Once the ML filter predicts an alert is false, the alert is filtered out. The performance of the ML filter is computed based on the remaining alerts using the evaluation metric described in Section 4.7.

6.2 Experiment on FP elimination using ML filter

To evaluate the performance of FP elimination based on SHAP values, we compare the ML filter with two baselines: the performance of the original ML model and that of the ML filter built based on the original features. We expect that training the ML filter based on SHAP values results in better performance than just tuning the output probability threshold. Moreover, the performance of the ML filter build on a combination of the original features and SHAP values is also tested.

The following sub-questions are considered in the experiment:

1. What are the dominant features in each type of fraud?
2. Can we visualize the difference between SHAP values in FPs and TPs?
3. What is the performance of filtering FPs with the ML filter built on SHAP values?

6.3 Results on FP elimination using ML filter

6.3.1 Visualization on SHAP values

In order to understand the model's decision, feature importances of each type of fraud are illustrated in Figure 6.2. The feature importance is computed using the mean of absolute SHAP values for each featur. The length of each colour block represents the feature importance of the corresponding class. We can observe that the dominant features in Mo1 fraud, such as C_083, N_143, N_046, N_148, and N_045, are also the major features in Mo0. In contrast, Mo2 has fewer features that overlap those of Mo0, and the dominant features are N_000, N_159, and C_206. As a result, Mo1 performs the worst out of the four types of fraud and many alerts predicted as Mo1 are actually belong to Mo0 (Figure 6.1). In addition, Mo3 and Mo9 fraud also have many dominant features similar to those of Mo0.

In the multiclass classification model, each instance has a different prediction probability for each different class. From the probability of each class, the corresponding SHAP values can be derived. The SHAP values of all the alerts sorted by TPs and FPs can be seen in the heatmap (Figure 6.3, 6.4). The first column shows their corresponding labels.

In Mo1 fraud, we can observe that for alerts that are classified as TPs, C_224, N_000, and N_143 have relative higher positive effects when compared to other features. For those classified as FPs, the features with negative effects are increased. Features with less impact on the prediction might

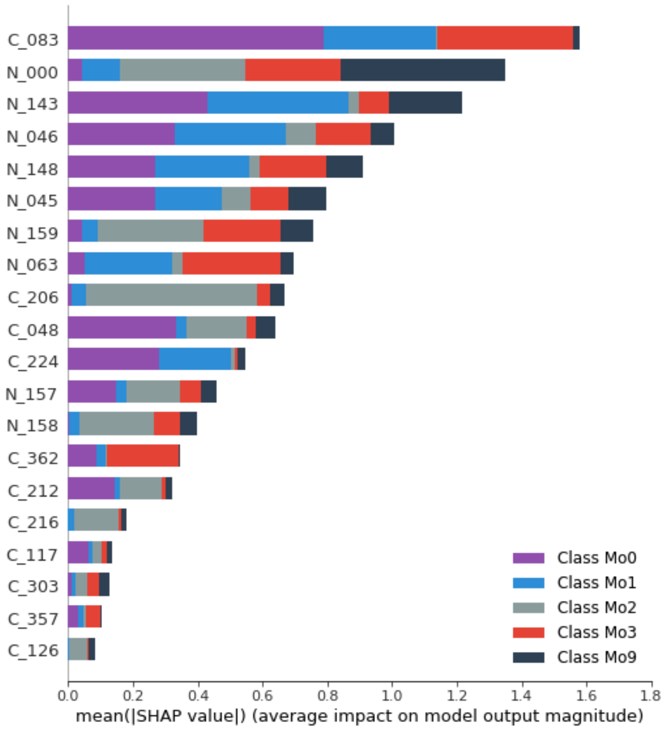


Figure 6.2: The feature importance of each type of fraud.

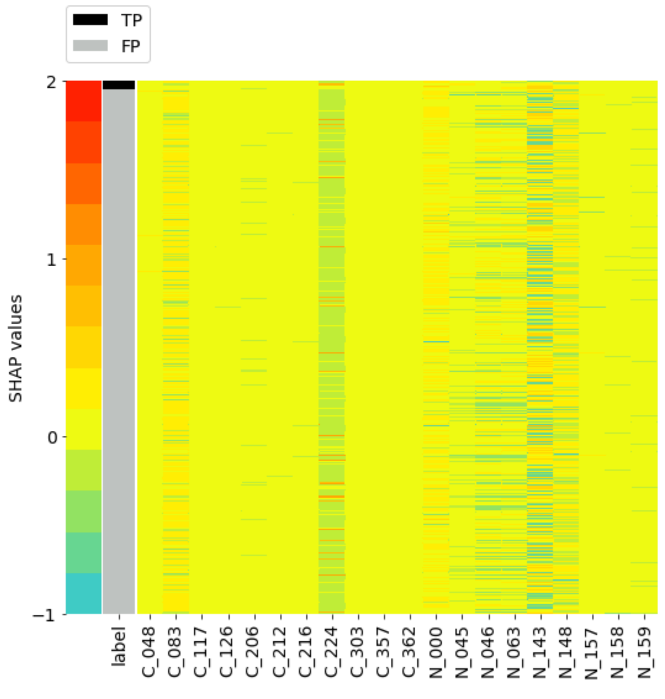


Figure 6.3: The heatmap of SHAP values of Mo1 fraud.

be considered as redundant features, for example, C_117, C_303, C_357 and C_362.

In Mo2 fraud, the difference of in the SHAP values between the two classes is more clear. Features C_206, C_216, N_000 and N_159 clearly had higher positive effects on alerts labeled as TPs (Figure 6.4). For FPs, most of those features have a negative impact on the prediction.

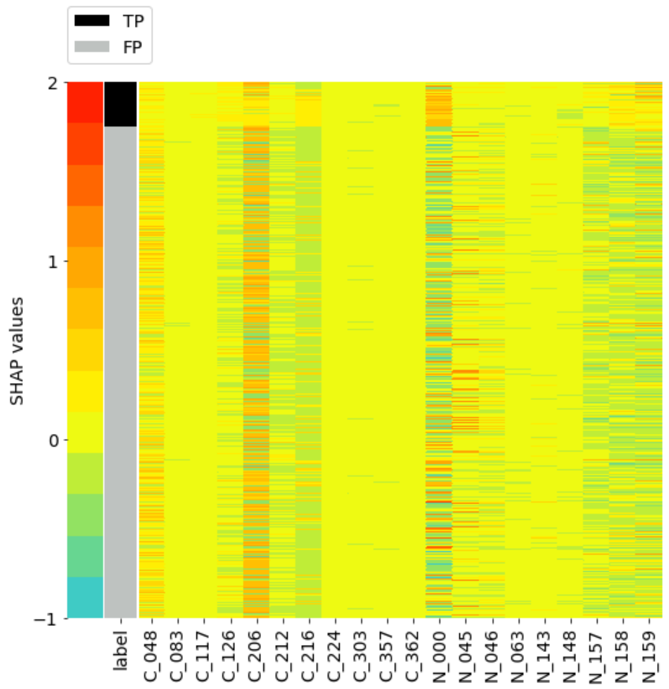
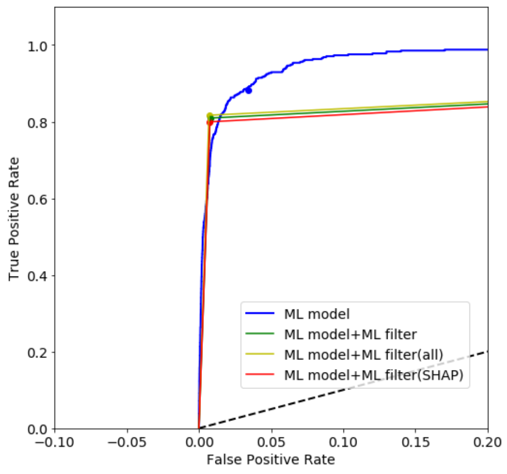


Figure 6.4: The heatmap of SHAP values of Mo2 fraud.

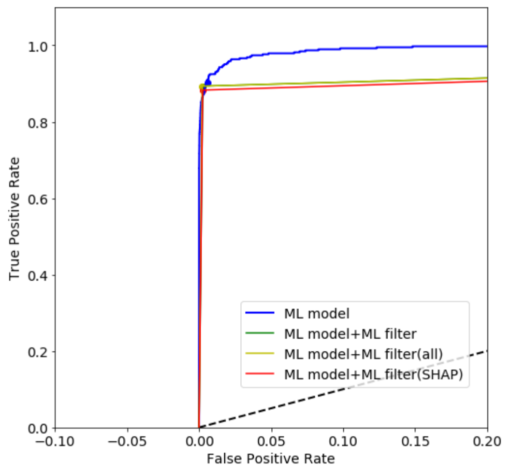
6.3.2 Evaluation in the ROC-plane

In this experiment, we only apply the ML filter on output probability threshold equal 0.5, which is the dot of corresponding model in the figure. Without the ML filter, the FPR can also be reduced by tuning the threshold of the ML model, which means we simply move the blue dot towards lower left of the ROC-curve. The idea behind is when the output probability threshold increase, the instances need to reach higher probability to be classified as alert. Hence, both TPR and FPR dropped with increased output probability threshold. We show that the ML filter can return better FPR while the TPR is the same as the ML model.

From the ROC-curve plot of Mo1 fraud (Figure 6.5a), we find the FPR of the ML filter with the SHAP value (red dot) is 0.0076 while the FPR of the ML model is 0.0132 with the same TPR. From the ROC-curve plot of Mo2 fraud (Figure 6.5b), we find the FPR of the ML filter with the SHAP value and the FPR of the ML model is 0.0026 and 0.0038 respectively. The ML filter trained on the original features performs slightly better than the proposed ML filter. The ML filter performs the best when both type of features are combined.



(a) The performance of ML filter for Mo1 fraud.



(b) The performance of ML filter for Mo2 fraud.

Figure 6.5: The performance of the original ML model and ML filters in the ROC-plane. The dots represent the performance of each model on the threshold 0.5.

6.3.3 Conclusion on FP elimination using ML filter

The first sub-question is addressed in Figure 6.2 which shows a bar chart indicating the feature importances for each type of fraud. Results indicate that the important features in fraud Mo1 and non-fraud (Mo0) cases are similar, making it harder for the ML model to distinguish these two cases compare to the other types of fraud. In contrast, fraud Mo2 has fewer important features in common with non-fraud cases, and fraud Mo2 shares some common features with Mo1.

The second sub-question is addressed in Section 6.3.1, where we illustrate the feature importances determined by the SHAP values. For both Mo1 and Mo2 fraud, some of the dominant features have different SHAP values for alerts labelled FPs and TPs. Especially for Mo2 fraud, the main dominant feature of FPs and TPs has a significantly different values. Based on the above results we suggest FPs and TPs can be distinguished by SHAP values.

The third sub-question is addressed in Section 6.3.2, where we compare the AUC-ROC performance on ML filter with baselines. The ML filter based on SHAP values in both Mo1 and Mo2 fraud have better performance than the ML model. The result indicates that the ML filter based on the SHAP values can efficiently detect and filter out FPs. Moreover, the ML filter trained on a combination of original features and SHAP values performs even better. This experiment demonstrate the future possibility of using SHAP values as extra features for domain experts.

6.4 Experiment on FPs elimination using rule-based filter

The goal of this experiment is to filter out FPs using the extracted rules from the original features. In order to associate SHAP values with the original features, clustering and subgroup discovery techniques are applied. In this experiment, we first perform clustering on SHAP values. In each cluster, a subgroup discovery technique is applied to extract the rules that have highest coverage. We then set a threshold to reduce the rules based on the support of TPs to ensure that most of TPs are retained. The remain rules are then used in a rule-based filter to eliminate FPs. We consider two baselines: one baseline where no clustering on SHAP values is performed, and another where no filtering is done at all. For the results, we first investigate the patterns of features after clustering on SHAP values. Then, we present the FPs rules derived from subgroup discovery.

The following sub-questions are considered in the experiment:

1. Can we visualize the rule structure based on the clustered SHAP values and their corresponding feature values?
2. Can discovered rules represent each cluster?
3. How well does a rule-based filter perform?

6.5 Subgroup discovery parameters

We use a grid search to select optimal parameters for subgroup discovery. The tuning parameters in CN2-SD are gamma, beam width and the minimum coverage of examples. In order to preserve the interpretability of the rules, the length of a rule is restricted to between 3 and 5. The cover value is set as the percentage of the whole that the individual clusters size represent or the total alert size. After the rules are extracted, we apply different TP support threshold to select the best rules. The parameters are listed in Table 6.4.

parameters	gamma	beam width	cover	TP support
	{0.6, 0.7, 0.8}	{10, 20, 30}	{0.1, 0.2}	{0.05, 0.1, 0.15}

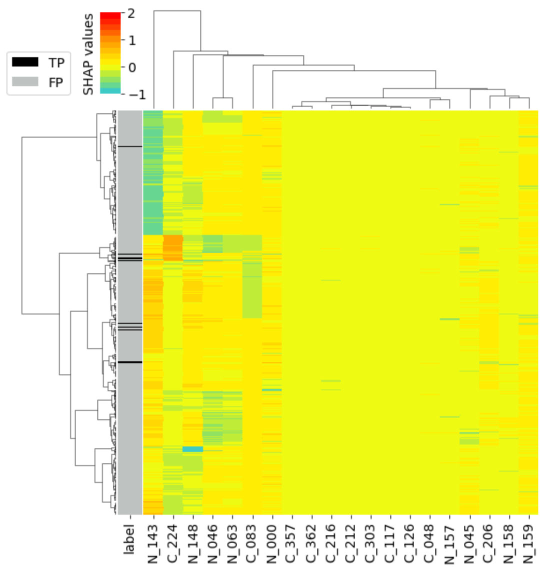
Table 6.4: The parameters used in Gridsearch for subgroup discovery.

6.6 Results of FPs elimination using a rule-based filter

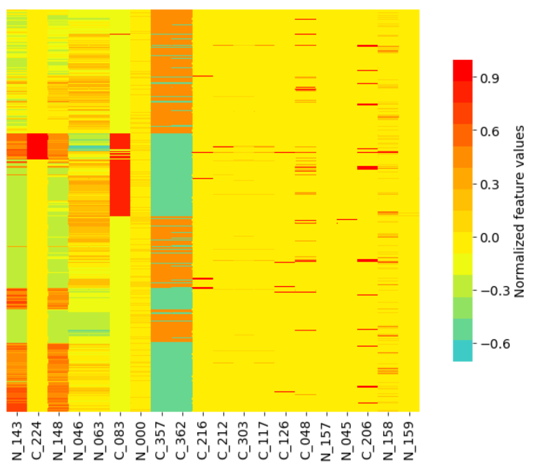
6.6.1 Visualization of clustering SHAP values

Hierarchical clustering is performed on the SHAP values to discover the dominant features in each cluster. After clustering, the structure of SHAP values can be observed clearly. As shown in Figure 6.6a, the most significant groups are dominated by C_083, C_224 and N_046. In order to understand the relationship between SHAP values and corresponding original features, the cluster groups made by SHAP values are mapped back to the original feature space (Figure 6.6b). The feature values are scaled by mean normalization for visualization purposes. Let x be an original feature value, $x' = (x - average(x))/(max(x) - min(x))$ is the normalized feature value. A comparison of the two heatmaps reveals a similar pattern in the original feature space.

From the clustermap of Mo2 fraud (Figure 6.7a), we observe that the distribution of SHAP values is more even than for Mo1 fraud. In other words, the prediction is affected by more distinctive features. Most of the clusters are leaded by C_048, C_206, N_000, N_045, and N_159. We also observe that some combination of features result in TPs, for example, C_048, C_206 and N_000. From both types of fraud, similar pattern to that of the clustermap are found in the original feature spaces. Therefore, we suggest subgroup discovery can be applied to reveal the rules for subgroups within each cluster.

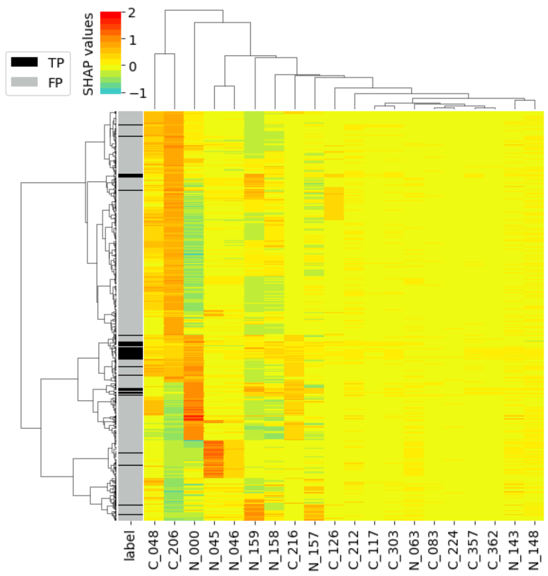


(a) The clustermap of SHAP values.

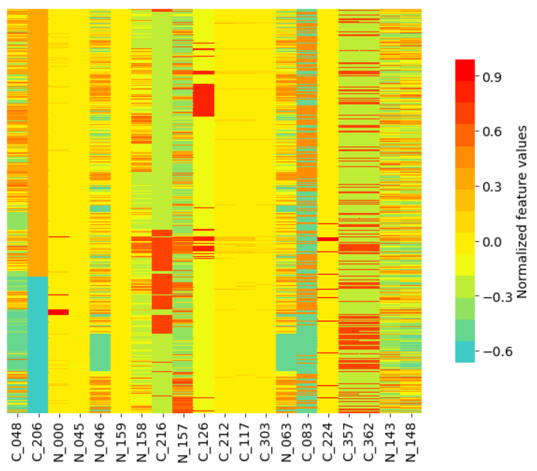


(b) The heatmap of normalized original features clustered by SHAP values.

Figure 6.6: The cluster heatmap for Mo1 fraud.



(a) The clustermap of SHAP values.



(b) The heatmap of normalized original features clustered by SHAP values.

Figure 6.7: The cluster heatmap for Mo2 fraud.

6.6.2 Rules discovered from each cluster

After Hierarchical clustering, fraud Mo1 and Mo2 are clustered into 6 and 7 groups respectively. The subgroup discovery is applied to discover the rules within each cluster. We present the rules with highest coverage of each cluster, which means the rule cover most of alerts within the cluster. Local FP support and TP support within each cluster are computed. Table 6.5 shows that most of rules have FP support higher than 0.5, meaning the rule represents at least 50% of FP cases in the cluster.

Rules	**FP support**	**TP support**	**Coverage**
$(C\_048 \leq 0.146) \wedge (C\_212 \leq 0.104) \wedge (N\_143 \geq 0.105) \wedge (N\_158 \leq 0.104)$	0.82	0.19	0.82
$(C\_083 \geq 0.102) \wedge (C\_303 \leq 0.102) \wedge (N\_157 \leq 0.114)$	0.62	0.25	0.59
$(N\_000 \geq 0.111) \wedge (N\_000 \leq 0.188) \wedge (N\_045 \leq 0.178) \wedge (N\_159 \leq 0.126)$	0.33	0.00	0.33
$(C\_048 \leq 0.102) \wedge (N\_046 \leq 0.166) \wedge (N\_143 \leq 0.154)$	0.66	0.19	0.64
$(C\_117 \leq 0.102) \wedge (N\_143 \leq 0.163) \wedge (N\_159 \leq 0.118)$	0.63	0.33	0.63
$(C\_117 \leq 0.100) \wedge (C\_303 \leq 0.102) \wedge (N\_148 \geq 0.159)$	0.42	0.20	0.42

Table 6.5: Rules discovered from each cluster in Mo1 fraud. The rule with the highest coverage in each cluster is listed.

6.6.3 Rule-based filter

The discovered rules from clusters and global alerts are shown in Table 6.6 and Table 6.7 respectively. The global support and confidence of each rule are computed to ensure the quality of the rules in all the alerts. The FP confidence of all the rules are higher than 0.99 and is therefore not included in the tables. In Mo1 fraud, the results show that some rules discovered both within clusters and globally have the same FP and TP support. However, the other rules discovered within clusters have lower FP support but equivalent TP support.

Rules	**FP support**	**TP support**
$(C\_303 \leq 0.102) \wedge (N\_046 \leq 0.159) \wedge (N\_143 \leq 0.154)$	0.40	0.08
$(C\_303 \leq 0.106) \wedge (C\_083 \geq 0.102) \wedge (N\_143 \leq 0.18)$	0.19	0.08
$(C\_083 \geq 0.102) \wedge (C\_303 \leq 0.102) \wedge (N\_157 \leq 0.114)$	0.16	0.08
$(C\_303 \leq 0.124) \wedge (N\_046 \leq 0.137) \wedge (N\_148 \leq 0.168)$	0.14	0.03

Table 6.6: Rules discovered from clusters in Mo1 fraud.

Rules	**FP support**	**TP support**
$(C\_303 \leq 0.104) \wedge (N\_046 \leq 0.159) \wedge (N\_143 \leq 0.154) \wedge (C\_117 \leq 0.102)$	0.40	0.08
$(C\_083 \geq 0.102) \wedge (N\_045 \geq 0.158) \wedge (N\_148 \leq 0.166)$	0.14	0.02

Table 6.7: Rules discovered globally in Mo1 fraud.

In fraud Mo2, the rules discovered within clusters also have lower FP support but not significantly lower TP support. Only one rule discovered globally has good support quality for both, with FP support of 0.74 and TP support of 0.08.

Rules	**FP support**	**TP support**
$(N\_045 \leq 0.173) \wedge (N\_143 \geq 0.101) \wedge (N\_157 \leq 0.112)$	0.50	0.05
$(C\_048 \leq 0.102) \wedge (N\_000 \geq 0.124)$	0.19	0.06

Table 6.8: Rules discovered from clusters in Mo2 fraud.

Rules	**FP support**	**TP support**
$(N\_157 \leq 0.119) \wedge (N\_000 \geq 0.101) \wedge (C\_303 \leq 0.2)$	0.74	0.08

Table 6.9: Rules discovered globally in Mo2 fraud.

To observe the rule coverage for each cluster, the cluster confidence of each rule is computed. As shown in Figure 6.8, each rule has the higher confidence within the specific cluster. Two rules have the highest confidence in Cluster 3. However, not all the cluster can find such a rule. The reason is again because the discovered rules exceed the threshold of TP support.

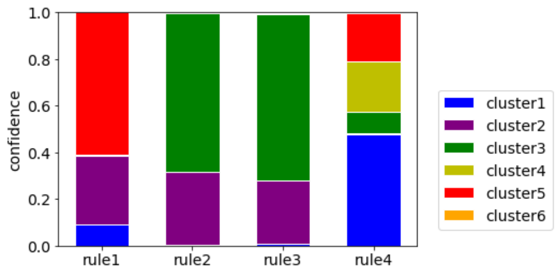
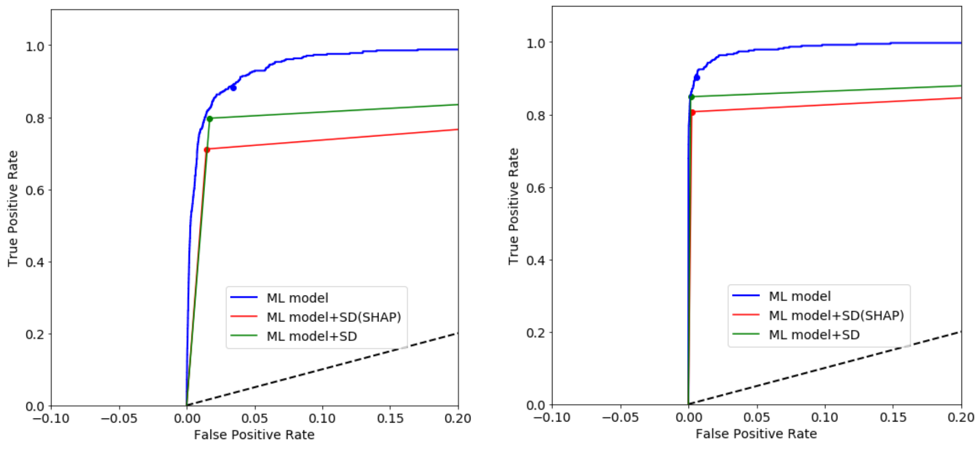


Figure 6.8: The cluster confidence of each rule.

6.6.4 Evaluation in the ROC-plane

From the ROC-curves (Figure 6.9), we can observe that the filter build on global rules perform at a similar level to the ML model. However, the proposed filter has a lower TPR, resulting in a worse performance.



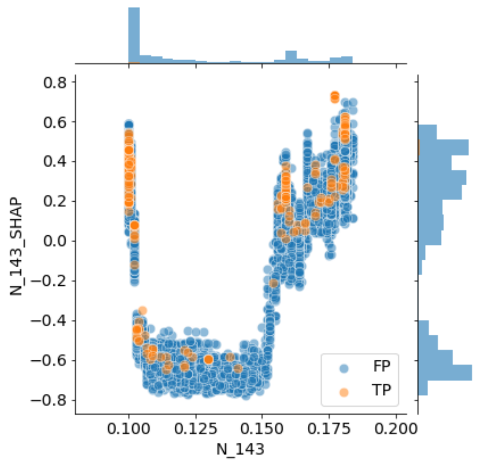
(a) The performance of filter for Mo1 fraud in ROC-plane.
(b) The performance of filter for Mo2 fraud in ROC-plane.

Figure 6.9: The performance of the original ML model and rule-based filters in the ROC-plane. The dots represent the performance each model on the threshold 0.5.

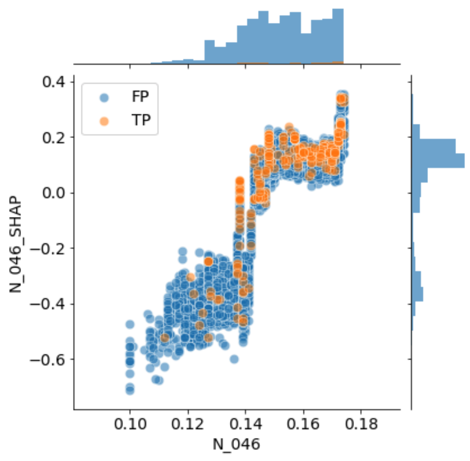
The results are consistent with the support of rules that proposed filter removing too many TPs. We observe that some rules in proposed filter (Table 6.6, 6.8) have less difference between FP support and TP support. If the threshold of the filter is well defined, the proposed filter might be made to perform as well as the filter build on global rules.

6.6.5 Explanation utility at different levels

We further discuss the possible utility of explanation at each level for domain experts. At the model level, the relation between the original feature values and the SHAP values of the important features can be visualized. For example, Figure 6.10a shows the important feature N_143 for all the alerts in fraud Mo1. We can observe that the model considers this feature to have a positive impact on the prediction of an alert when the feature value is 0.1 or higher than 0.155. In addition, the plot can also observe the feature that has an abnormal behaviour and a higher impact on the prediction, such as IP addresses and multiple failed login attempts. The abnormal behaviour can be observed in Figure 6.10b. When the feature value of N_046 is less than 0.13, the negative effect is higher than -0.2.



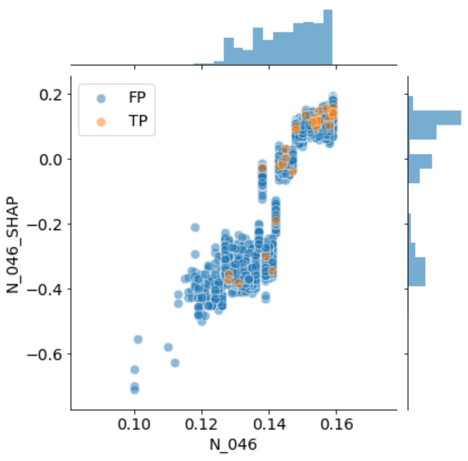
(a) The scatter plot of "N_143".



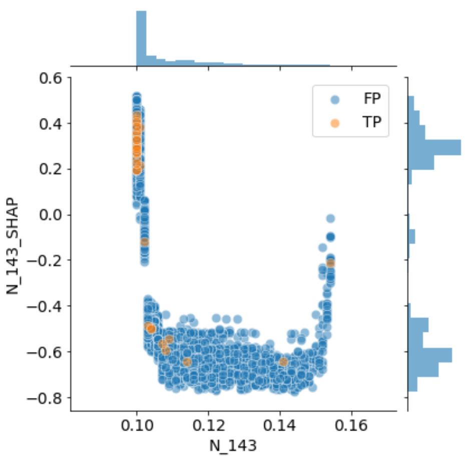
(b) The scatter plot of "N_046".

Figure 6.10: The scatter plot of the original feature value with its SHAP value and the joint distribution at model level.

At the group level, the visualization can be used to refine the rule identified from the FPs elimination framework. We plot the first rule from Table 6.6, which is $(N\_046 \leq 0.159) \wedge (C\_303 \leq 0.102) \wedge (N\_143 \leq 0.154)$. From Figure 6.11 we can observe that most of the TPs are discarded when $N\_046 \geq 0.15$ and $N\_143 = 0.10$. Then, the rule can be further refined to reduce the decrease in TPR.



(a) The scatter plot of "N_046" covered by the rule.



(b) The scatter plot of "N_143" covered by the rule.

Figure 6.11: The scatter plot of the original feature value with its SHAP value and the joint distribution at group level.

6.6.6 Conclusion on FPs elimination using rule-based filter

The first sub-question is addressed in Section 6.6.1. From the clustermap of SHAP values, we were able to visualize the relationship between SHAP values and the original feature values. When the clusters formed by the SHAP values were applied to the original feature space, similar structures were found in both spaces. The clustermap illustrates the most important features for each type of fraud and identifies some subsets of features which usually occur at the same time. These subsets of features represent different rules for each type of fraud. As a result, the clustermap could assist a domain expert in confirming consistency between the subsets of features and the attributes of rules in the rule-based system. Once the subsets of features are found to be meaningful, a domain expert can further check the discovered rules from each cluster.

The second sub-question is addressed in Section 6.6.2. We extracted the rule with highest coverage to represent each cluster. The result shows that the coverage of alerts by discovered rules varied from 33% to 82%. Moreover, some redundant features occurred in the rules and should be removed to allow more insight into the fraud. If the discovered rules were meaningful, the trust between the ML model and a domain expert could be established.

The third sub-question is addressed in Section 6.6.4. Although the extracted rules covered too many TPs, they could still be used to describe the behaviour of FPs. Due to the class imbalance between FPs and TPs, we might further consider support threshold based on the ratio between FP support and TP support. This consideration would ensure we do not discard too many TPs while filtering out FPs. For example, if we set threshold as a ratio of FP support to TP support larger than 4, only the first and fourth rules will be used in the rule-based filter in Table 6.6. The performance of the filter could therefore be improved. However, the problem of how to define the ratio in each type of fraud still remains. Moreover, some redundant features can be observed in the clustermap of the SHAP values. Those features could be removed during the rule discovery process, making the discovered rules more precise.

Chapter 7

Conclusions

In this thesis, we evaluated the quality of explanations for determining the utility of different approaches and assisting domain experts in eliminating FPs. The first goal was to define an accuracy threshold such that the local explanation would be interpretable. The second goal was to determine the utility of the explanation for aggregating local explanations into groups and at global levels. The third goal was to apply explanations to assist domain experts in automatically eliminating FPs while retaining most TPs. In the first part of this chapter, we present the research questions and summarize the achieved goals. We then discuss limitations and suggestions for future work.

7.1 Research questions

The research questions are as follows:

1. Will interpretability drop as the reference model become less accurate / biased?

In the experiment discussed in Section 5.1, we compared the explanation of six models with different accuracy and biases. We observed that the soundness of explanations decreases with the accuracy score in the simple models. However, the behaviour of the explanations became diverse when we encountered different model complexities, class distributions and dimensions of features. These results made it challenging to relate ML model accuracy with soundness and define the ML model accuracy threshold regarding interpretability. Hence, we concluded the experiment with an unknown threshold and move to the utility of aggregating explanations at different levels.

2. Which type of useful information can be obtained at the model-, group- or local-level explanations?

In the experiment discussed in Section 5.4, we determined that explanations that are good to humans are useful. Hence, we started by defining the types of good explanations. We then investigated the utility of two state-of-art model-agnostic explanation models SHAP and LIME. Based on their strengths and weakness, we aggregated the explanations on different levels and applied visualization techniques to derive useful explanations. We also demonstrated the possible utility for domain experts. The properties that can be fulfilled at each level of explanation are summarized in Table 5.5.

3. Can the explanation help a domain expert to post process fraud alerts?

In the experiment discussed in Section 6.2, we proposed two approaches for eliminating FPs automatically. In the first approach, we built an additional ML model filter and substituted the explanation feature values for the models original values. The performance of the ML filter for both Mo1 and Mo2 types of fraud was better than that of the ML model. With the same TPR rate, the FPR of the ML filter was 40% less and 32% less than the ML model in Mo1 and Mo2 respectively. Moreover, the ML filter trained on the combination of original features and SHAP values performed even better. This experiment demonstrated the future potential for using SHAP values as extra features for the domain experts. Hence, the goal was achieved, and SHAP values are recommended for future in automatic post-processing of fraud alerts.

4. Can the explanations indicate the structure of different fraud detector rules?

We addressed this question with the second approach in Section 6.4. This approach used visualization methods to illustrate clustered SHAP values and then extracted rules using subgroup discovery for each cluster. The extracted rules were subsequently used to filter out FPs. The visualization illustrated the structure of the dominant features in each cluster. Similar patterns occurred in both SHAP values and in the original features. The extracted rules from each cluster were found to represent the general rule of some clusters. Those rules, therefore, can help domain experts establish the reliability of the ML system. However, most of the rules still affected too many TPs. As a result, the rule-based filter did not outperform any baseline. We further discussed two possible future solutions including changing the threshold and removing redundant features.

7.2 Limitations and future work

The future tasks with respect to the explanation methods:

- The local explanation methods LIME and SHAP only ensure the explanation is local faithful, which means when the ML model is highly non-linear the explanation may not be faithful. We can estimate the faithfulness of the explanation and present the information to domain experts in the future.
- Tree SHAP explanation model is applied in the framework for the sake of computational efficiency. Therefore, the framework is not model-agnostic. In addition, we only tested the explanation using an XGBoost model. There are several alternative tree models which could also be applied, for example, LightGBM, CatBoost, and scikit-learn tree models. The Kernel SHAP could also be considered for use with other ML models.

The future tasks with respect to the FPs elimination framework:

- The framework was based on one statistic ML model. This offline setting might be too optimistic and unable to capture the characteristics in a real online system. In addition, the false alerts generated by the ML model might be different from those of the rule-based system.
- The ML filters only performed on output probability threshold equal 0.5. Future experiment could perform the ML filters on different thresholds to obtain more conclusive results.
- The baselines only considered model accuracy for FPs elimination. Future experiment could perform a real or simplified task to measure the improvement in time efficiency.
- There are many techniques involved in the framework. Due to the limitation of time, not all parameters were well-tuned. Parameter settings can have a strong impact on the results. For example, an under-sampling ratio may affect the distribution of data, the cut-off length in clustering and the rule coverage in subgroup discovery.

Bibliography

[1] Altair Developers. Altair: Declarative visualization in python. https://altair-viz.github.io, Last accessed on 2018-08-29.

[2] Chandrashekar, G., and Sahin, F. A survey on feature selection methods. *Computers & Electrical Engineering 40*, 1 (2014), 16–28.

[3] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research 16* (2002), 321–357.

[4] Doshi-Velez, F., and Kim, B. Towards a rigorous science of interpretable machine learning.

[5] Kotsiantis, S., Kanellopoulos, D., Pintelas, P., et al. Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering 30*, 1 (2006), 25–36.

[6] Kulesza, T., Burnett, M., Wong, W.-K., and Stumpf, S. Principles of explanatory debugging to personalize interactive machine learning. In *Proceedings of the 20th international conference on intelligent user interfaces* (2015), ACM, pp. 126–137.

[7] Lipton, Z. C. The mythos of model interpretability. *Queue 16*, 3 (2018), 30.

[8] Lundberg, S. M., Erion, G. G., and Lee, S.-I. Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888* (2018).

[9] Lundberg, S. M., and Lee, S.-I. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems* (2017), pp. 4768–4777.

[10] Lundberg, Scott M. Interpretable ml package designed to explain any machine learning model. https://github.com/interpretable-ml/iml, Last accessed on 2018-08-29.

[11] Michael Waskom. seaborn: statistical data visualization. https://seaborn.pydata.org/, Last accessed on 2018-08-29.

[12] Miller, T. Explanation in artificial intelligence: insights from the social sciences. *arXiv preprint arXiv:1706.07269* (2017).

[13] Molnar, C. A guide for making black box models explainable. https://christophm.github.io/interpretable-ml-book/,, 2018.

[14] Novak, P. K., Lavrač, N., Gamberger, D., and Krstačić, A. Csm-sd: Methodology for contrast set mining through subgroup discovery. *Journal of Biomedical Informatics 42*, 1 (2009), 113–122.

[15] Novak, P. K., Lavrač, N., and Webb, G. I. Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. *Journal of Machine Learning Research 10*, Feb (2009), 377–403.

[16] Ribeiro, M. T., Singh, S., and Guestrin, C. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016), ACM, pp. 1135–1144.

[17] University of Ljubljana. Orange-data mining fruitful and fun. https://orange.biolab.si/, Last accessed on 2018-08-29.

[18] Wei, W., Li, J., Cao, L., Ou, Y., and Chen, J. Effective detection of sophisticated online banking fraud on extremely imbalanced data. *World Wide Web 16*, 4 (2013), 449–475.