Niklas Hopfgartner, BSc

Fraud Detection in Online Transactions Using Sequential User Information

MASTER'S THESIS

to achieve the university degree of

Diplom-IngenieurIn

Master's degree programme: Computer Science

submitted to

Graz University of Technology

Supervisor

Assoc.Prof. Dipl.-Ing. Dr.techn. Denis Helic

Institute of Interactive Systems and Data Science
Head: Univ.-Prof. Dipl.-Inf. Dr. Stefanie Lindstaedt

Graz, February 2019

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date

Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

Datum

Unterschrift

In Dank an meine Eltern, Rupert und Brigitte,
für die endlose Liebe und Unterstützung auf meinem Wege.

Abstract

Automated fraud detection in online transactions can be considered as an essential part of a financial company to provide great user experience, save work and money. In this thesis, we analysed the behaviour of fraudulent users on a currency exchange platform (VirWoX) using sequential user information. We reviewed five different machine learning models to learn hidden patterns within the provided customer data. Further, we explained all the necessary steps to prepare and clean the data, as well as how to train and evaluate the different models. The results showed that it is possible to separate fraudulent from regular users. Therefore, we also suggested how the proposed fraud detection system could be operationalized and what actions should be set to prevent fraud.

Contents

List of Figures

List of Tables

1 Introduction

In recent years the blockchain and especially cryptocurrencies were in the raise and gained a lot of attention. Bitcoin is probably the most famous example of a cryptocurrency. The reason for its popularity was not only the fast increase of its exchange rate but also its capability to send and receive money anonymously. This lead to the effect that cybercriminals began to use Bitcoin for money laundering. For example, the so-called ransomware which got famous in the last years in the cybercrime scene used Bitcoin to receive the blackmailed money from the victims anonymously, because once ransomware infects a computer, it encrypts all personal files on the hard drive and demands ransom money in bitcoins for the decryption of the documents. As a consequence, one needs to register on a website which offers an exchange service for cryptocurrencies to buy bitcoins. Such exchange services are not only used by victims of ransomware and Bitcoin traders, but also by fraudsters. An example of such an exchange service is Virtual World Services (VirWoX[1]). On this platform, users can deposit money from various sources like PayPal, credit card, PaySafeCard or per bank transfer. This money can then be exchanged into other currencies and withdrawn with Bitcoin. This lead to the effect that cybercriminals abuse this service to withdraw money from hacked PayPal accounts or stolen credit cards and exchange it into bitcoins. Since Bitcoin transactions are irreversible and anonymous, the stolen money cannot be reclaimed. Therefore, Virtual World Services implemented several counter-measures to prevent fraudsters from abusing their service and reduce their financial loss. Since cybercriminals continually adapt their strategies to bypass such counter-measures, the goal of this thesis is to develop an additional data-driven fraud detection system.

This chapter gives an overview of the typical use cases of regular and fraud users on VirWoX and their existing counter-measures to reduce the loss produced by those cybercriminals. Furthermore, we define the requirements for the proposed fraud detection system followed by the limitations which we had. Finally, we end this chapter with an outline of this thesis.

[1] `https://www.virwox.com`

Limits in EUR	Level 0	Level 1	Level 2	Level 3	Level 4
active	immediately	after 10 days	after 30 days	after 60 days	on request
per 24 hours	90	120	200	400	1,000
per 30 days	270	900	2,000	5,000	15,000

Table 1.1: **Deposit limits for PayPal and Skrill.** After registration users start with the limit level of 0, where they can deposit a maximum of 90 euros per day and a maximum of 270 euros per month. Ten days after the first deposit users get into the next level, where they can deposit 120 euros per day and 900 euros per month. To get to the highest limit level of 4, users need to contact the customer support and request the activation of this limit.

1.1 Background and Motivation

Before we define the requirements of the fraud detection system, we need to understand what the typical behaviour of a fraudulent and a regular user is. Following this, we investigated the current counter-measures to block and limit fraudulent behaviour and reviewed the available data to train and test our system.

Figure 1.1 describes the behaviour of a typical user and Figure 1.2 illustrates the behaviour of a fraudulent user. One can see, that the sequence of performed actions differs between these two groups and therefore an algorithm should be able to separate them. Based on the behaviour of fraudsters, VirWoX implemented three counter-measures to block or at least limit this kind of behaviour. First, VirWoX introduced deposit limits, as listed in Table 1.1, for most payment methods, as listed in Table 1.2. Those limits do not prevent fraudsters from abusing VirWoX, but it reduces the resulting loss. Second, a rule-based system detects if one person uses multiple accounts by fingerprinting the web browser and tracking the IP address of the user. The intuition behind this approach is, that once fraudsters get blocked, they immediately create new accounts to continue with their fraudulent actions. Third, there is a delay of 48 hours for the payout of the first Bitcoin withdrawal. This delay enables VirWoX to stop the withdrawal process if the victim, PayPal or a credit card institute report malicious transactions within this period.

During the last few years, not only the existing system identified many fraudulent users, but also the credit card operators, PayPal and the victims reported fraudulent activities to VirWoX. This data provided the basis for this thesis to develop an extension to the current fraud detection system to reduce the number of incidents and the resulting loss. Further, VirWoX provided us with a collection of Apache logs of one year and the corresponding customer data. This data can be used to reconstruct the browsing behaviour of each user on the website. Table 1.3 describes the provided customer data,

Payment Method	Pay-In	Pay-Out	Deposit Limit
Bitcoin	yes	yes	no
PayPal	yes	yes	yes
Bank transfer	yes	yes	yes
Paysafecard	yes	no	yes
Skrill	yes	yes	yes
OKPAY	yes	no	yes
My Virtual Community Pay Terminal	yes	no	no

Table 1.2: **Available payment methods to deposit and withdraw money.** Depending on the chosen method different deposit limits apply as listed in Table 1.1.

Data	Description
Customer dump	A list of registered users including the username and the status of the user.
Login events	A list of all login events of all customers including the timestamp, IP address and the customer ID.
Incidents	A list of incidents including timestamps, customer IDs, loss and a description produced by the current fraud detection system and reported through PayPal and other payment providers.

Table 1.3: **Customer data provided by VirWoX.**

and Figure 1.3 explains the structure of an Apache log file. The next section describes the requirements for the proposed fraud detection system and defines the research questions.

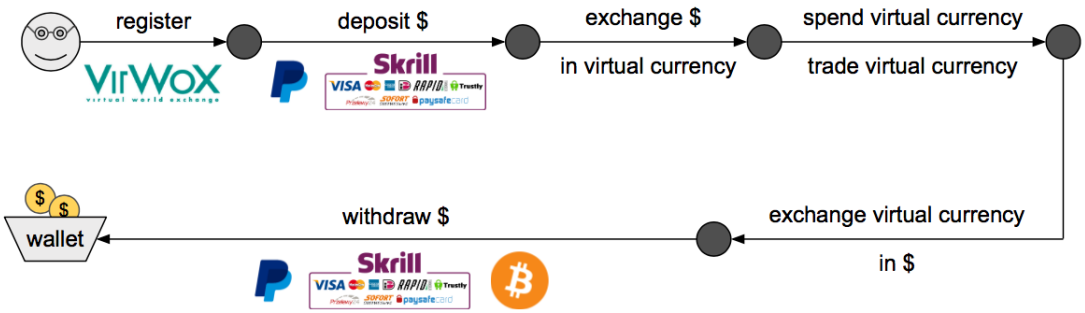

Figure 1.1: **Typical user behaviour described in a flowchart.** After the registration on the website users can deposit money into their account using the payment methods listed in Table 1.2 according to the deposit limits highlighted in Table 1.1. The user can then exchange the money into other virtual currencies, trade it with other users on VirWoX and transfer it to a Second Life account. Additionally, the user can exchange the virtual currency back to euros, dollars or Swiss francs and withdraw it using the payment providers listed in Table 1.2.

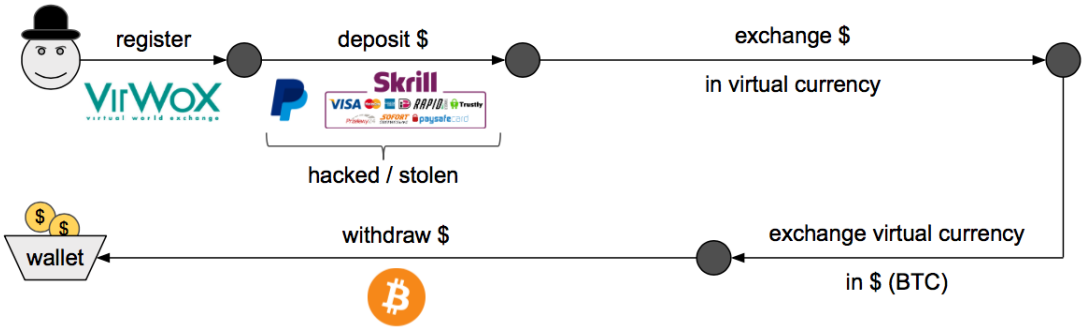

Figure 1.2: **Fraudster behaviour described in a flowchart.** After the registration on the website, fraudsters deposit money into their account using stolen or hacked credit cards and PayPal accounts. In the first step, they exchange the money into Linden dollars followed by the exchange of Linden dollars into bitcoins. Finally, the cybercriminals withdraw those bitcoins to their wallet.

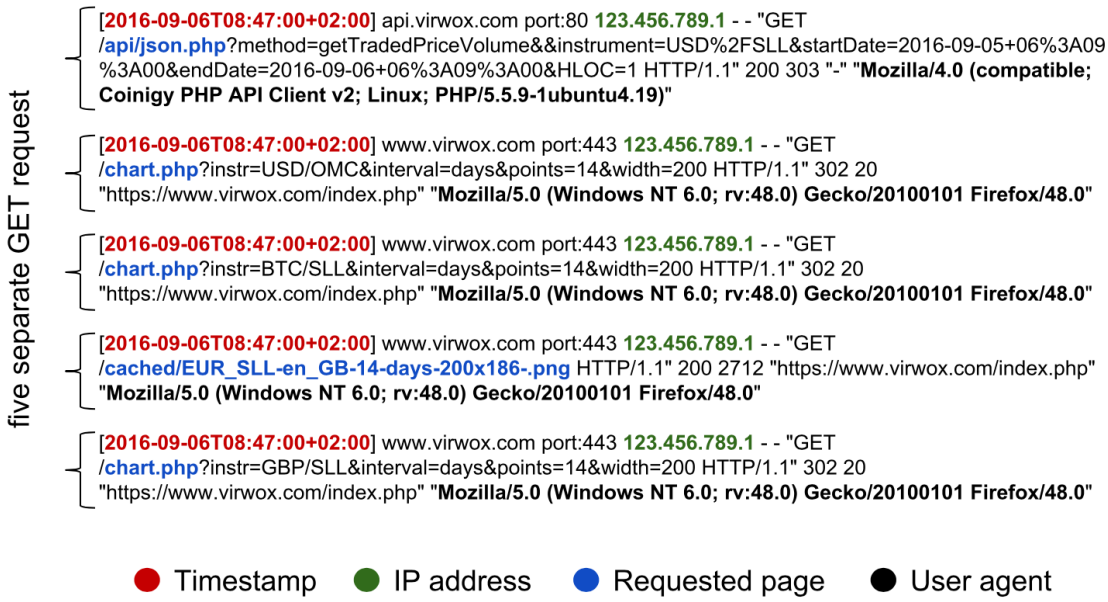

Figure 1.3: **The structure of an Apache log file.** Each entry contains four main parts of information. Highlighted in red one can find the timestamp when the resource was requested. The IP address of the user requesting the resource is coloured in green, followed by the requested page highlighted in blue. Black indicates the user agent of the user's web browser.

1.2 Problem Statement

There are four essential aspects, which we considered to develop the fraud detection system. The selection of the right algorithms and the way of presenting and explaining the results is crucial to design a system, which can be trusted and therefore used on a daily basis. This section explains the four key aspects, which we considered for the choice of the algorithms to ensure such requirements. Further, we formulate the research questions defined out of these requirements.

1.2.1 Requirements

Transparency

One key property of the proposed fraud detection system should be the transparency of the results it shows to the operator. The system only provides an added value to the operator if the proposed results are comprehensibly and not only a simple number, indicating the risk level of a user. This transparency increases the trust in the system.

Concept Drift

Virtual World Services defines a fraudster as a user, who uses stolen credit cards or PayPal accounts and abuse their service to launder the stolen money through Bitcoin. Although the definition of fraudsters stays the same over time, they constantly develop new strategies to bypass fraud detection systems. Therefore, a system needs to be able to deal with this concept drift and adjust to new strategies.

Comparability

Another goal of the system is to provide the analyst a ranked list of users with a score, indicating how abnormal a given user behaviour is. This score enables the analysts to prioritise on the most suspicious users and to investigate why potential false positives and negatives are ranked incorrectly, which is not possible with a binary system that classifies users as fraud or not.

Performance

One obvious property to consider in the selection of the algorithm is its performance. It is split up into two sub-properties.

First, the algorithms must be able to identify as many fraudsters as possible and at the same time keep the number of false positives as low as possible. Second, due to the concept drift it is crucial that the system can be trained in a moderate amount of time and predict if a user is fraudulent or not in almost real time.

1.2.2 Research Questions

Out of the above-defined requirements, we stated the following research questions.

- **RQ1:** Is it possible to separate fraudulent from regular users by analysing the browsing behaviour of the users on the website?
- **RQ2:** Which algorithms are suitable to perform such a task, while meeting the stated requirements?
- **RQ3:** What kind of behaviour indicates fraudulent behaviour?

1.3 Limitations

VirWoX provided us as little information as needed to implement a proof of concept and demonstrate if a data-driven fraud detection system is feasible. Due to this limited amount of data, some limitations arose which are described in this section.

One of the main drawbacks we encountered was the lack of cookie information in the Apache logs to precisely assign the sequences of visited web pages to the customers. Therefore, we needed to rely on some heuristics, which we described in Section 4.1, to assign the sequences to a customer. This lead to the effect that the sequences were not always correctly assigned to a customer, for example, when two users connected from the same network at almost the same time.

Another limitation we encountered was the missing information about when an actual deposit or withdrawal took place since the Apache logs only contained information about the requested resources but not the actions performed on them. Such data could be used to train a model to make a decision only at the point in time when users withdraw money using Bitcoin from their accounts. Only then a potential loss can be

produced. Therefore, we focused our research on detecting fraudulent behaviour as early as possible and made predictions after each finished session of the users.

1.4 Outline

This thesis consists of seven chapters. First, the introduction provided an overview of the topic and the problems which should be addressed by this thesis. Chapter 2 reviews the related work in the field of fraud and outlier detection. In Chapter 3, we present the algorithms we used to perform fraud detection. Further, we explain the performance and evaluation metrics. Chapter 4 describes our experimental setup covering the necessary steps to clean, process and aggregate the provided data as well as an in-depth data analysis. We then close the chapter with the explanation on how we fitted, optimized and tested all the different machine learning models. In Chapter 5 we present the results of the conducted experiments, followed by the discussion of the results in Chapter 6. Finally, we conclude this thesis in Chapter 7 by summarizing our main findings and outlining future work which could be considered to improve the proposed fraud detection system.

2 Literature Review

Algorithms to detect abnormal behaviour are addressed continuously in current research, as shown by Ji Zhang [21]. Also in the industry (e.g. banking industry) sophisticated fraud and outlier detection algorithms are needed. Besides that, there are several other domains in which one could apply outlier detection algorithms. Therefore, we decided to structure the related work into the more general category of outlier detection, fraud detection and intrusion detection.

2.1 Outlier Detection

There is one common problem which is addressed by all the outlier, fraud or intrusion detection systems. Namely, they are designed to detect unwanted and abnormal behaviour. Grubbs [11] defined an outlier as follows:

"An outlying observation, or outlier, is one that appears to deviate markedly from other members of the sample in which it occurs."

This is a more abstract definition of the problem, which also fits the more specific fields of fraud and intrusion detection. For example, one can argue that fraudulent behaviour deviates notably from normal behaviour. Otherwise, it would not be fraudulent. The same argument is valid for intrusion detection. An intruder behaves substantial different than regular users, for example by brute-forcing the login credentials.

Hodge and Austin [13] surveyed contemporary outlier detection methods. The authors explained the initial motivations of the different algorithms and discussed their advantages and disadvantages in a comparative review. Another survey by Gupta et al. [12] focused on outlier detection methods dealing with temporal data. The importance of detecting outliers in temporal data increased drastically in recent years due to the exponential increase in data production. This is also a problem we faced in our research, namely handling the vast amount of data which arrives continually, which requires the model to detect outliers in a single scan. Besides the techniques for detecting outliers in temporal data, Gupta et al. also presented various outlier definitions and introduced the corresponding techniques and applications.

2.2 Fraud Detection

As already mentioned at the beginning of this chapter, the banking industry, for example, is one sector where an automated fraud detection system is crucial. Carminati et al. [7] proposed an online banking fraud detection system (BankSealer) to support the decision-making process of whether transactions should be executed immediately or handed over to domain experts for a manual review before execution. Similar to the requirements stated in this thesis, the authors of BankSealer also emphasized developing a transparent system. Therefore, BankSealer uses three simple algorithms and combines the results into one fraud score. First, they used the histogram-based outlier score (HBOS) developed by Goldstein and Dengel [10] to build local profiles for each user. Second, they used the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) clustering algorithm to calculate global profiles. For new users, the authors then calculated the Cluster-Based Local Outlier Factor (CBLOF) using the clusters found by DBSCAN.

Brause et al. [2] chose another approach and combined a rule-based system with a neural network to perform credit card fraud detection. Another novel approach was proposed by Vlasselaer et al. [20] for automated credit card fraud detection. The authors combined features based on the customer's spending history with network-based features exploiting the network of credit card owners and merchants. The network-based features captured information about which merchants were often related to fraud, if credit card holders acted irregularly and transactions by combining evidence of the associated credit card owner and merchant.

2.3 Intrusion Detection

Another domain of outlier detection is intrusion detection in computer networks. Garcia-Teodoro et al. [9] gave a brief overview of the foundations of well-known anomaly-based network intrusion detection systems (A-NIDS) in their paper. The authors covered statistical-based, knowledge-based and machine learning based methods in their work. Buczak and Guven [6] conducted a more recent survey of data mining and machine learning methods for intrusion detection.
Besides the evaluation and comparison of almost any widely-used algorithms like neural networks, Bayesian networks, clustering methods, decision trees, ensemble methods, genetic algorithms, Hidden Markov Models (HMM) and many others, they also gave an overview about the major steps in machine learning and data mining. Further, the authors provided recommendations on when to use a given method.

In the present thesis, we focus on detecting a small set of fraudulent users out of other, regular users. In the next chapter, we describe the methods we used to achieve this goal, together with the explanation on why we chose these methods. Further, we present the theoretical background of the evaluation metrics used to measure the performance of the fraud detection system.

3 Methodology

In our work, we focused on two domains of machine learning algorithms, namely supervised and unsupervised learning algorithms. Supervised learning algorithms use labeled data, in our case if a user is fraudulent or not, to learn a function that maps a given set of input features to a single number (probability), indicating how fraudulent a given user is. In other words, given a set of input and output pairs, machine learning algorithms try to learn a function that best approximates the relationship of those data pairs. Labeled data is not always accessible since it often needs to be done by humans, which is expensive. Further, the labeling process can be error-prone, especially when humans are involved. Therefore, we also considered unsupervised learning algorithms, because they do not require any labeled data to learn such a decision function. Instead, they try to learn the underlying distribution of the data.

Another aspect which we considered in the selection of the algorithms was their transparency, as already mentioned in Section 1.2. For this reason, we did not use complex algorithms (e.g. neural networks), since most of them do not provide straightforward explanations and therefore, are hard to interpret for humans.

In the following sections, we will first describe the algorithms, which we selected as candidates for the fraud detection system, together with the explanation why we chose them. After that, we will explain the selected metrics to evaluate the performance of the algorithms. Finally, we will illustrate the process of finding the best hyperparameters for the algorithms which are then used to achieve the results shown in Chapter 5.

3.1 Supervised Fraud Detection

3.1.1 Logistic Regression

In supervised learning, the logistic regression is often the first algorithm which comes into mind to predict the probability of a binary event. In contrast to the linear regression, logistic regression has the benefit that the output of the model is calibrated, which means that the output is always between 0 and 1. To ensure that the output always lies within this range the so-called sigmoid function is applied to the result of a linear

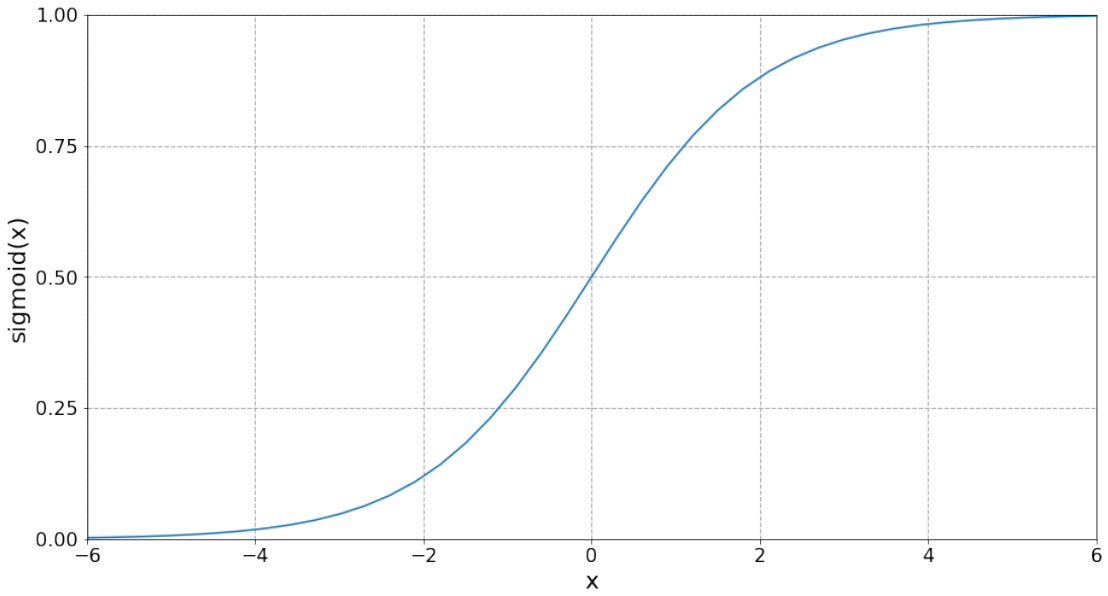

Figure 3.1: **Sigmoid function.**

model as shown in Equation 3.3. In Equation 3.1 one can find the definition of the sigmoid function with the corresponding plot in Figure 3.1, where x is provided by a linear model as defined in Equation 3.2.

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{3.1}$$

$$y' = b + w_0 x_0 + w_1 x_1 + \cdots + w_n x_n = b + \sum_{i=0}^{n} w_i x_i = w^T x + b \tag{3.2}$$

$$p(C = 1|x) = sigmoid(w^T x + b) = \frac{1}{1 + e^{-(w^T x + b)}} = y' \tag{3.3}$$

To fit the model to the data one minimizes the cross-entropy error (CEE) also known as the logistic loss as defined in Equation 3.4. The logistic loss has asymptotic behaviour on both ends when the difference between the predicted and the actual value is close to 1 as shown in Figure 3.2. This asymptotic behaviour leads to large loss values, and therefore regularization is essential for logistic regression. One possible regularization is the L_2 regularization which penalizes huge weights.

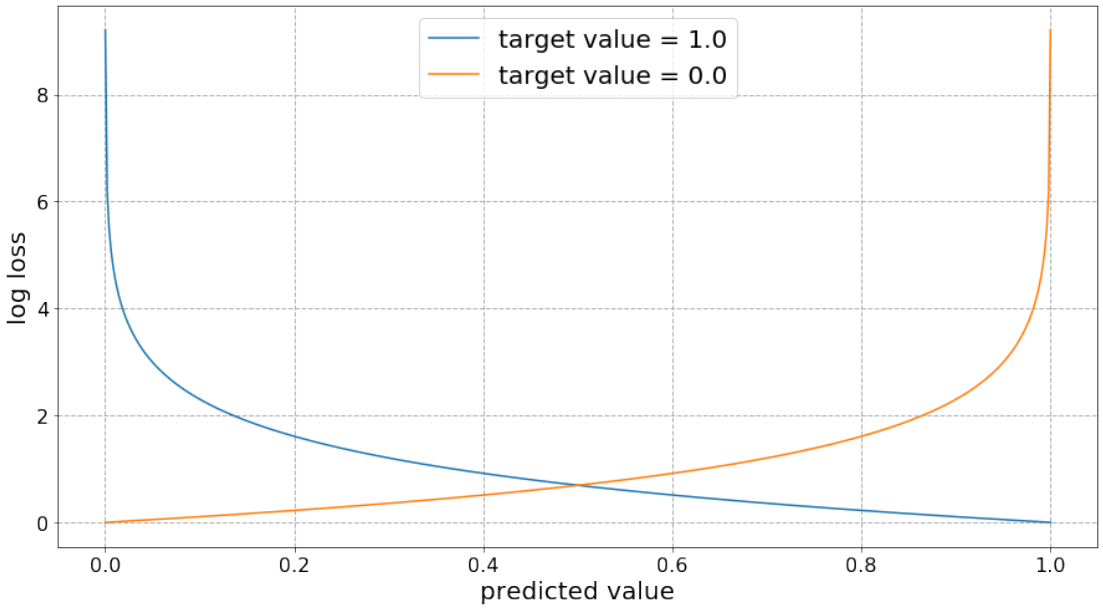

Figure 3.2: **Logistic loss.**

$$CEE = -\sum_{n=1}^{N} [y_n \log(y'_n) + (1 - y_n) \log(1 - y'_n)] \tag{3.4}$$

We concluded that logistic regression matches our performance requirement due to its simplicity while providing interpretable results through the coefficients of the logistic model. Further, the probabilistic output can be used to compare and rank the predictions of the model. These properties lead to the inclusion of logistic regression in our analysis.

3.1.2 Decision Tree

Another well-known algorithm to perform supervised learning classification tasks is the decision tree as introduced by Breiman [4]. The basic idea behind this algorithm is to break up a complex decision into a structured set of simple decisions represented by a binary tree as shown in Figure 3.3. The goal of the tree is to separate all samples by asking the right questions at the right time resulting that in every leaf node there are only samples of one class.

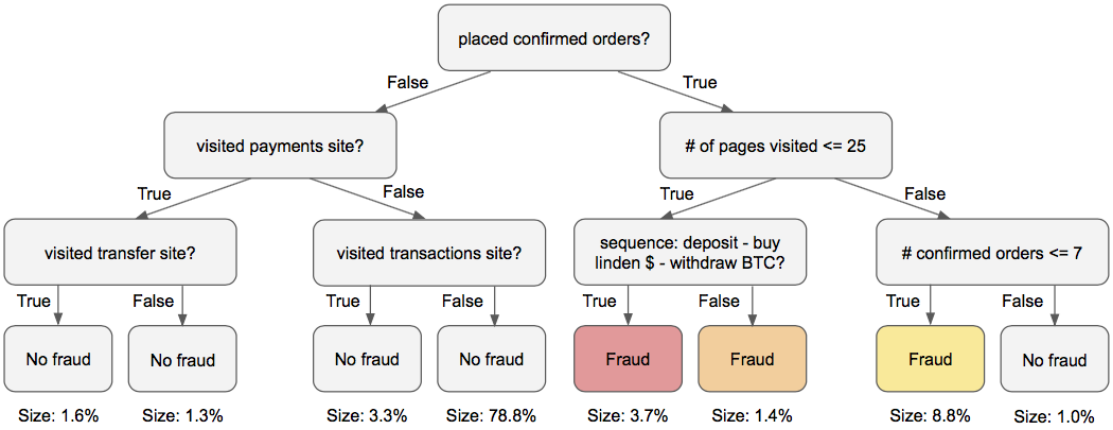

Figure 3.3: **Example decision tree.** Starting at the root node, a sample is assigned a probability of belonging to a class by answering the stated questions in the tree until one reaches a leaf node. This probability is calculated during the training process and represents the fraction of true classes which ended up in this leaf node. This leads to the effect that the classification process of a sample also explains why the sample was classified this way. Further, the higher the node in the tree, the higher is the importance of the question.

To train such a tree, one needs to know which questions to ask and when. To do so, one needs first to quantify how much a question helps to separate the classes. This can be measured using the Gini impurity and the information gain. The Gini impurity measures the chance of labelling an example incorrect by randomly assigning a label from a set to an example of that set. For example, if one has two bowls, one with three black balls in it and the other one with the three labels of the balls. In this case, the chance to incorrectly assign a random ball to a random label is zero. On the other hand, if the first bowl contains one black, one green and a red ball and the other one the corresponding labels, the chance to incorrectly assign a random ball to a random label is two thirds, because there is a one out of three chance to being right. One can find the definition of this metric in Equation 3.5, where $|C|$ represents the number of different classes and p_i the probability of i-th class.

$$I_G(p) = 1 - \sum_{i=1}^{|C|} p_i^2 \tag{3.5}$$

The information gain quantifies how much a question reduces the uncertainty or in other words how much a question helps to separate the samples at a node. It calculates the Gini impurity before the split and the weighted average of the Gini impurity on the split sets. The difference of the Gini impurity before and after the split is then used as a metric for the information gain. During training, all possible questions get evaluated,

and the one with the highest information gain is chosen. This procedure repeats until there are no more mixed classes in the leaf nodes or until a stopping criterion like a maximal tree depth is reached.

To sum up, decision trees have one important property, namely its intuitive interpretability. As already explained in Section 1.2, the transparency of a fraud detection is crucial to trust such a system. However, there is a difference between the theoretical and the practical interpretability of an algorithm. No matter how deep and wide a decision tree gets, it is always theoretically interpretable. Due to the limited capacity of a human, the threshold between the practical and theoretical interpretability needs to be defined together with the domain-experts, who will operate and use the fraud detection system on a daily basis. In Section 4.3 we discuss how we set this parameter. Further, depending on which leaf a sample ends in, one can compare and rank them to focus only on the most suspicious users. Last but not least, the training time of this model also reduces by limiting the maximum depth of the tree, which is a beneficial property to handle the discussed concept drift in Section 1.2.

3.1.3 Random Forest

Random forest as introduced by Leo Breiman [5] is an ensemble method that uses a combination of several decision trees. It uses the concept of randomization to learn a diverse set of tree-based classifiers and combines them into a forest. In an empirical comparison, Caruana and Niculescu-Mizil [8] showed that random forests outperform decision trees on several tasks and datasets.

The first step in the training process of a random forest is to set the number of individual trees, followed by bagging as introduced by Breiman [3]. Bagging means to split up the whole training set into several bootstrap replica sets and train each tree on a separate set. Each tree classifier has a high variance on its own, but the combination of all trees eliminates this effect and results in a stable classifier. To ensure this property, the individual trees must be diverse and therefore bagging is introduced. It also has the property that it speeds up training of each tree because only a subset of the original training set is used.
Further, Breiman introduced random feature selection [5], where in the growing process of a tree only a random subset of features is used at each node to conduct the splitting test. Once all trees are grown the class of a sample is determined by the majority of the individual votes.

Since a random forest is a collection of decision trees, it also has the property of being theoretically interpretable. As already mentioned in the previous subsection, the maximum depth of a single decision tree needs to be limited to stay interpretable for humans.

So by using a random forest, one loses the property of the practical interpretability, because one can not expect an operator of the fraud detection system to understand and remember the relationship between all the trees, especially if the number of individual trees is large even if the maximum depth is restricted. However, we included random forests in our analysis to evaluate if we can achieve better performance with a more complex model.

3.1.4 Handling the Class Imbalance

When applying supervised learning on real-world problems, one often has the problem of imbalanced class distribution in the dataset. Especially in fraud detection scenarios, this distribution can be very skewed since the fraction of fraud users is very small compared to the rest of the users. There are several ways to address this problem like oversampling or undersampling the dataset to get equally distributed classes or to cost-modify the loss. Oversampling randomly selects and duplicates samples of the underrepresented class until one reaches an equal distribution. On the other hand, undersampling randomly selects and removes samples of the overrepresented class until one reaches an equal distribution. Cost-modifying the loss means to penalize the misclassification of an underrepresented sample with a higher cost and the misclassification of an overrepresented sample with a lower cost. Japkowicz and Stephen [14], as well as Lawrence et al. [15], showed in their study that cost-modifying the loss produces better results without artificially increasing the training set size through sampling techniques. For this reason, we chose this strategy to address the class imbalance.

To modify the cost according to the class distribution in a logistic regression, one needs to add the ratio of the class distribution to the logistic loss function. In our case, the ratio is the number of fraudulent samples divided by the total number of samples as shown in Equation 3.6. Assuming that a fraudulent sample is indicated by $y_n = 1$, the ratio is added to the logistic loss as shown in Equation 3.7.

$$r = \frac{N_{fraud}}{N} \tag{3.6}$$

$$CEE_{weighted} = -\sum_{n=1}^{N} [y_n(1-r)\log(y'_n) + (1-y_n)r\log(1-y'_n)] \tag{3.7}$$

In decision trees, the modification of the cost is done by weighting the class probabilities according to the class distribution within the Gini impurity. The Gini impurity is defined

as follows:

$$I_G(p) = 1 - \sum_{i=1}^{|C|} p_i^2 \tag{3.8}$$

The weights are then multiplied with the number of samples in the region of the feature space represented by the node of the decision tree as shown in Equation 3.9.

$$p_i = \frac{w_i n_i}{\sum_i w_i n_i} \tag{3.9}$$

In case of a binary classification problem the weight vector can be defined as follows:

$$\mathbf{w} = [r, 1 - r] \tag{3.10}$$

Since random forests consist of multiple decision trees no additional changes to the cost function, except the one for the impurity within the decision tree, are needed.

With the stated modifications to the loss function of the logistic regression and the impurity of the decision tree and random forest, one can handle arbitrarily balanced classes without artificially increasing the size of the training set. Further, it can be interpreted as a mixture of oversampling and undersampling, because in oversampling one also gives the underrepresented samples higher weights by duplicating them. The same applies to undersampling, where one gives lower weights to overrepresented samples by removing them. Therefore, we considered this approach in our work.

3.2 Unsupervised Fraud Detection

3.2.1 Histogram-based Outlier Score

Histogram-based Outlier Score (HBOS) introduced by Goldstein and Dengel [10] is a fast method to compute an outlier score based on univariate histograms. Such univariate histograms are not only fast to calculate, but also easy to interpret. Further, with histograms, one can model the typical behaviour of a user or a user group and therefore understand its characteristics. For example, Carminati et al. [7] use HBOS in their online banking fraud detection system to build personal profiles for each customer and use it to evaluate the anomaly of new transactions. Since it is an unsupervised method, it also requires no labeled training examples.

First, for each feature, a univariate histogram is constructed. If the feature consists of categorical data, the histogram is constructed by counting the number of elements per

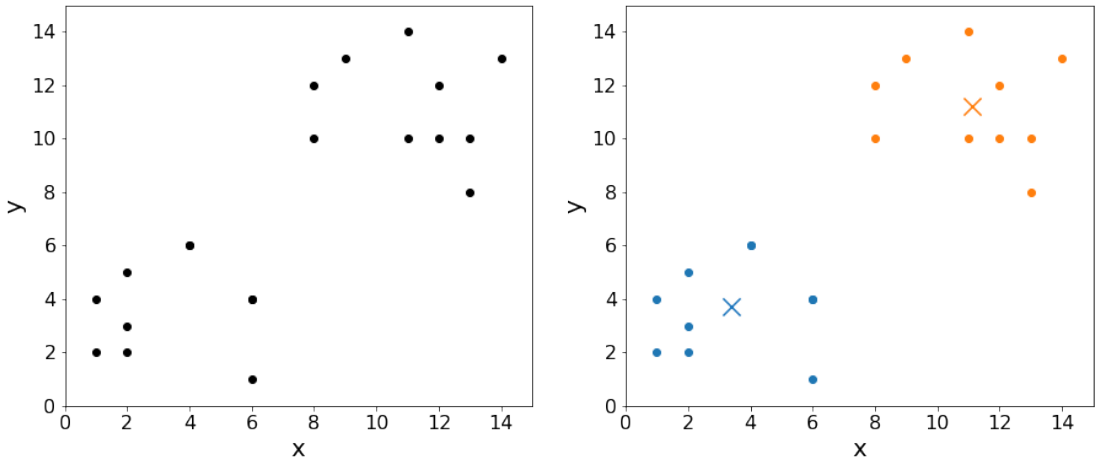

Figure 3.4: **Result of k-Means clustering on a toy dataset.** On the left-hand side, one can see a two-dimensional toy dataset. The right-hand side shows the result of a k-Means clustering with $k = 2$, where both clusters are coloured differently. The two crosses indicate the cluster centers.

category divided by the total number of elements. For numerical data, the histograms can be constructed with either static bin widths or dynamic bin widths, as stated in the paper of Goldstein and Dengel [10]. These histograms are then used to calculate the HBOS for a sample x as shown in Equation 3.11, where d holds the number of dimensions of the sample.

$$HBOS(x) = \sum_{i=0}^{d} log(\frac{1}{hist_i(x)}) \tag{3.11}$$

It can be seen as the inverse of a discrete Naive Bayes assuming independence of the features. A high outlier score implies that the sample is most likely fraudulent, whereas a low outlier score indicates that the sample is normal.

Overall HBOS is easy to understand and therefore transparent to the analyst. Further, the calculation of the histograms is fast which results in short training time. Histograms can also be updated continously as new data arrives and therefore it is capable of online learning. However, all these benefits also come with the disadvantage that the algorithm assumes independence of the features.

3.2.2 k-Means Clustering

One of the probably most used clustering algorithms for unsupervised learning is k-Means. Lloyd [16], Ball and Hall [1], as well as MacQueen [17], independently developed the algorithm in different scientific fields. The algorithm tries to identify groups within the unlabeled data. Figure 3.4 shows an example dataset with the discovered clusters. The objective is to identify groups, where all the samples within a group are similar to each other whereas the groups themselves should be dissimilar to each other. In case of fraud detection, the goal is to identify one or more groups where the fraction of fraudulent users is higher than expected.

The algorithm consists of the following three steps:

1. Initialization of the k cluster centers
2. Assignment of each sample to its closest cluster center
3. Update the cluster centers by shifting it to the mean of the samples within each cluster

The algorithm repeats step 2 and 3 until it reaches a convergence criterion. As a distance measure, the Euclidean metric is typically used in step 2 to calculate the distance between a sample and a cluster center. Further, the number of clusters K in the data needs to be defined. Unfortunately, there is no exact mathematical criterion on how to choose the number of clusters, but there are some different heuristics as discussed by Tibshirani et al. [19] on how to choose a proper K.

To sum up, k-Means clustering is a simple algorithm to find clusters in the data. This simplicity also comes with some drawbacks. As already mentioned, the user has to define the number of clusters. The algorithm also assumes that the clusters have a spherical shape and it can only handle numerical features. Although k-Means clustering is not explicitly designed for outlier or fraud detection, we included it in our research due to its property to find similar users within the data without relying on human labels. Further, it is a rather simple algorithm and the classification of a new sample by finding its closest cluster center can be done in constant time.

3.3 Performance Measures and Evaluation Methods

The choice of the performance measures and evaluation methods is as important as the choice of the algorithms to perform fraud detection. The evaluation of the performance of a machine learning model is a well-studied area. For example, Sokolova and Lapalme [18] performed a systematic analysis of the most used performance metrics. This section

also provides an overview of the basic concepts of training, validation and test sets and how we used them to find the best hyperparameters for each algorithm.

3.3.1 Precision, Recall and F1-Score

Since one requirement of the fraud detection system is its interpretability, we also decided to choose performance metrics which are also easy to interpret. There are two metrics which are easy to interpret and frequently used to measure the performance of fraud detection systems. Namely, how many of the predicted fraudulent users are true fraudsters (precision) and how many true fraudsters are detected in total (recall).

To formalize these metrics, we first need to introduce four commonly used terms to describe the possible predictions of a machine learning model.

1. **True positives (TP):** This term reflects the number of correct classifications of fraudulent users in the prediction.

2. **False positives (FP):** This term reflects the number of incorrect classifications of non-fraudulent users in the prediction.

3. **True negatives (TN):** This term reflects the number of correct classifications of non-fraudulent users in the prediction.

4. **False negatives (FN):** This term reflects the number of incorrect classifications of fraudulent users in the prediction.

The precision measures the fraction of correctly identified fraudsters within the set of users the model predicted as fraudulent and is formalized as follows:

$$Precision = \frac{TP}{TP + FP} \tag{3.12}$$

The recall measures the fraction of the identified fraudsters within the set of all fraudulent users and is formalized as follows:

$$Recall = \frac{TP}{TP + FN} \tag{3.13}$$

A perfect model would achieve an accuracy and recall score of one, by correctly classifying all users. Therefore, the higher the scores, the better performs the model. However, there is a trade-off between these two measures. For example, one can easily

achieve a recall of one by classifying all users as fraudulent and therefore reduce the precision of the prediction. On the other hand, high precision can be achieved by classifying only a few examples, where the model is very confident to be correct resulting in a low recall. To simplify the optimization process for precision and recall at the same time, we used the so-called beta-score. The beta-score is the weighted harmonic mean of precision and recall with the optimal value at one and the worst value at zero and is defined as follows:

$$F\beta\text{-}score = (1 + \beta^2)\frac{Precision \cdot Recall}{(\beta^2 \cdot Precision) + Recall} \tag{3.14}$$

In our research, we fixed the parameter $\beta = 1$ to weight precision and recall equally resulting in the so-called F1-score which can be simplified to:

$$F1\text{-}score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{3.15}$$

As highlighted in the example above, when using two evaluation metrics like precision and recall independently, it is difficult to optimize both at the same since an increase in one metric can lead to a decrease in the other one. Therefore, to conclude this section, we want to highlight that the choice of a single evaluation metric to optimize for is essential because it defines the direction in which the model should emphasize.

3.3.2 Hyperparameter Optimization

In the previous sections, we explained the algorithms which we used for fraud detection and the evaluation metrics to measure and compare the performance of the models. In this section, we describe the process of finding the best performing parameters for the algorithms also known as hyperparameter optimization.

Hyperparameter optimization often includes expert knowledge and a deep understanding of the used algorithms to find good configurations for the machine learning models or at least to narrow down the number of possible parameter configurations. To illustrate the hyperparameter optimization process, Table 3.1 shows an example defining a set of possible hyperparameter configurations for a logistic regression.

In the machine learning domain, it is common practice to split up the dataset into three distinct sets, namely into a training, validation and test set. As the name suggests, the training set is used to train the algorithm with the pre-defined hyperparameters. Once the model is trained, its performance is evaluated on the validation set. To find the best combination of hyperparameters we performed a so-called grid search. It is a simple

Hyperparameter	Description	Values	Parameters
C	Inverse of regularization strength	[0.5, 1.0, 2]	3
intercept_scaling	Synthetic feature equal to intercept_scaling	[0.5, 1.0, 1.5, 2]	4
penalty	The norm used in the penalization	['l1', 'l2']	2

Table 3.1: **Example set of hyperparameter configurations for logistic regression.** In case of a grid search, all 24 combinations are evaluated.

brute-force approach, which tries out all possible combinations within a pre-defined set of parameters. In case of the example shown in Table 3.1, the algorithm is trained with all 24 possible combinations on the training set and evaluated on the validation set. The configuration, which achieved the highest F1-score, is then used to evaluate the final performance of the algorithm on the test set. This procedure is applied to all presented algorithms in Section 3.1 and 3.2. One can then use the F1-score on the test set to determine which algorithm performed best to identify fraudulent users on previously unseen data.

In this chapter, we described the algorithms, the performance measures as well as the methodology to find the best configuration of hyperparameters for each algorithm. Further, we explained how we addressed the problem of class imbalance. In the next chapter, we describe the experimental setup we used to perform the fraud detection experiments.

4 Experimental Setup

In this chapter, we describe our experimental setup which we used to perform the fraud detection experiments. First, we present the data which was provided by VirWoX to perform our analysis. We then describe how we combined and processed the data to construct a dataset. Thereby the data processing step included data preprocessing, data cleaning, feature engineering as well as feature selection. Further, we present the results of an exploratory data analysis to gain a deeper understanding of the data. Finally, we explain the setup to train, validate and test the models and list the best performing hyperparameters.

4.1 Dataset and Terminology

In this section, we take a closer look at the data which was provided by VirWoX. We explain all the features in the data and how we combined them to create a dataset. For all the preprocessing, analysis and prediction experiments we used `Python 3.6` with `scikit-learn`, `numpy` and `pandas` as additional libraries.

The data was provided in four parts. Table 4.1, 4.2 and 4.3 explain the structure of the customer, incident and login event dump provided as comma-separated files. The Apache log files ranged from `2016-06-15` to `2017-07-18` and were provided as separate compressed files per day. One can find the structure of such a log file in Section 1.1. Unfortunately, the log file for `2017-02-20` was missing. Therefore, we could only make use of the data until that date.

Column	Description
customerID	Unique customer ID
username	Username of the customer
disabled	Boolean value indicating if the customer is disabled

Table 4.1: **Structure and description of customer dump.** The dump was provided as comma-separated values file.

Column	Description
customerID	Unique customer ID
timestamp	Timestamp of the incident
readon	Incident description
loss	Amount lost due to the incident

Table 4.2: **Structure and description of incident dump.** The dump was provided as comma-separated values file.

Column	Description
timestamp	Timestamp of the login event
username	Username of the customer who logged in
ipAddress	IP address of the customer during login

Table 4.3: **Structure and description of login event dump.** The dump was provided as comma-separated values file.

customerID	timestamp login	IP address	timestamp incident	readon
1	2016-06-15 11:29:32	50.1.1.1	2016-06-20 12:39:12	PP fraud
1	2016-06-16 12:19:42	50.1.1.1	2016-06-20 12:39:12	PP fraud
1	2016-06-20 10:12:33	64.1.1.2	2016-06-20 12:39:12	PP fraud
2	2016-06-16 21:12:33	80.0.0.1	-	-
2	2016-06-16 22:32:13	80.0.0.1	-	-

Table 4.4: **Example of login event based dataset.** Each row represents one login event including the timestamp and the IP address. For fraudulent users, also the timestamp and the reason of the incident are included.

The goal of the fraud detection system is to detect fraud as early as possible. Therefore, we designed the system to predict if a user is fraudulent or not each time at the end of the user's browsing session. In other words, after the users visited the last webpage of their current browsing session, the system predicts if they are fraudulent or not. To enable such predictions, we constructed a dataset where each sample consists of one browsing session of a user. The data containing the customer information, the login events and the incidents were joined together to form a login event based dataset as shown in Table 4.4.

The next step was to enrich this dataset with the sequential Apache log entries per session and user. Unfortunately, no cookies were provided to assign the log entries directly to a login event. We addressed this problem by using the following steps as a workaround:

1. Filter all Apache log entries to contain only requests to the `index.php` site since users are redirected to this page after the login.

2. Assign the log entry with the smallest difference of the log entry timestamp and login timestamp to the user with the matching IP address. We allowed a maximum delta of 45 seconds to assign an entry to a user.

3. Assign all other log entries to the user by using the IP address of the login, the user agent of the already assigned log entry in step 2 and a maximum time delta of all entries to the login timestamp of 80 minutes.

With this three steps, we fixed the lack of cookie information and successfully combined the provided data. The next section describes the steps to generate our final dataset to train our machine learning algorithms.

4.2 Data Processing and Analysis

Before we feed the data into our machine learning models, we first need to process the data to remove noise, missing and unneeded data. This step is crucial because even the best models are only as good, as the data on which they are trained. A prominent quote in the machine learning domain is *"garbage in, garbage out"*. Which means if one trains a model on faulty data it will also make poor predictions. Afterwards, we describe the features that we built and finish with an exploratory data analysis.

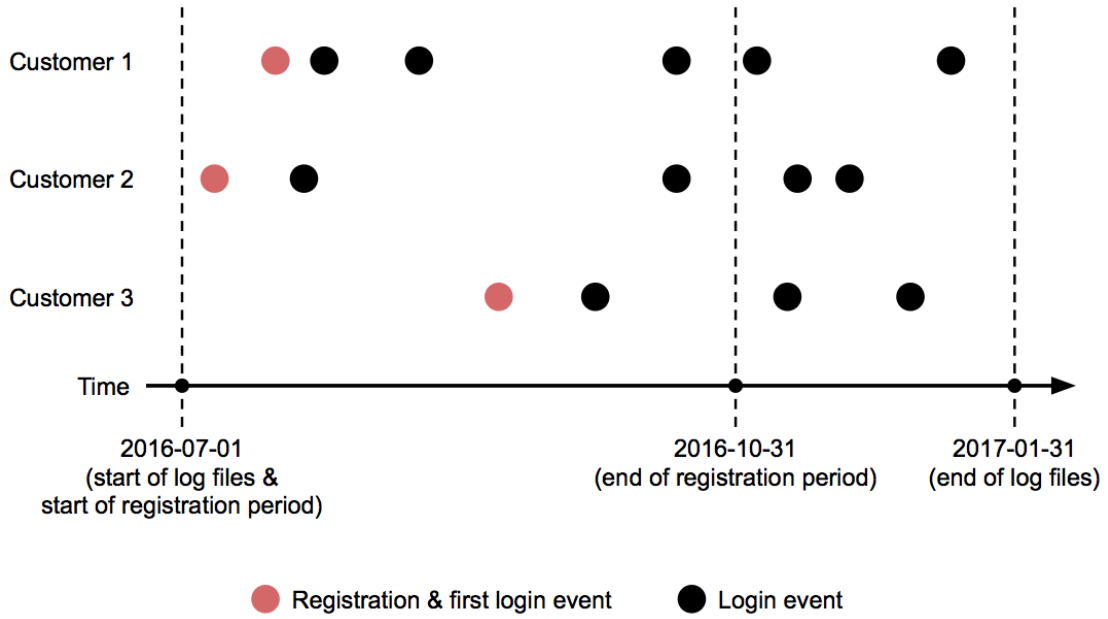

Figure 4.1: **Illustration of registration period.** The time is on the x-axis with three highlighted dates by a dashed line. The first and last dashed lines show the time range of the Apache log files. The dashed line in the middle defines the end of the registration period. The y-axis shows three example customers. The red points indicate the time when the customers have registered and logged in for the first time. The black points indicate the subsequent logins. Note that there are no customers, who had their registration and first login between the last two dashed lines.

4.2.1 Data Cleaning

The first step in our data cleaning procedure was to remove all customers who registered beyond the time frame covered by the Apache log files because the fraud detection system should primarily classify newly registered users. As already mentioned in the previous section, the Apache log files ranged from `2016-06-15` until `2017-02-19`. We decided to cut off the half months at both ends for the sake of simplicity which resulted in a new time frame ranging from `2016-07-01` until `2017-01-31`. To avoid customers in the dataset who registered at the end of this period and therefore only have one browsing session, we defined the final time frame for the registration to be between `2016-07-01` and `2016-10-31`. All other customers who registered beyond this time frame were dropped. In Figure 4.1 we illustrate this process with some example customers.

The Apache log files included every request, which was sent to the server, and therefore contained a lot of unneeded data. For example, requests for images, charts or JavaScript

files. All these requests are irrelevant since they are not triggered by the user consciously but rather as a consequence to properly display the website. Therefore, we browsed through the website of VirWoX and created a whitelist by adding relevant PHP pages. We only kept requests including the PHP files in the whitelist since they reflect the browsing behaviour of a user.

The whitelist included the following PHP files:

- index
- do_trade
- place_order
- deposit
- orders
- cancel
- transfer
- confirm_order
- withdraw
- register
- settings
- transactions
- qrcode
- paypal_receipt
- withdrawal
- logout
- reset_password
- market_depth
- related
- faq
- terminals
- vpn_help
- mfa_settings
- start_paypal_deposit
- start_paysafecard_deposit
- finalize_paypal_deposit
- payments
- help

After the two steps of removing all irrelevant customers followed by the filtering of requests including PHP files, we continued to the process of feature engineering, as explained in the next section.

4.2.2 Feature Engineering and Selection

Feature engineering is one of the most important steps when preparing a dataset. Creating new and relevant features requires domain knowledge and experience. Therefore, it is crucial to have a deep understanding of the data and how it was created. Ideally, feature engineering should be an iterative process and include discussions with involved domain experts to collect and exchange ideas.

In our case, most of the features were created out of the sequential browsing behaviour provided by the Apache log files. All the features are divided into two groups. The first group holds features describing the current browsing session of a user, whereas the second group holds features summarizing all historic browsing sessions, including the current one. We built a cumulative feature for each webpage in the whitelist shown in the previous section indicating the fraction of sessions where a user visited this webpage at least one time. However, we performed a manual feature selection process and removed all features with a small or no difference in the distribution between the fraudulent and regular users. Table 4.5 shows the complete list of features we engineered, after the manual feature selection process.

Feature	Description
# of pages	Number of pages visited in the current session
session duration	Length of the current session in minutes
pages per minute	Number of pages divided by the session duration
recency	How long ago was the last login (in days)
account age	How old is the account currently (in days)
avg. pages	Average number of pages visited in all sessions
log(login velocity)	Average logarithmized login velocity over all successive logins
# of logins	Number of logins up until the current one
# of logins per day	Number of logins divided by the account age
# of countries	Number of unique countries from which a user logged in
avg. pages per minute	Average pages per minute in all sessions
avg. session duration	Average session duration of all sessions
% confirm order	% of sessions where the confirm order page was visited
% withdraw	% of sessions where the withdraw page was visited
% transfer	% of sessions where the transfer page was visited
% deposit	% of sessions where the deposit page was visited
% settings	% of sessions where the settings page was visited
% FAQ	% of sessions where the FAQ page was visited
% help	% of sessions where the help page was visited
% transactions	% of sessions where the transactions page was visited
% withdrawal	% of sessions where the withdrawal page was visited
% terminals	% of sessions where the terminals page was visited
% payments	% of sessions where the payments page was visited
avg. count confirm order	Average number of confirmed orders during the sessions
% mobile device	% of sessions browsed on a mobile device
% pattern deposit & withdraw (PP)	% of sessions where the user deposited money with PayPal, exchanged and withdrawed it using Bitcoin
% pattern deposit & withdraw (Skrill)	% of sessions where the user deposited money with Skrill, exchanged and withdrawed it using Bitcoin
% pattern deposit & withdraw	% of sessions where the user deposited money, exchanged and withdrawed it using Bitcoin
first pattern deposit & withdraw	indicates if the user deposited money, exchanged and withdrawed it using Bitcoin on the first session
fraud	Binary variable indicating if a user is fraudulent or not
readon	Reason why a user got blocked (empty otherwise)
loss	Produced loss by the fraudster (empty otherwise)

Table 4.5: **Feature list of the final dataset.**

After the process of data cleaning, feature engineering and feature selection the dataset can now be used to conduct an exploratory data analysis and to train the different machine learning models.

4.2.3 Exploratory Data Analysis

Before applying machine learning models, it is necessary to gain a deeper understanding of the data and its characteristics. First, we evaluated what kind of incidents produce the largest financial impact to determine on which we should focus. Further, we calculated the distributions of the features and target variables as well as the correlation between the features.

The provided list of incidents contained many different reasons why a given user was blocked. We grouped them into four different categories as shown in Figure 4.2. The current fraud detection system implemented at VirWoX already detects if one user creates multiple accounts through a set of hand-crafted rules. Since the goal of the fraud detection system presented in this thesis is to learn the behaviour of fraudulent users, we need to exclude the users detected by those rules for the following two reasons:

1. We do not want the fraud detection system to learn and replace the hand-crafted rules to detect multiple accounts.

2. The users who get blocked by the current fraud detection system likely were not able to finish the fraudulent behaviour of depositing and withdrawing stolen money. The fact that the users blocked by the current fraud detection system (see Figure 4.2 labeled as `"multiple accounts"` and `"IP address mismatch"`) only produce about 15% of the total loss although they comprise nearly 70% of the incidents confirms this assumption.

Further, we also dropped all other users who got blocked for other reasons than deposit fraud. Deposit fraud represents only about 10% of the incidents but produces over 80% of the total loss. This skewed distribution can be explained by the fact that these sort of incidents are only detected when the victims report the unauthorized transactions on their payment accounts or credit cards. This time window enables the fraudsters to deposit and withdraw enough money to produce this large fraction of loss for the company. To get a deeper understanding of this time window, we analysed how many logins the fraudsters performed until the fraud got detected and reported by the victims. In Figure 4.3, we plotted the fraction of fraudsters and the loss they produced over the number of logins they performed until they got blocked. One can see, that about 56% of the fraudsters got blocked after five logins. However, with an increasing number of logins, also the loss per fraudster increased drastically. For this reason, we only kept

Figure 4.2: **Distribution of incidents.** The left chart shows the distribution of the incidents. Nearly 60% of the incidents are due to users creating multiple accounts. The typical fraud scenario as explained in Figure 1.2 is labeled as `"deposit fraud"` and represents about 10% of the incidents. The right chart shows how the loss produced by those incidents is distributed. Over 80% of the loss is produced by deposit fraud.

	Total	Fraudulent	Fraction fraudulent
Users	55367	1135	0.020
Browsing Sessions	226461	9899	0.044

Table 4.6: **Size of the dataset.** From `2016-07-01` until `2016-10-31` 55367 users registered and 2% of them were fraudsters. They produced 226461 browsing sessions from `2016-07-01` until `2017-01-31` in total where 4.4% were fraudulent. This means that on average fraudulent users produce more sessions than regular users.

the group of deposit fraud to enable the machine learning models to focus on the most harmful and currently undetected incidents. Table 4.6 reports the final numbers of the dataset including only the deposit fraudsters. Further, we plotted the distribution of all features as shown in Figure 4.4 and 4.5 with the corresponding numbers in Table 4.7. To avoid highly correlated features in the dataset, we also plotted the correlation matrix in Figure 4.6 using the Pearson correlation coefficient.

In this section, we investigated the characteristics of the dataset and discovered that over 80% of the loss is produced by a small group of incidents, namely the deposit fraud. We prepared the dataset to contain only this type of fraud to enable the machine learning algorithms to learn and identify those fraudulent patterns. Further, we analysed the distributions of the features.

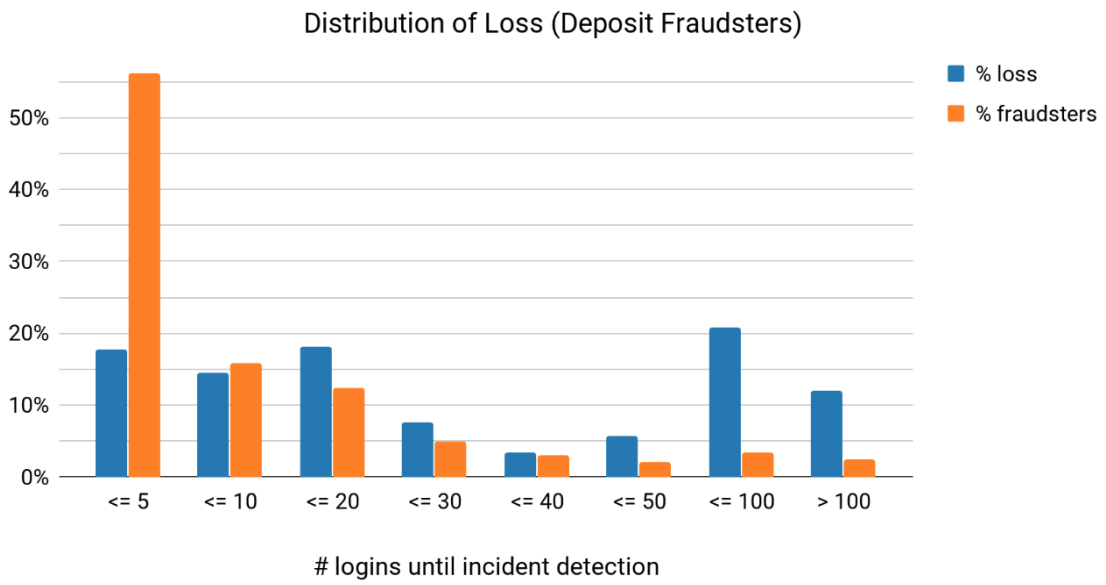

Figure 4.3: **Distribution of loss and fraudsters over the number of logins until incident detection.** The orange bars represent the fraction of incidents (fraudsters) which got detected within the given number of logins. Most of the fraudsters (56%) got blocked within the first five logins. With an increasing number of logins also the fraction of incidents decreases, due to the higher probability of the transactions getting reported by the victim. The last two bins ($<= 100$ and > 100) include a wider range of logins and therefore contain a slightly higher fraction of fraudsters than the previous bins. The blue bars indicate the fraction of loss produced by those fraudsters. One can see the effect of the limits as described in Table 1.1. Most of the incidents occur within the first five logins, but they only produce 18% of the overall loss. However, the loss per fraudster rises rapidly with increasing number of logins. This can be explained due to the increasing deposit limits through time as well as the fact that the longer the fraud stays undetected, the more money can be withdrawn from the victim's account.

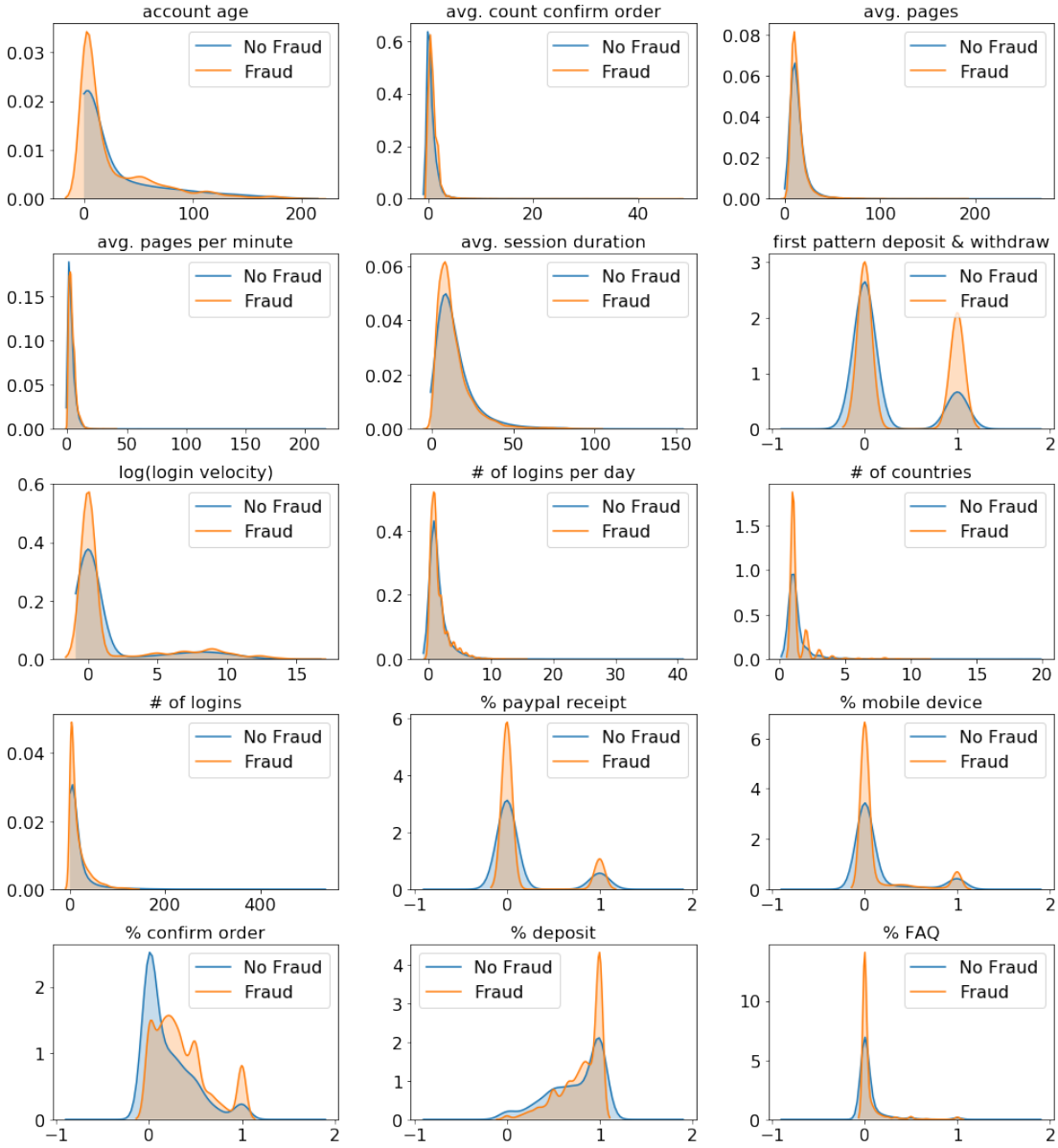

Figure 4.4: **Distribution of features I.** The distribution of the fraudulent samples is colored in orange and the distribution of the regular samples is colored in blue.

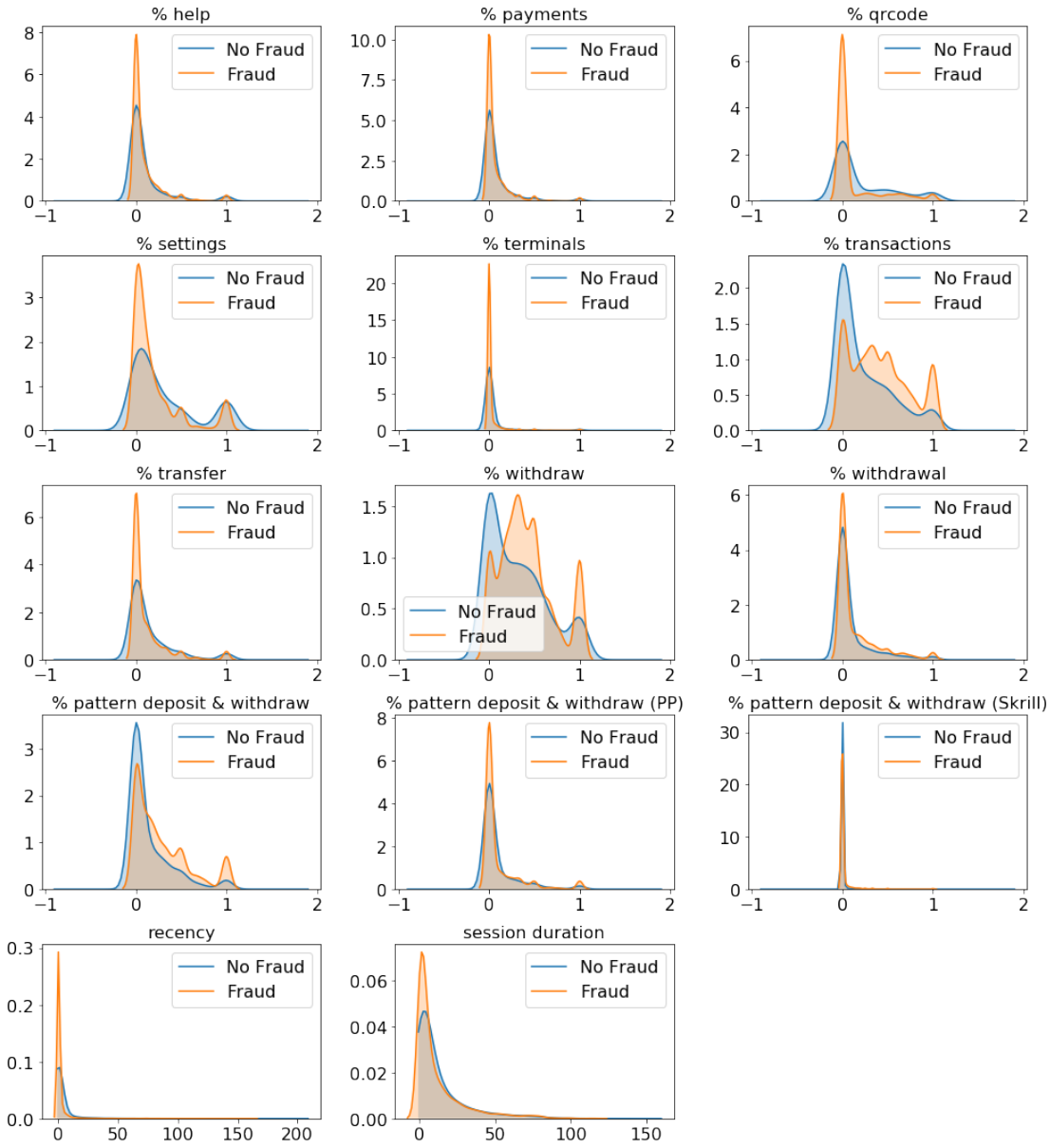

Figure 4.5: **Distribution of features II.** The distribution of the fraudulent samples is colored in orange and the distribution of the regular samples is colored in blue.

			Fraud		No fraud		
Feature	type	min	median	mean	median	mean	max
account age	numeric	1.0	9.0	26.38	3.0	24.68	214.0
avg. count confirm order	numeric	0.0	0.67	0.94	0.22	0.64	48.0
avg. pages	numeric	1.0	11.8	14.11	11.59	13.85	268.0
avg. pages per minute	numeric	0.0	3.32	4.02	2.5	3.37	216.56
avg. session duration	numeric	0.0	10.0	13.21	10.65	14.21	153.65
first pattern deposit & withdraw	binary	0.0	0.0	0.41	0.0	0.2	1.0
log(login velocity)	numeric	0.0	0.0	1.64	0.0	1.23	15.88
# of logins per day	numeric	0.01	1.0	1.56	1.0	1.49	40.0
# of countries	numeric	1.0	1.0	1.39	1.0	1.33	19.0
# of logins	numeric	1.0	9.0	17.3	4.0	14.49	536.0
% paypal receipt	numeric	0.0	0.0	0.15	0.0	0.15	1.0
% mobile device	numeric	0.0	0.0	0.13	0.0	0.14	1.0
% confirm order	numeric	0.0	0.29	0.35	0.1	0.21	1.0
% deposit	numeric	0.0	0.87	0.8	0.83	0.74	1.0
% FAQ	numeric	0.0	0.0	0.05	0.0	0.05	1.0
% help	numeric	0.0	0.0	0.1	0.0	0.1	1.0
% payments	numeric	0.0	0.0	0.07	0.0	0.08	1.0
% qrcode	numeric	0.0	0.0	0.13	0.0	0.22	1.0
% settings	numeric	0.0	0.09	0.21	0.17	0.31	1.0
% terminals	numeric	0.0	0.0	0.02	0.0	0.04	1.0
% transactions	numeric	0.0	0.37	0.41	0.08	0.23	1.0
% transfer	numeric	0.0	0.0	0.11	0.0	0.15	1.0
% withdraw	numeric	0.0	0.36	0.41	0.25	0.32	1.0
% withdrawal	numeric	0.0	0.0	0.15	0.0	0.1	1.0
% pattern deposit & withdraw	numeric	0.0	0.18	0.28	0.0	0.14	1.0
% pattern deposit & withdraw (PP)	numeric	0.0	0.0	0.11	0.0	0.09	1.0
% pattern deposit & withdraw (Skrill)	numeric	0.0	0.0	0.02	0.0	0.0	1.0
recency	numeric	0.0	0.0	1.55	0.0	3.23	208.0
session duration	numeric	0.0	4.28	11.47	5.0	11.81	159.13

Table 4.7: **Dataset base statistics.** This table shows the minimum and maximum values for all features, as well as the median and mean values for both classes.

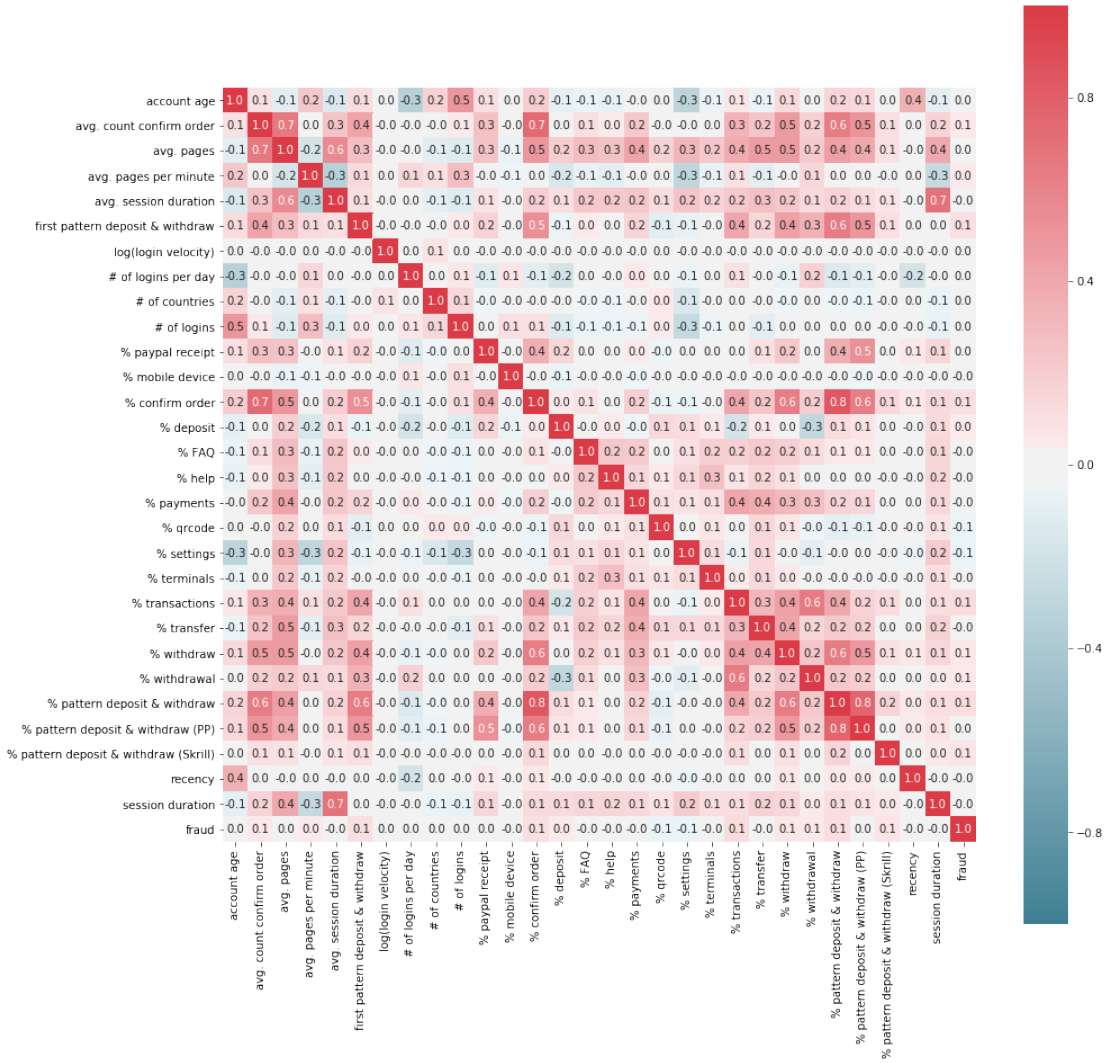

Figure 4.6: **Feature correlation matrix.** The Pearson correlation coefficient was used to calculate to correlation values. No features were dropped due to a too high correlation coefficient.

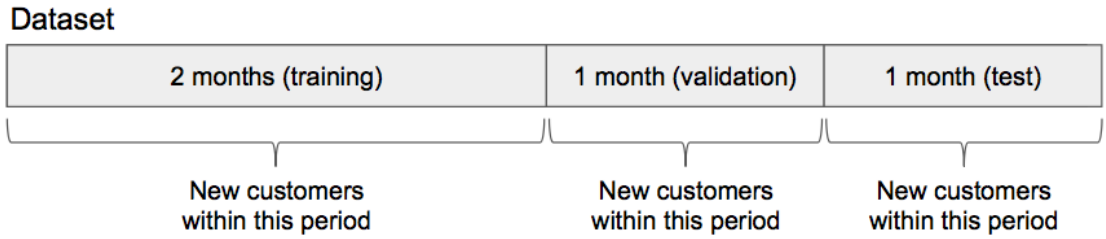

Figure 4.7: **Data splitting.** We divided the dataset into three parts. The training set contained users who registered within two months ranging from 2016-07-01 to 2016-08-31. The validation and test set included users who registered in September 2016 and October 2016 respectively.

4.3 Model Fitting, Optimization and Testing

Once the data is in the right form, and the target for the model is defined one can start to train the machine learning models. First, we need to define how we split the dataset into a training, validation and test set. Further, we show how we used those three sets to find the right hyperparameters for our models and how we evaluated their performance.

There are two ways how we can split the dataset to obtain the mentioned subsets to train, validate and evaluate our fraud detection system. The first one is to assign the users randomly to one of these sets. Usually, the training set should hold about 70%, and the validation and test set 15% of the data. The problem with this approach is that it does not consider the time aspect of the data. For example, fraudsters may have developed new strategies to bypass the current fraud detection system within the last few weeks. Through the random selection of the users to the training set, the model would be able to learn those new strategies and therefore could lead to an artificial increase of the overall performance. Therefore, we decided to split the data time-based. Figure 4.7 shows an illustration of this process. The time-based split has the advantage that it models a real-world scenario where patterns are learned from historical data only and are then applied to future data.

Since one goal of the fraud detection system is to detect fraudulent behaviour as early as possible, we defined the evaluation metric as highlighted in Figure 4.8. With this setup, we can measure the performance of the models after each browsing session and evaluate how much the performance increases by observing the users for a longer period.

After splitting the dataset, we trained the algorithms using the training set and performed a hyperparameter search using the validation set. We then reported the evaluation metrics on the test set using the best performing hyperparameters on the validation

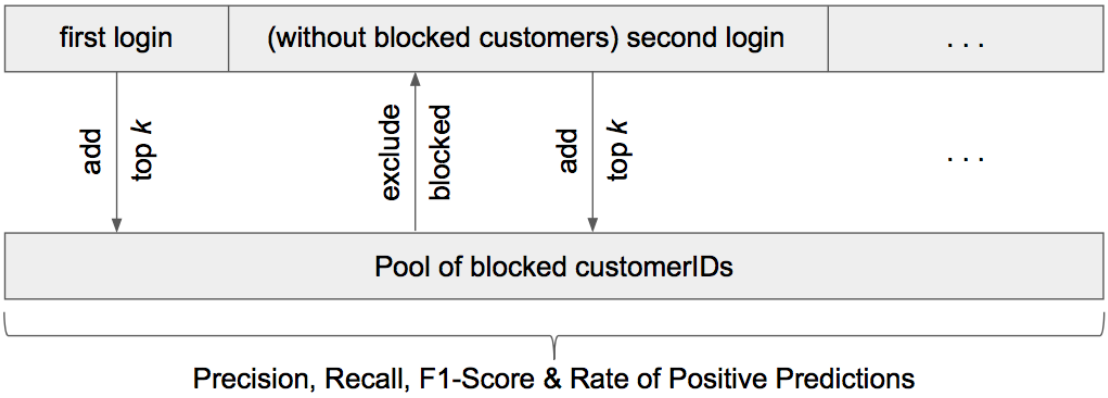

Figure 4.8: **Session-based evaluation.** First, for each user, a prediction is made after the first login (browsing session). The top k users, which got the highest probability to be fraudulent are added to a pool of blocked customers. After that, for each user except the ones already blocked, a prediction is made after the second login (browsing session). Again, the top k users with the highest probability to be fraudulent are added to the pool of blocked customers. This evaluation process is continued until all samples got classified or all fraudulent users are detected. The number k is always equal to the number of fraudsters in the current subset of samples.

set. Figure 4.9 highlights this setup and the Tables 4.8, 4.9, 4.10 and 4.11 contain the hyperparameter configurations which were used in the grid search for the supervised learning models and the K-means clustering. For HBOS we had to use a different approach to find the best hyperparameters. The HBOS algorithm does not have any numerical parameters which can be tuned. Instead, one can search for a list of histograms which performed best on the validation set. Since such a list could include one feature multiple times, which could be interpreted as giving that feature a higher weight, trying out all possible combinations would not be feasible. Therefore, we implemented a so-called "greedy-add" approach as follows:

1. Start with an empty list of features referred to as feature list and initialize a variable holding the maximal F1-score to zero.

2. Iterate through all features, where in each step the feature list is added to the current feature and its performance on the validation set is measured.

3. Add the feature producing the highest F1-score to the feature list if the F1-score is higher than the currently maximal F1-score and update the maximal F1-score.

4. Repeat steps 2. and 3. until no more features are added.

Hyperparameter	Description	Values	Parameters
C	Inverse of regularization strength	[0.05, 0.2, 0.4, 0.8, 1.0, 1.2, 1.4, 2, 2.5, 3]	10
intercept_scaling	Synthetic feature equal to intercept_scaling	[0.1, 0.5, 0.75, 1.0, 1.5, 2.0]	6
penalty	The norm used in the penalization	['l1', 'l2']	2

Table 4.8: **Hyperparameter configurations for logistic regression.** Using a grid search, we evaluated all 120 combinations.

Hyperparameter	Description	Values	Parameters
max_depth	Maximum depth of the tree	[2, 5, 8, 11, 15, 18, 21]	7
min_samples_leaf	Minimum number of samples per leaf (fraction)	[0.001, 0.002, 0.005, 0.01, 0.02]	5
max_features	Number of features to consider when looking for the split	[3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29]	14

Table 4.9: **Hyperparameter configurations for decision tree.** Using a grid search, we evaluated all 490 combinations.

Hyperparameter	Description	Values	Parameters
max_depth	Maximum depth of the tree	[2, 4, 6, 8, 10, 12, 14]	7
min_samples_leaf	Minimum number of samples per leaf (fraction)	[0.001, 0.002, 0.005, 0.01, 0.02]	5
max_features	Number of features to consider when looking for the split	[3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29]	14
n_estimators	Number of trees to use	[5, 10, 15, 30, 50]	5

Table 4.10: **Hyperparameter configurations for random forest.** Using a grid search, we evaluated all 2450 combinations.

Hyperparameter	Description	Values	Parameters
k	Number of clusters	[2 - 30]	29

Table 4.11: **Hyperparameter configurations for k-Means.** Using a grid search, we evaluated all 29 combinations.

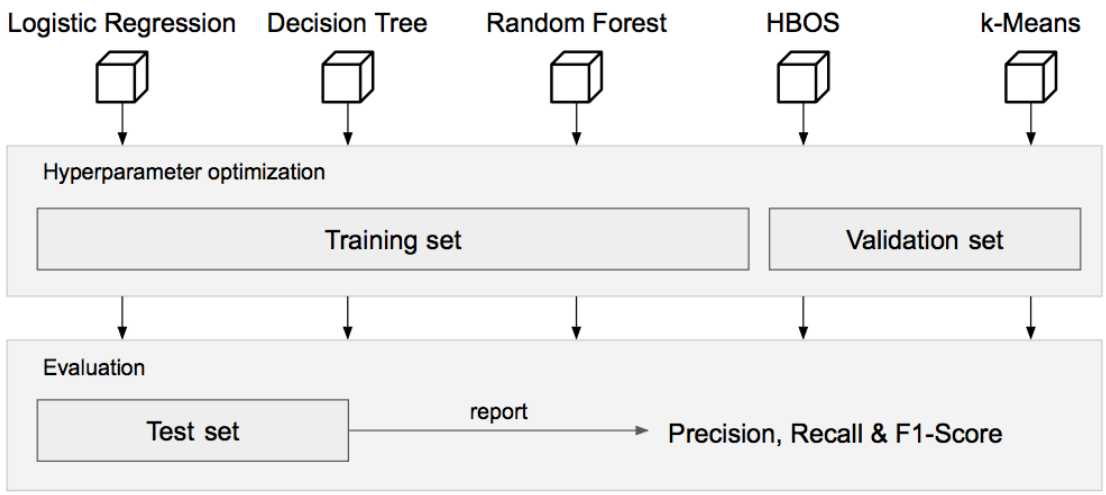

Figure 4.9: **Illustration of training, validation and evaluation process.** All models are fitted on the training set, and the hyperparameter search is performed on the validation set. Finally, precision, recall and the F1-score are reported on the test set.

Finally, in Table 4.12 we reported the best hyperparameter configurations for all algorithms. Additionally, we trained two smaller decision trees with a limited tree depth of three on two subsets of data to increase human interpretability. The first tree was trained only using those samples, where the users performed the first login. The second tree was trained on all other samples. We chose this splitting criterion because some features required more than one login session for its calculation (e.g. login velocity).

In the next chapter, we show the final results reported on the test set using the hyperparameters stated in Table 4.12. Further, we compared all algorithms to a random baseline and included the available feature importance plots of the algorithms. Finally, we plotted the two limited decision trees.

Algorithm	Hyperparameter	Value
Logistic regression	C	1
	intercept_scaling	0.75
	penalty	'l2'
Decision tree	max_depth	21
	min_samples_leaf	0.001
	max_features	11
Decision tree (limited) (first login session)	max_depth	3
	min_samples_leaf	0.005
	max_features	17
Decision tree (limited) (after second login session)	max_depth	3
	min_samples_leaf	0.005
	max_features	11
Random forest	max_depth	12
	min_samples_leaf	0.001
	max_features	29
	n_estimators	10
HBOS	feature_list	[% pattern deposit & withdraw, % transactions, % confirmed orders % pattern deposit & withdraw (Skrill), # of countries, log(login velocity)]
k-Means	k	29

Table 4.12: **Best hyperparameter configurations.**

5 Results

In this chapter, we present the results of our fraud detection system. All the algorithms described in Chapter 3 are evaluated on the test set using the best hyperparameters found during the validation process as described in Section 4.3. We report the performance measures described in Section 3.3 for all algorithms to determine the best model for the fraud detection system. Further, we plot the accumulated recall, accumulated loss as well as the accumulated rate of positive predictions over the number of logins. We also show the plots highlighting the contribution of each feature in the logistic regression, decision tree and the random forest. To increase the human interpretability, we also trained two separate decision trees with a limited depth of three on two subsets of data. For HBOS we show the histograms which were selected for the prediction. Last, we calculated the silhouette coefficients for the k-Means clusters to evaluate it's separation capabilities.

Table 5.1 reports the precision, recall, F1-score and the rate of positive predictions for all algorithms. As described in Section 4.3, we slightly adopted these metrics to model a real-world scenario, where samples are classified in chronological order. We plotted this chronological classification process in Figure 5.1, 5.2 and 5.3 to evaluate the performance of the algorithms, given the number of past browsing sessions used for the prediction.

In Figure 5.4, 5.5 and 5.6 one can find the feature importance for logistic regression, decision tree and random forest. High feature importance values indicate that the feature provides good separation capabilities. Figure 5.7 and 5.8 hold the two decision trees, which were trained to increase the human interpretability. As an equivalent, we plotted the feature histograms used by the HBOS in Figure 5.9 which were selected in the hyperparameter optimization process for the classification. Finally, Figure 5.10 shows the quality of the k-Means clusters using the silhouette coefficients. The coefficients range from -1 to +1, where positive silhouette coefficient values indicate that the samples are far away from neighbour clusters and negative values mean that the sample is probably assigned to a wrong cluster.

Algorithm	Precision	Recall	F1-Score	RPP
Logistic Regression	12.26%	45.37%	19.31%	8.02%
Decision Tree	13.66%	48.88%	21.35%	7.76%
Decision Tree (limited depth)	12.29%	48.24%	19.58%	8.52%
Random Forest	16.03%	53.35%	24.65%	7.22%
HBOS	8.08%	33.23%	14.69%	8.59%
k-Means	9.84%	38.34%	15.66%	8.45%
Random Baseline	3.67%	17.89%	6.09%	10.57%

Table 5.1: **Evaluation results on the test set.** The metrics are slightly modified as described in Section 4.3 to reflect a real-world scenario, where the samples are classified in chronological order.

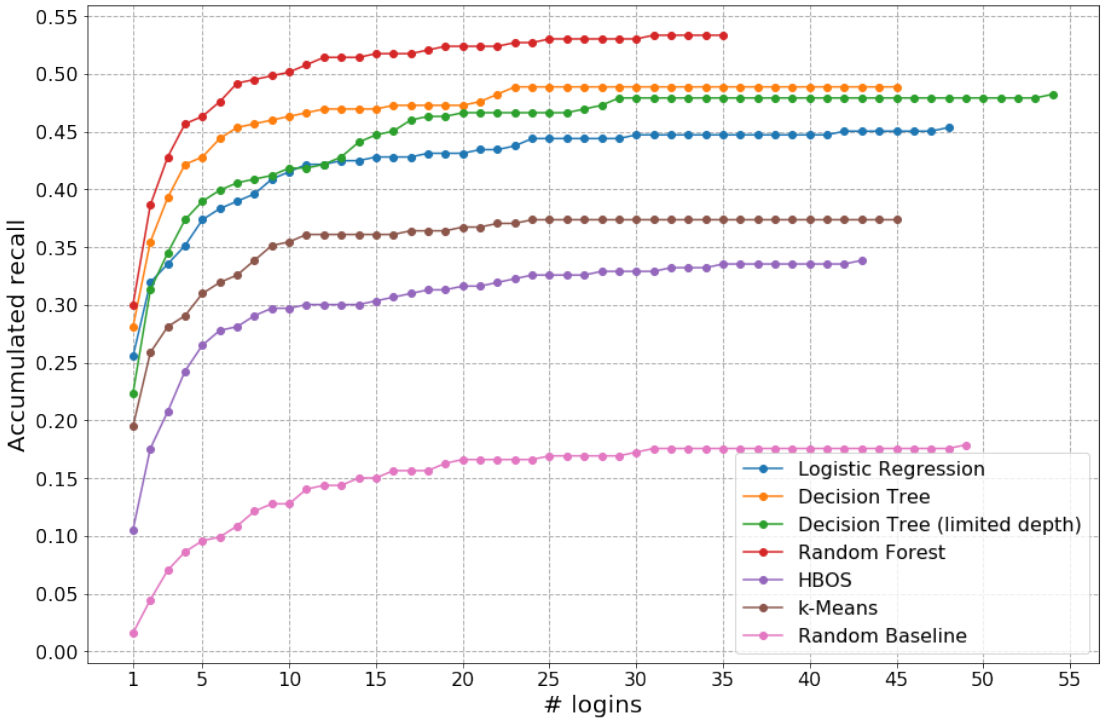

Figure 5.1: **Accumulated recall vs. logins.** The y-axis shows the accumulated recall over the number of logins on the x-axis. Each algorithm is represented with a different color. The lines have a different length due to the definition of the evaluation metric. The evaluation stops if there are no more undetected fraudulent users with the given number of logins or if there are no more samples to evaluate.

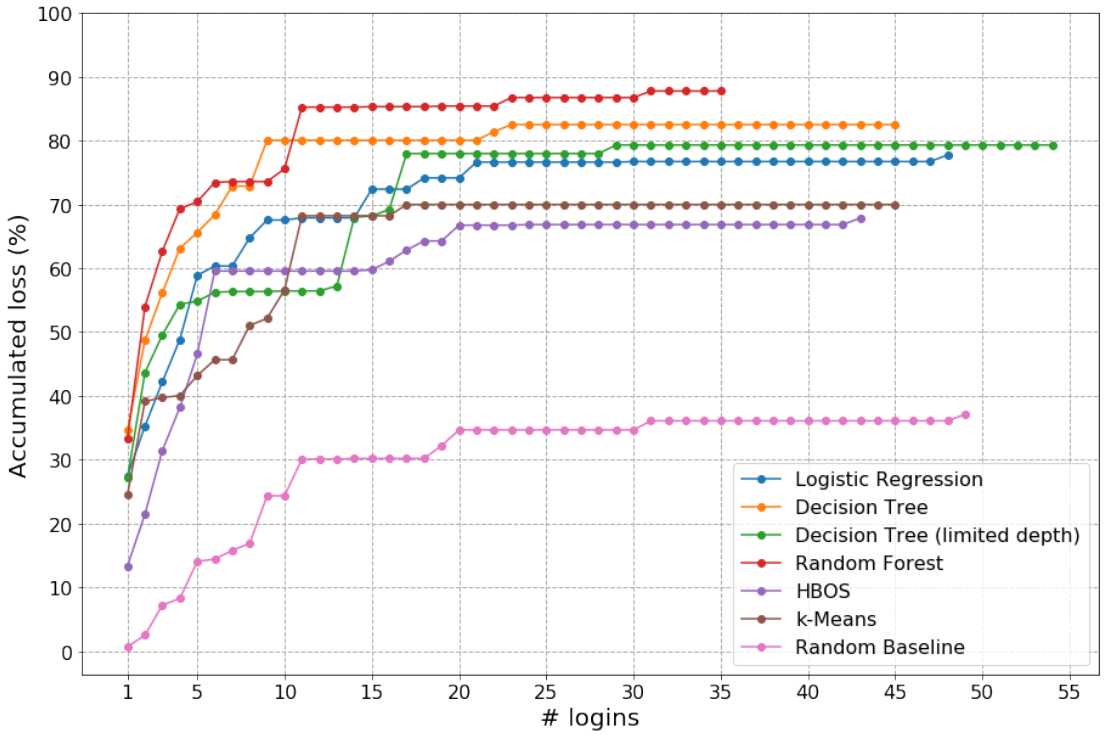

Figure 5.2: **Accumulated loss vs. logins.** The y-axis shows the accumulated loss over the number of logins on the x-axis. Each algorithm is represented with a different color. The lines have a different length due to the definition of the evaluation metric. The evaluation stops if there are no more undetected fraudulent users with the given number of logins or if there are no more samples to evaluate.

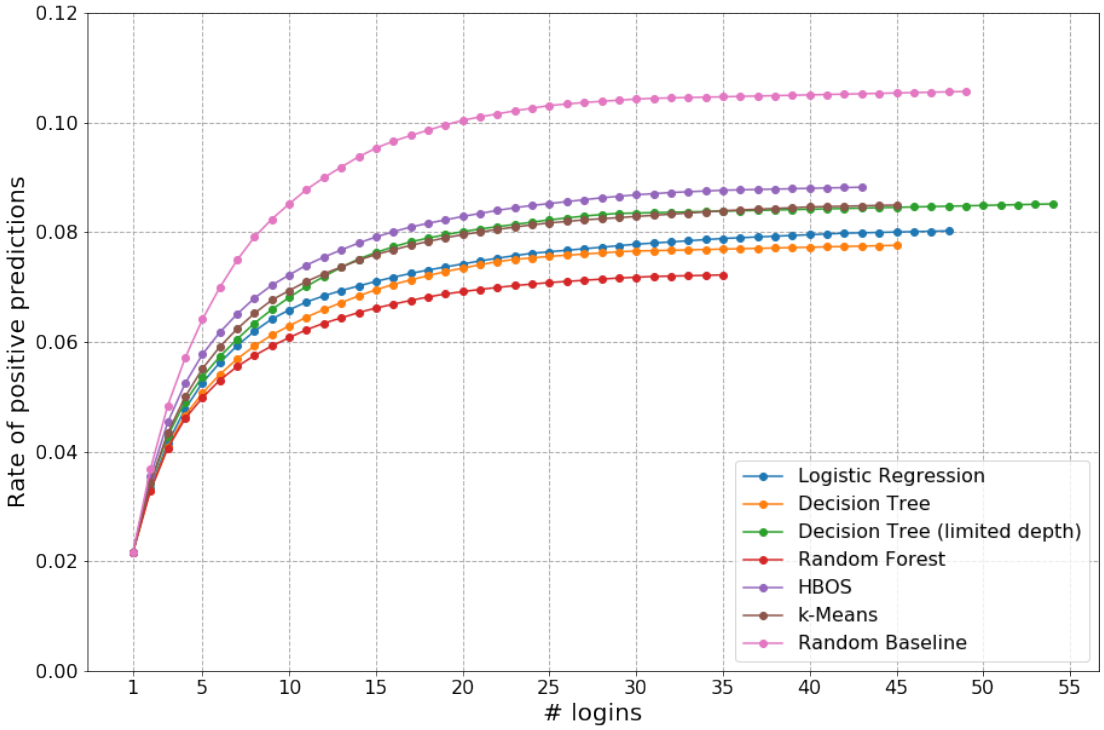

Figure 5.3: **Rate of positive predictions vs. logins.** The y-axis shows the rate of positive predictions over the number of logins on the x-axis. Each algorithm is represented with a different color. The lines have a different length due to the definition of the evaluation metric. The evaluation stops if there are no more undetected fraudulent users with the given number of logins or if there are no more samples to evaluate.

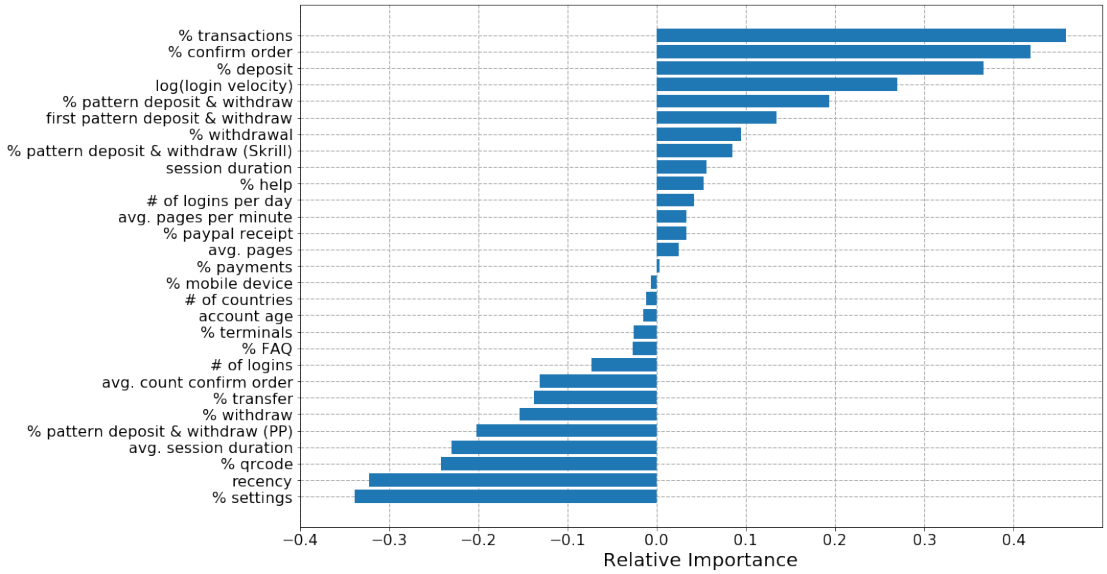

Figure 5.4: **Feature importance of the logistic regression.** The y-axis holds the features used by the logistic regression and the x-axis shows the corresponding relative feature importance of each feature. Negative values indicate a contribution to the class of non-fraudulent users and positive values indicate a contribution to the class of fraudulent users.

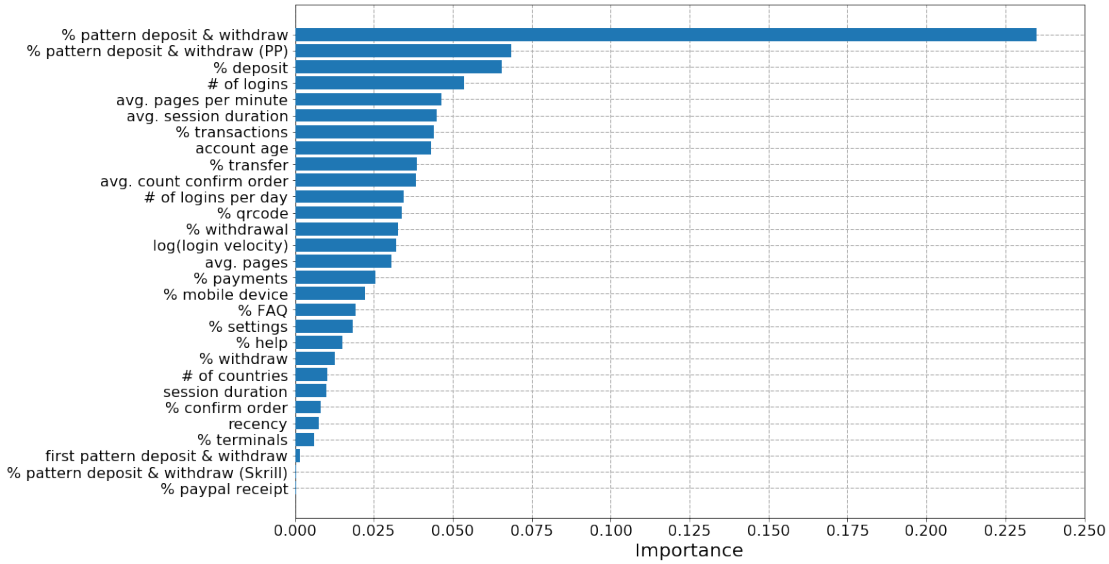

Figure 5.5: **Feature importance of the decision tree.** The y-axis holds the features used by the decision tree and the x-axis shows the corresponding feature importance of each feature. The importance of the features is computed as the normalized total reduction of the Gini impurity.

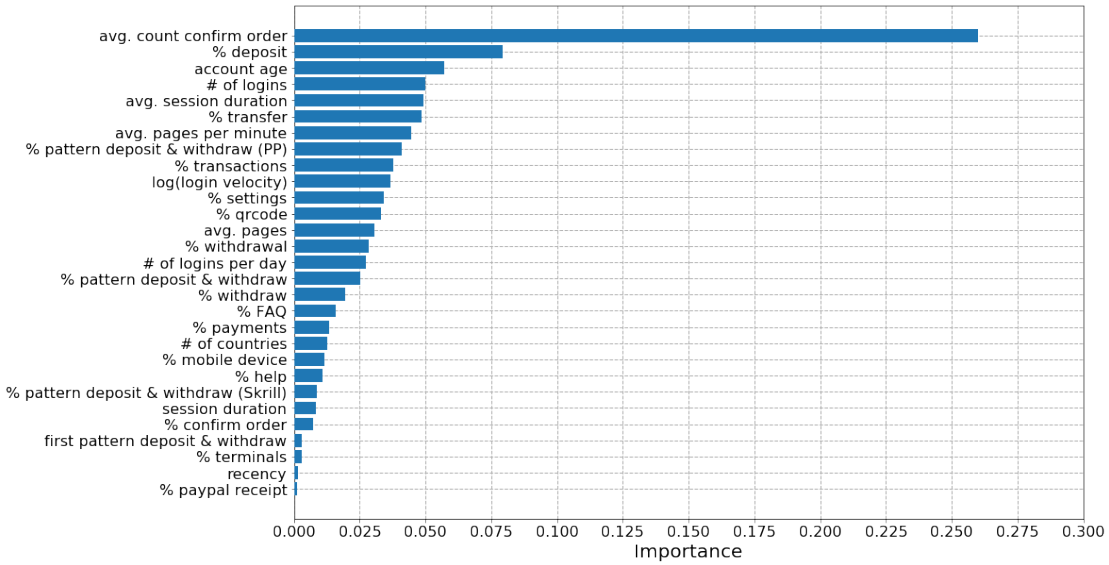

Figure 5.6: **Feature importance of the random forest.** The y-axis holds the features used by the random forest and the x-axis shows the corresponding feature importance of each feature. The importance of the features is computed as the average normalized total reduction of the Gini impurity over all trees.

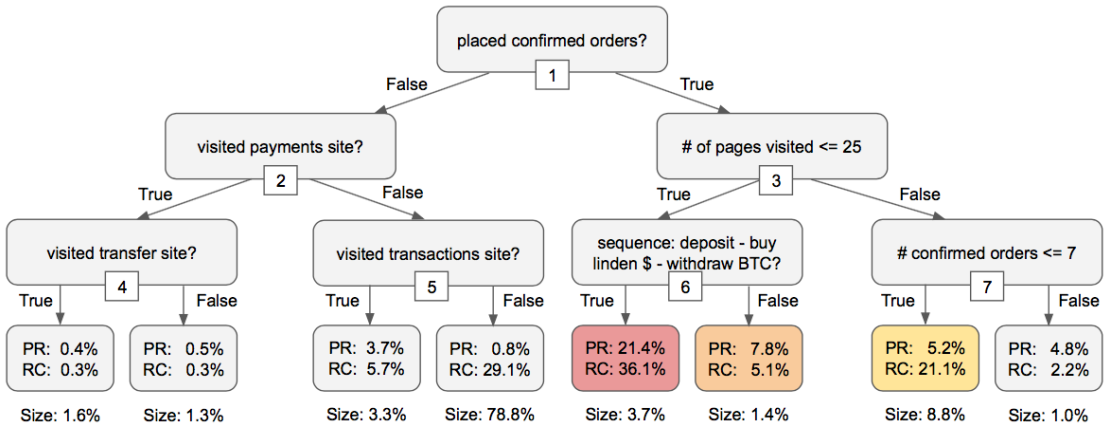

Figure 5.7: **Fraud indicators after first login.** We considered the users belonging to the three leaf nodes with the highest precision as being most likely fraudsters. We label the red leaf node with a precision of 21.4% as high-risk users, the orange leaf node with a precision of 7.8% as medium-risk users and the yellow leaf node with a precision of 5.2% as low-risk users. All other users in the other leaf nodes are considered as regular users.

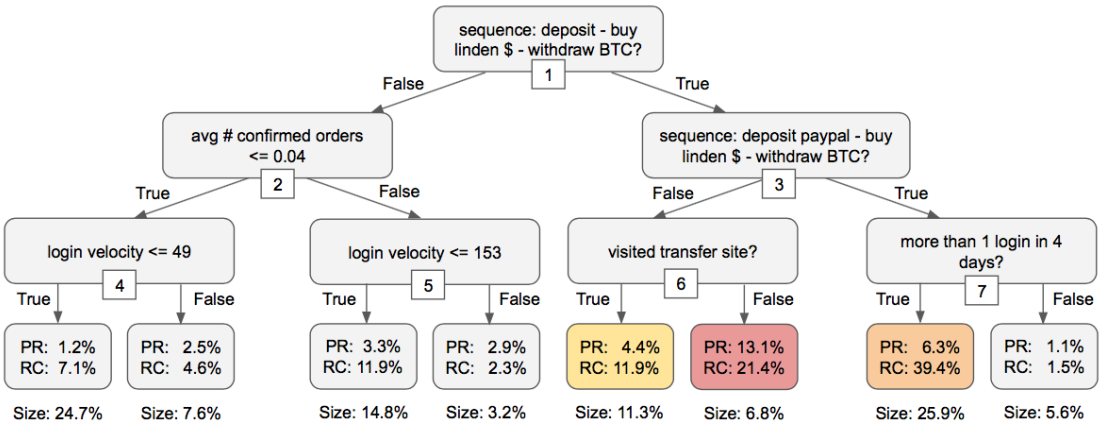

Figure 5.8: **Fraud indicators after multiple logins.** We considered the users belonging to the three leaf nodes with the highest precision as being most likely fraudsters. We label the red leaf node with a precision of 13.1% as high-risk users, the orange leaf node with a precision of 6.3% as medium-risk users and the yellow leaf node with a precision of 4.4% as low-risk users. All other users in the other leaf nodes are considered as regular users.

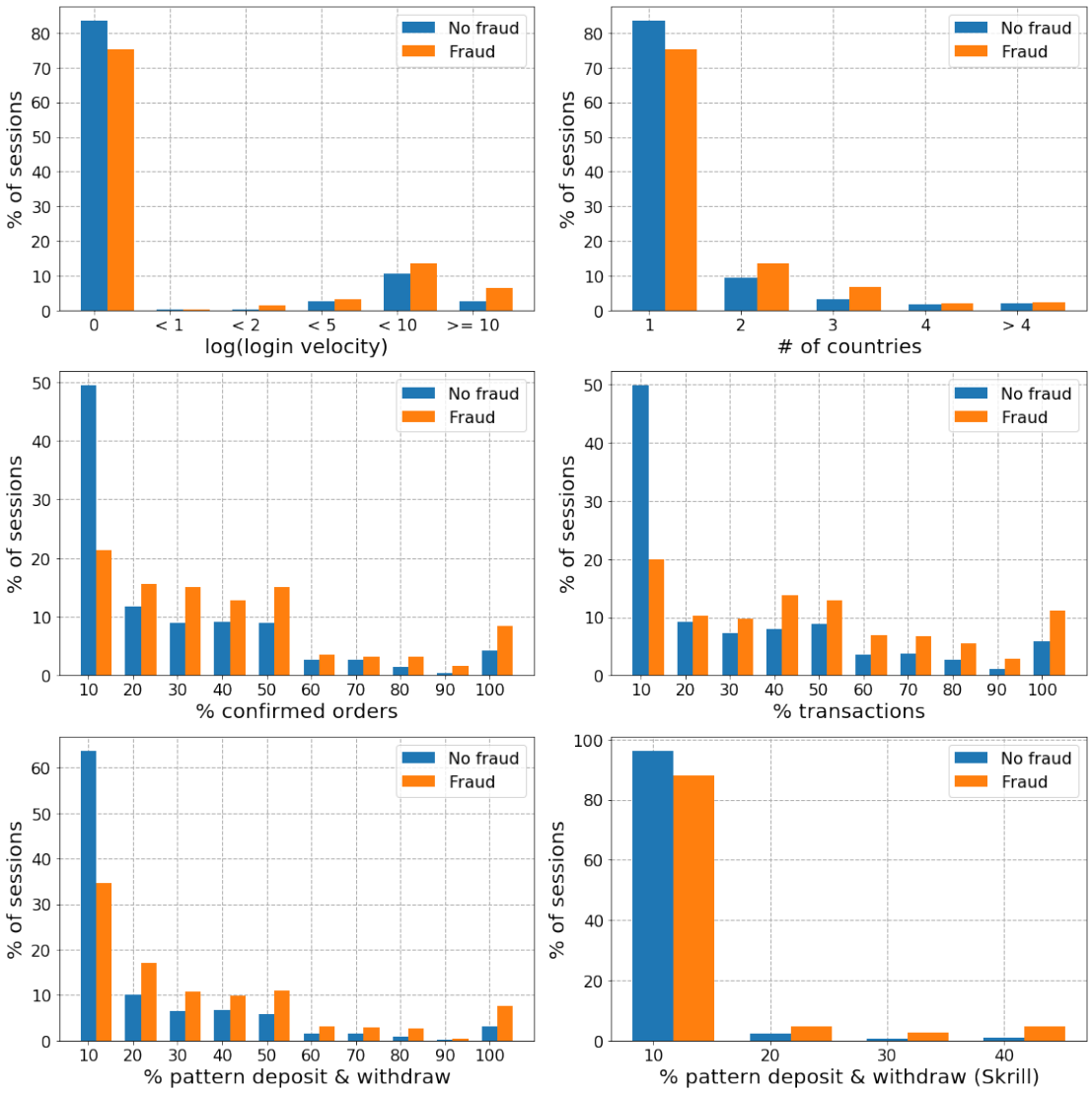

Figure 5.9: **Feature histograms used by HBOS.** Each plot shows the discretized distribution of the features used by the HBOS. In each plot, the x-axis holds the corresponding feature values and the y-axis the fraction of samples (sessions) with the respective feature value. The distribution of the fraudulent sessions is colored in orange and the distribution of the non-fraudulent sessions is blue.

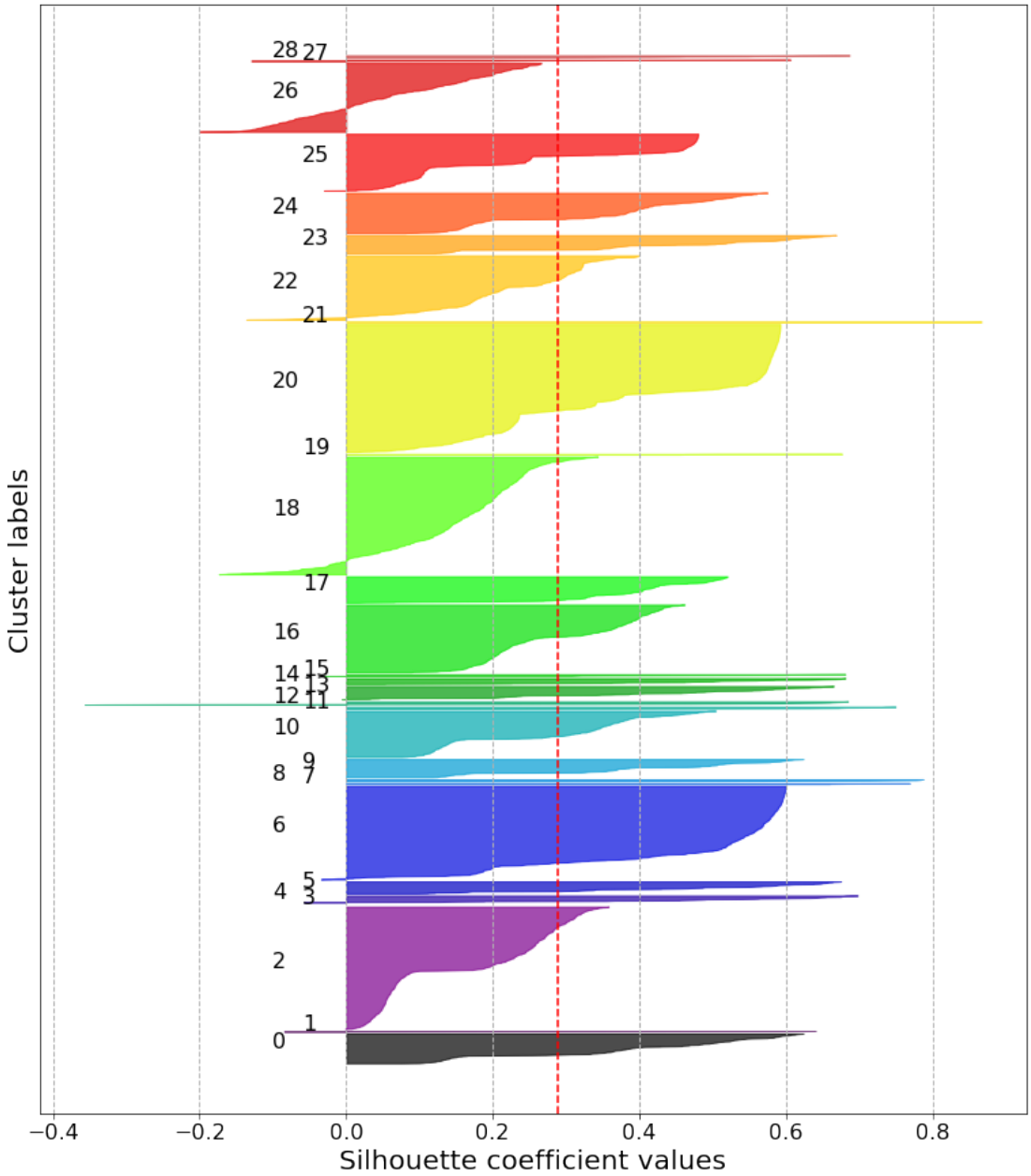

Figure 5.10: **Silhouette coefficients of all clusters.** This plot shows the silhouette coefficients of a random sample with the size of 5000 on the test set. The x-axis indicates the value of the silhouette coefficient and the y-axis represents the clusters to which a sample belongs. Further, all clusters have a separate color. The silhouette coefficient measure has a range of -1 to +1, where a coefficient of +1 indicates that the sample is far away from the neighbouring clusters. A value of zero means that the sample is on the decision boundary between two clusters and a negative value indicates that the sample might have been assigned to the wrong cluster. The red dashed vertical line marks the average silhouette coefficient which is 0.289.

6 Discussion

In this chapter, we discuss the results reported in the previous chapter. This discussion aims to answer the research questions stated in Section 1.2.2. To do so, we compare the results of the different algorithms and analyse the features which contributed the most to the classification. Finally, we evaluate the practical application of the proposed fraud detection system and conclude with our main findings.

6.1 Algorithm Comparison

In Table 5.1 we reported for all algorithms the evaluation metrics described in Section 3.3. To answer the first research question, if it is possible to separate fraudulent from non-fraudulent users by tracking their browsing behaviour on the website, we look at the results of the best performing model. We use the F1-score as an overall performance criterion. Therefore, the random forest achieved the best result with a precision of 16.03% and a recall of 53.35% by predicting 7.22% of the users to be fraudulent. In total 2% of the users were fraudsters, which means that the fraud detection system would have flagged 3.61 times more users as fraudulent as there were actually in the dataset. This high number of false positives results of the trade-off between precision and recall. In the current setup, we aimed for high recall by defining the evaluation metric and therefore the scoring process in a way that it predicts as many samples as fraudulent, as there are left in the dataset. Figure 4.8 illustrates in detail how this scoring process works.

For example, if there are 100 users in the dataset, 10 of them are fraudsters, and each of them makes 10 logins. After the first login, the model would classify the top 10 users with the highest probability of being fraudsters as fraudulent. If all of those ten users are true fraudsters, then the evaluation stops and the model achieves precision and recall of 100% with an RPP of 10%. However, if only 8 of the predictions are correct, the model continues and classifies the top two users with the highest probability of being fraudulent after the second login as fraudsters. If the model is not able to detect the last two fraudsters within these 10 logins, then the RPP increases to $10 + 9 * 2 = 28$ users $= 28\%$. At the same time, the precision drops to $\frac{8}{28} = 28.57\%$ with a recall of $\frac{8}{10} = 80\%$.

To come back to the answer of **RQ1**, compared to a random baseline the random forest achieved a four times higher F1-score and a three times higher recall while keeping the RPP low. Therefore, we conclude that it is indeed possible to separate fraudulent from regular users by analysing their browsing behaviour on the website.

In **RQ2** we evaluated, which algorithms were suitable to perform such a classification task. We evaluated algorithms from both classes of learning algorithms, namely supervised and unsupervised learning. Again, we used the F1-score to decide, which models suite best to perform such a fraud detection task. First, we compared both categories of learning algorithms to each other by calculating the mean F1-score. The logistic regression, decision tree and random forest achieved an average F1-score of 21.77%. We did not consider the limited decision trees in this comparison, due to the restriction we made for the maximum tree depth. The k-Means clustering and HBOS reached an average F1-score of 15.18%. Therefore, we concluded that supervised learning algorithms are more suitable to perform fraud detection on sequential user interaction data than unsupervised ones. However, it might not always be possible to apply supervised learning algorithms since they require labeled data. In this case, k-Means clustering and HBOS still provided a twice as high F1-score than a random baseline.

Next, we compared the individual algorithms to each other. As already mentioned, the random forest achieved the highest F1-score, followed by the decision trees and the logistic regression. The random forest is the most complex and powerful algorithm in our setup and therefore performed best in learning the non-linear relationship in the data. The k-Means clustering and HBOS got the last two ranks. k-Means clustering assumes a spherical distribution of the data around the centres which might result in poor separation capabilities. HBOS, on the other hand, uses histograms to calculate an outlier score. It assigns a high outlier score to samples with rare occurring feature values. In theory, fraud detection can be seen as an outlier detection problem. However, it might not always be the case that the fraudulent samples in the data are outliers which could also lead to poor classification.

We highlighted in Figure 4.3, that within the first ten logins 72% of the incidents occured producing 33% of the overall loss. Therefore, before we move on to the fraud indicators and the feature importance, we take a closer look at how the models perform within the first ten logins. In Figure 5.1 we plotted the accumulated recall over the number of logins according to the evaluation metric, as explained in Figure 4.8. Already after the first login, the random forest achieved a recall of 30% followed by the decision trees and logistic regression. After five logins all models produced their largest recall gain with the random forest reaching a recall of 47%. One can also see that the margin between the top four algorithms increased compared to their performance after the

first login (except for the limited decision trees and the logistic regression). After ten logins the recall reached a plateau and from there on it only rises slightly.

Figure 5.2 shows the accumulated loss over the number of logins. Again, the random forest performed best, closely followed by the decision tree (full depth). After the first login both the RF and the DT performed quite similar and were able to identify fraudsters who produced about 35% of the overall loss. After five logins the RF was able to double the accumulated loss up to 70%, and the DT achieved 65%. The fact that the RF identified fraudsters producing 70% of the loss just after five logins shows that the model is capable of detecting critical fraudulent users early, which might otherwise have stayed undetected for a long time and therefore produce a significant fraction of the loss. After ten logins the DT even surpassed the RF by reaching 80% of the accumulated loss. However, after eleven logins the RF again achieved the best performance and stagnated at about 85% of the overall loss.

Last but not least, we take a closer look at the rate of positive predictions over the number of logins, as plotted in Figure 5.3. As expected, the RF needed the least number of positive predictions, namely 6% after ten logins, to achieve its performance. The decision tree (full depth) and the logistic regression share both the second rank, followed by the limited decision trees, k-Means and HBOS. To sum up, the random forest overall achieved the best performance. After ten logins it detected 50% of the fraudulent users producing 75% of the loss by reporting 6% of the total users to be fraudulent. The simpler decision trees also performed well. The models detected 46% (full depth) and 42% (limited depth) of the fraudsters producing 80% (full depth) and 56% (limited depth) of the loss by also reporting about 6% (full depth) and 7% (limited depth) of the total users as potential fraudsters. Regarding recall, the logistic regression detected on average 5% fewer fraudsters than the decision tree. Both k-Means and HBOS achieved the worst results and are therefore not suited to perform a fraud detection in our setting.

6.2 Feature Importance and Fraud Indicators

In this section, we investigated the importance of individual features, as well as potential fraud indicators to answer **RQ3**. To do so, we first analysed the different feature importance plots provided by scikit-learn of the logistic regression, decision tree and random forest. Further, we discussed the feature histograms selected by HBOS. In the end, we plotted the result of training two decision trees with a maximum depth of three to extract simple rules as fraud indicators.

In Figure 5.4 we plotted the relative feature importance of the logistic regression. High positive values indicate a strong contribution to the class of fraudulent users,

whereas negative values indicate a contribution to the class of non-fraudulent users. The top three positive features are `% transactions`, `% confirm order` and `% deposit`. For example, if a user would visit at each browsing session the transaction, confirm order and deposit site at least once, all these features would reach the maximum value of one, probably resulting in classifying the user as fraudulent. It is reasonable that these features are relevant to the logistic regression because one can argue that fraudsters spend as little time as possible on the website and conduct all necessary steps to deposit money from stolen payment accounts and withdraw it to their Bitcoin wallet within one browsing session. Also, the login velocity has a high relative importance which can be explained through the fact that fraudsters likely use proxies or VPN tunnels to hide their real IP address. For example, if a fraudster uses a proxy server from Russia and one hour later uses a VPN tunnel from the USA, this leads to a high login velocity. The top negative features are `% settings`, `recency` and the `average session duration`. This is also reasonable because regular users tend to change their settings more frequently, use the service more irregular and tend to spend more time per session on the website. Fraudsters on the other hand, merely change their account settings, likely do a login once per day to fully exploit the daily limits and spend less time per session since they only perform the necessary steps to conduct the fraud.

Next, we take a closer look at the feature importance plot produced by the decision tree in Figure 5.5. In contrast to the plot of the logistic regression, the decision tree plot provides no information on whether a given feature contributes to the fraudulent or regular class of users. Instead, it measures the normalized total reduction of the Gini impurity and therefore indicates how well the features are suited to separate the two classes. The feature `% pattern deposit & withdraw` reduced the Gini impurity by far the most, followed by `% pattern deposit & withdraw (PayPal)` and `% deposit`. The first two features measure how often users perform the complete Bitcoin exchange sequence of depositing money, exchanging it to virtual currency followed by the exchange into Bitcoin and then withdraw it back to their wallet. The difference between the first two features is that the latter measures if PayPal was used to deposit money, whereas the first feature does not care about the payment provider. Again, one can argue that in contrast to regular users, fraudsters tend to execute all the necessary steps to perform the fraud within one browsing session which makes this features a reasonable choice.

The feature importance plot of the random forest in Figure 5.6 can be interpreted the same as the one of the decision tree. The only difference is that the reduction in Gini impurity is averaged over all trees in the random forest. In the case of the RF, the most relevant features are `average count confirm order`, `% deposit` and `account age`. The most important feature is `average count confirm order`, and it measures how many orders a user places each browsing session on average. One can see in Table 4.7 that for fraudulent users the median value of this feature is three times higher than

for regular users. As already pointed out previously, fraudsters tend to perform all the necessary steps to conduct the fraud within one session and therefore at least two confirmed orders need to be placed. The first order is needed to exchange the deposited money into Second Life Linden dollars (SLL) and the second one to exchange the SLL back to bitcoins.

The "greedy-add" algorithm to find the best performing features for HBOS selected the six features plotted in Figure 5.9. The first feature `log(login velocity)` *(top left)* describes the average logarithmized login velocity over all successive logins of a user. One can see that 83% of the browsing sessions of regular users have a login velocity of zero (before calculating the logarithm of the login velocity, +1 was added to avoid minus infinity values if the login velocity is zero), whereas only 75% of the fraudulent browsing sessions have a login velocity of zero. For all login velocity values greater than zero the fraction of fraudulent sessions is higher than the fraction of non-fraudulent sessions. This observation supports our hypothesis that fraudsters likely use proxies or VPN tunnels to hide their real IP address. The feature `# of countries` *(top right)* correlates to the login velocity since it describes the number of different countries of which a user logged in. 83% of the regular browsing sessions come from one country, whereas only 75% of the fraudulent sessions come from one country. Same as before, for all feature values comprising more than one country the fraction of fraudulent sessions is higher than the fraction of non-fraudulent sessions. Again, this observation indicates that fraudsters use proxies or VPN tunnels. The next three features `% confirmed orders` *(middle left)*, `% transactions` *(middle right)* and `% pattern deposit & withdraw` *(bottom left)* measure the browsing behaviour on the website and have similar distributions. Those features were also picked by the logistic regression and decision tree as top features. Therefore we refer to the previous part of the discussion for the interpretation of these features. One can see that those features have a power-law distribution in the class of regular users, whereas the sessions of the fraudsters are almost equally distributed. Again, these distributions indicate that fraudsters tend to perform all necessary steps to deposit and withdraw money within one browsing session. The last feature `% pattern deposit & withdraw (Skrill)` *(bottom right)* consists of two power-law distributed classes, where in case of the fraudulent class the tail of the distribution is higher compared to the regular class. Therefore, the difference between the two classes is not as significant as in the previous features. However, one can still spot the difference between the two classes and again argue that fraudsters tend to perform all necessary steps to conduct the fraud within one session.

The feature importance plots and feature histograms already provided a good overview on which features are suited best to predict fraudulent behaviour. However, we wanted to extract some simple rules out of the data which can be understood and applied by humans. To do so, we trained two decision trees with a maximum depth of three.

The first tree was trained using only the first browsing sessions of each user whereas the second tree was trained using the remaining browsing sessions of the users. We chose this setup because some of the features (e.g. the login velocity) can only be calculated after two or more browsing sessions. In Figure 5.7 and 5.8 one can find the resulting decision trees. During the first session, the most reliable indicator of fraudulent behaviour is when the user has placed a confirmed order, visited less than 25 pages and accessed the pages to deposit money, buy Linden dollars, exchange Linden dollars to Bitcoin and withdraw money. Only 3.7% of all users fulfill those three conditions resulting in a precision of 21.4% and a recall of 36.1%. For all other browsing sessions, the second tree suggests that the most reliable indicator of fraudulent behaviour is when the user again visited the pages to deposit money (without PayPal), buy Linden dollars, exchange Linden dollars to Bitcoin and withdraw money and did not visit the transfer money page. 6.8% of all browsing sessions fulfill these criteria resulting in a precision of 13.1% and a recall of 21.4%.

Further, we labeled the users in the top three leaf nodes with the highest precision as high-risk, medium-risk and low-risk users. The new fraud detection system could be operationalized according to the classification process using those two trees. The next section gives a brief explanation on how this could be achieved.

6.3 Practical Application

As an extension to **RQ3** on what kind of behaviour indicates fraudulent behaviour we also focused on developing a transparent and easy to implement fraud detection system. Therefore, we trained two separate decision trees with a maximum depth of three on two subsets of data as described in the previous section. Although the optimized decision tree with a depth of 21 as described in Section 4.3 is theoretically interpretable, practically it is not. A depth of 21 results in $2^{21} = 2.097.152$ leaf nodes, which is impossible for humans to keep track. The benefits of the two smaller trees are that they are still interpretable by humans due to their limited depth while still providing good classification results.

For the operationalization of the two trees, the following two topics need to be addressed. First, one needs to define an infrastructure to collect and process the required data to feed the decision trees. The data processing step includes data preparation, data aggregation and data cleaning as described in Section 4.2. Since the decision trees only consist of seven if-else conditions no specific infrastructure to execute machine learning models is required. Second, one needs to define how to apply the results of the classification to effectively prevent fraud while minimizing the impact of potential false positives. One way to incorporate the results of the decision trees would be to increase

the current Bitcoin payout delay (as described in Section 1.1) for high, medium and low-risk users. Further, additional personal identification steps could be introduced for high-risk users like providing passport or ID scans. These two steps would increase the probability, that the victim or payment provider timely reports the unauthorized payments to VirWoX and complicate the Bitcoin payout process for potential fraudsters, while still being acceptable for a small fraction of incorrectly flagged users.

6.4 Conclusion

To summarize the discussion of the previous sections, we concluded this chapter by answering the stated research questions as well as highlighting our main findings.

RQ1: Is it possible to separate fraudulent from regular users by analysing the browsing behaviour of the users on the website?

As highlighted in Table 5.1 it is indeed possible to separate fraudsters from regular users. Compared to a random baseline the best performing model (random forest) achieved 4.4 times higher precision and three times higher recall while predicting one third fewer users to be fraudsters. Further, the model successfully identified those users who produced about 85% of the overall loss.

RQ2: Which algorithms are suitable to perform such a task, while meeting the stated requirements?

We evaluated five different algorithms, both supervised and unsupervised ones — namely, logistic regression, decision tree, random forest, k-Means clustering and histogram-based outlier score. All algorithms beat the baseline by quite some margin as shown in Figure 5.1. However, the supervised algorithms performed better than the unsupervised ones. Although the random forest achieved the best results, it does not meet the transparency requirement stated in Section 1.2. Therefore, the most suitable algorithm to perform such a classification, while meeting all stated requirements is the decision tree with limited tree depth. Overall it achieved the third best result, while staying interpretable by humans. The logistic regression and the HBOS also fulfill all the stated requirements. However, they performed slightly worse.

RQ3: What kind of behaviour indicates fraudulent behaviour?

There are several different fraud indicators. Depending on the used algorithm the order of their importance varies. However, there is a common pattern throughout all different algorithms. One can see in Figure 5.4, 5.5, 5.7, 5.8 and 5.9 that the feature `% pattern deposit & withdraw`, which measures the total fraction of sessions, where a user executes all necessary steps to deposit, exchange and withdraw money within

the same browsing session is of high importance. This behaviour seems reasonable because most fraudsters exactly know which steps are necessary to conduct the fraud. Therefore, they execute all the steps measured by the feature `% pattern deposit & withdraw` within one session to be as efficient as possible. Node 3 in Figure 5.7 also supports this observation by assigning the users who visit less or equal than 25 pages during their first session a higher probability to be fraudsters than users who visit more than 25 pages. Another reasonable fraud indicator can be extracted from node 7 in Figure 5.8. Due to the daily deposit limits mentioned in Section 1.1, fraudulent users tend to log in at least once per day to fully exhaust those limits. Therefore, node 7 assigns a higher fraud probability to those users who log in more than once within four days.

In this chapter, we discussed the results shown in the previous chapter and answered the stated research questions. To summarize this chapter, we can say that the combination of two individual decision trees with a limited tree depth is the overall best choice for the proposed fraud detection system. Although this model combination did not achieve the highest performance regarding precision and recall, it is superior in terms of interpretability and simplicity. Further, we suggested a method on how to operationalize the two decision trees, to classify the users into high, medium and low-risk users and proposed fraud prevention actions which could be applied to those users.

7 Future Work and Conclusion

This chapter provides a brief outline of future work which could be done to improve the proposed fraud detection system. Further, we summarized the whole thesis and its main findings.

7.1 Future Work

We already discussed two limitations, which we encountered during our research in Section 1.3. Therefore, the next steps to improve the fraud detection system and its predictions would be to address those two limitations.

First, one could use cookie information to improve the mapping between the Apache logs and the customers. Currently, this is done using heuristics as described in Section 4.1, which could result in incorrectly assigned browsing sequences. For example, when two users access the website nearly at the same time using a proxy or VPN server the corresponding Apache logs could be mixed up. This problem could be fixed completely with cookie information.

Second, the lack of timestamps, when a withdrawal took place, forced us to make predictions after each browsing session. However, it would be more efficient only to make predictions when a user withdraws money from the account, because at this point the system needs to decide whether the transaction should be executed or delayed and ask for additional user information. Another benefit of this approach would be that the system potentially collected more valuable user data until it needs to make a prediction.

Besides that, one could also implement a neural network based approach and compare it to the algorithms evaluated in this thesis. Further, an infrastructure to perform A/B tests should be implemented to compare the proposed fraud detection system and the suggested actions to control groups.

7.2 Conclusion

The goal of this thesis was to design, implement and evaluate a new fraud detection system for the VirWoX platform. Currently, VirWoX uses three counter-measures to prevent or limit fraudulent behaviour. Namely, daily and weekly deposit limits, a simple rule-based fingerprinting system to identify multiple accounts and an initial Bitcoin payout delay for up to 48 hours. Despite those three fraud mitigation techniques, many incidents occur on a daily basis and produce a vast financial loss. Therefore, the need for a new data-driven fraud detection system arose.

In this thesis, we laid the groundwork for such a new fraud detection system by analysing the current system deployed at VirWoX, followed by an in-depth analysis of the provided customer data and the implementation and evaluation of five machine learning algorithms to perform fraud detection. According to the stated requirements in Section 1.2, we chose logistic regression, decision trees, random forest, k-Means clustering and HBOS as potential candidates for the new system. We compared the performance of the algorithms to a random baseline and found out that the random forest achieved the best results regarding precision and recall. Since random forests are practically not interpretable, we also trained two small decision trees with a limited tree depth of three. The two simple decision trees achieved overall the third best F1-score after the random forest and the full decision tree with a depth of 21. However, the two simpler decision trees are superior regarding their interpretability and simplicity and therefore the best-suited option for the fraud detection system fulfilling all declared requirements.

Besides the algorithmic evaluation, we explained how to perform all the necessary steps of data cleaning, data preparation and data aggregation. Further, we provided suggestions on the operationalization of this fraud detection system and closed the thesis with an outline for future work to further improve the system.

Bibliography

[1] Geoffrey H Ball and David J Hall. *ISODATA, a novel method of data analysis and pattern classification*. Tech. rep. Stanford research inst Menlo Park CA, 1965 (cit. on p. 20).

[2] Rüdiger W. Brause, Timm Sebastian Langsdorf, and Hans-Michael Hepp. *Credit card fraud detection by adaptive neural data mining*. Tech. rep. 99,7. Informatik, 1999 (cit. on p. 10).

[3] Leo Breiman. "Bagging predictors." In: *Machine learning* 24.2 (1996), pp. 123–140 (cit. on p. 16).

[4] Leo Breiman. "Classification and regression trees." In: (1984) (cit. on p. 14).

[5] Leo Breiman. "Random forests." In: *Machine learning* 45.1 (2001), pp. 5–32 (cit. on p. 16).

[6] A. L. Buczak and E. Guven. "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection." In: *IEEE Communications Surveys Tutorials* 18.2 (2016), pp. 1153–1176. ISSN: 1553-877X. DOI: 10.1109/COMST.2015.2494502 (cit. on p. 10).

[7] Michele Carminati et al. "BankSealer: An Online Banking Fraud Analysis and Decision Support System." In: *ICT Systems Security and Privacy Protection*. Ed. by Nora Cuppens-Boulahia et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 380–394. ISBN: 978-3-642-55415-5 (cit. on pp. 10, 18).

[8] Rich Caruana and Alexandru Niculescu-Mizil. "An Empirical Comparison of Supervised Learning Algorithms." In: *Proceedings of the 23rd International Conference on Machine Learning*. ICML '06. Pittsburgh, Pennsylvania, USA: ACM, 2006, pp. 161–168. ISBN: 1-59593-383-2. DOI: 10.1145/1143844.1143865. URL: http://doi.acm.org/10.1145/1143844.1143865 (cit. on p. 16).

[9] P. García-Teodoro et al. "Anomaly-based network intrusion detection: Techniques, systems and challenges." In: *Computers & Security* 28.1 (2009), pp. 18–28. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2008.08.003. URL: http://www.sciencedirect.com/science/article/pii/S0167404808000692 (cit. on p. 10).

[10] Markus Goldstein and Andreas Dengel. "Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm." In: *KI-2012: Poster and Demo Track* (2012), pp. 59–63 (cit. on pp. 10, 18, 19).

[11] Frank E Grubbs. "Procedures for detecting outlying observations in samples." In: *Technometrics* 11.1 (1969), pp. 1–21 (cit. on p. 9).

[12] M. Gupta et al. "Outlier Detection for Temporal Data: A Survey." In: *IEEE Transactions on Knowledge and Data Engineering* 26.9 (Sept. 2014), pp. 2250–2267. ISSN: 1041-4347. DOI: 10.1109/TKDE.2013.184 (cit. on p. 9).

[13] Victoria Hodge and Jim Austin. "A Survey of Outlier Detection Methodologies." In: *Artificial Intelligence Review* 22.2 (Oct. 2004), pp. 85–126. ISSN: 1573-7462. DOI: 10.1023/B:AIRE.0000045502.10941.a9. URL: https://doi.org/10.1023/B:AIRE.0000045502.10941.a9 (cit. on p. 9).

[14] Nathalie Japkowicz and Shaju Stephen. "The class imbalance problem: A systematic study." In: *Intelligent Data Analysis* (2002), pp. 429–449 (cit. on p. 17).

[15] Steve Lawrence et al. "Neural Network Classification and Prior Class Probabilities." In: *Neural Networks: Tricks of the Trade*. Ed. by Genevieve B. Orr and Klaus-Robert Müller. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 299–313. ISBN: 978-3-540-49430-0. DOI: 10.1007/3-540-49430-8_15. URL: https://doi.org/10.1007/3-540-49430-8_15 (cit. on p. 17).

[16] S. Lloyd. "Least Squares Quantization in PCM." In: *IEEE Trans. Inf. Theor.* 28.2 (Sept. 2006), pp. 129–137. ISSN: 0018-9448. DOI: 10.1109/TIT.1982.1056489. URL: http://dx.doi.org/10.1109/TIT.1982.1056489 (cit. on p. 20).

[17] James MacQueen et al. "Some methods for classification and analysis of multivariate observations." In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA. 1967, pp. 281–297 (cit. on p. 20).

[18] Marina Sokolova and Guy Lapalme. "A systematic analysis of performance measures for classification tasks." In: *Information Processing & Management* 45.4 (2009), pp. 427–437. ISSN: 0306-4573. DOI: https://doi.org/10.1016/j.ipm.2009.03.002. URL: http://www.sciencedirect.com/science/article/pii/S0306457309000259 (cit. on p. 20).

[19] Robert Tibshirani, Guenther Walther, and Trevor Hastie. "Estimating the number of clusters in a data set via the gap statistic." In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63.2 (2001), pp. 411–423 (cit. on p. 20).

[20] Véronique Van Vlasselaer et al. "APATE: A novel approach for automated credit card transaction fraud detection using network-based extensions." In: *Decision Support Systems* 75 (2015), pp. 38–48. ISSN: 0167-9236. DOI: https://doi.org/10.1016/j.dss.2015.04.013. URL: http://www.sciencedirect.com/science/article/pii/S0167923615000846 (cit. on p. 10).

[21] Ji Zhang. "Advancements of outlier detection: A survey." In: *ICST Transactions on Scalable Information Systems* 13.1 (2013), pp. 1–26 (cit. on p. 9).