False alert reduction in fraud detection

Vrijdag, T.S.

Award date:
2017

Technische Universiteit
Eindhoven
University of Technology

Department of Mathematics and Computer Science
Data Mining Group

False alert reduction in fraud detection

Master's Thesis

T.S. Vrijdag

Supervisors:
prof. dr. M. Pechenizkiy TU/e
W. van Ipenburg MSc. Rabobank

Assessment committee:
prof. dr. M. Pechenizkiy DM
dr. R. Medeiros de Carvalho AIS
W. van Ipenburg MSc. Rabobank

Publishing date: 31-08-2017

Eindhoven, August 2017

Rabobank

Abstract

Banks have to detect and prevent fraudulent transactions. Fraud detection is however a very difficult task for several reasons. To name a few, the data is highly *imbalanced* (as there are significantly more genuine transactions than fraudulent), there is *concept drift* (the label or the distribution of data change over time), and there is a limited amount of *resources* (e.g. domain experts). Fraud detection systems often generate more *false alerts* than *true positives* per day in order to achieve a high detection rate. There can for instance be significantly more false alerts. These false alerts can be expensive, because domain experts of the bank have to investigate each alert generated by the detection system.

In this thesis, research is conducted to reduce the number of false alerts of a Machine Learning based detection system of Rabobank while conserving most of the true positives. Several "filters" are proposed and used to achieve this false alert reduction. Moreover, multiple filters are "layered" after each other to discard more false alerts. An "Anomaly Description Mining Framework" is proposed to mine descriptions that have contrasting properties between false alerts and true positives of the detector. Descriptions mined with this framework are then used in false alert reduction. With the experiments described in this thesis, it is shown that several of the described techniques can reduce the number of false alerts and still conserve most of the true positives. On the provided benchmark dataset we shall expect one less detected positive for every reduction of $\sim$4,015 false alerts with one of our techniques. This approach yields less alerts per day such that it is more reasonable for a team of experts to investigate them.

Preface

This thesis is a result of the graduation project for Computer Science and Engineering (CSE) at Eindhoven University of Technology (TU/e). The research of this project is done within the Data Mining Group of the TU/e in collaboration with the Financial Economic Crime (FEC) department of Rabobank located in the Netherlands. The Data Mining Group is part of the department of Mathematics and Computer Science of the university. Within this group, data mining techniques and knowledge discovery approaches are studied. The group has contributed to the areas of predictive analytics, automation of machine learning and network science.

The FEC-department is part of the department of Compliance at Rabobank. The department of Compliance ensures that internal and external regulations (of integrity) are secured in the business structure and processes of Rabobank. The FEC-department contributes to this by managing and preventing incidents, such as fraud and other calamities that damage the integrity of Rabobank. This department consists of several teams, and each team focuses on different aspects.

I would like to thank my supervisor Mykola Pechenizkiy for the opportunity and his guidance during this project. From Rabobank, I would like to thank Werner van Ipenburg and Jan Veldsink for the opportunity and their guidance. I also would like to thank others from the university who collaborate with Rabobank for their research. Especially Simon, Oren and Tita for their valuable feedback during this project. Furthermore, I want to thank my brothers and friends. Especially Marco, Koen and Erik for their support during my study. Finally, I want to thank my parents for their support throughout the years of my study.

Tom Vrijdag
Geldrop
2017

Contents

Acronyms

BDR Bayesian Detection Rate.

BTNR Bayesian True Negative Rate.

FIM Frequent Itemset Mining.

FN False Negative.

FNR False Negative Rate.

FP False Positive.

FPR False Positive Rate.

KYC Know Your Customer. *Glossary:* Know Your Customer.

ML Machine Learning.

RQ Research Question.

SEPA Single Euro Payments Area. *Glossary:* Single Euro Payments Area.

SWIFT Society for Worldwide Interbank Financial Telecommunication. *Glossary:* Society for Worldwide Interbank Financial Telecommunication.

TN True Negative.

TNR True Negative Rate.

TP True Positive.

TPR True Positive Rate.

Glossary

Creditor A person (or institution) to whom is paid via transactions.

Debtor A person (or institution) who performs a payment via a transaction.

Discarded alert An alert of a detection system discarded by a filter. Also known as the predicted negatives of the filters.

False alert An erroneous alert which has been generated by a detection system and which causes unnecessary usage of a company's resources.

Filtered alert An alert of a detection system that is not discarded by a filter. Also known as the predicted positives of the filters.

FP-description Descriptions that have a low support for TPs and a high support difference and which are mined with our "Anomaly Description Mining Framework".

Know Your Customer A process of a business to identify and verify the identity of customers.

Model score The probability that a transaction is fraudulent according to the model.

Single Euro Payments Area SEPA is a payment integration initiative of the European Union for simplification of bank transfers denominated in Euro.

Society for Worldwide Interbank Financial Telecommunication SWIFT provides and manages a network that enables financial institutes worldwide to send and receive information about financial transactions.

TP-description Descriptions that have a low support for FPs and a high negative support difference and which are mined with our "Anomaly Description Mining Framework".

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, an introduction to the project and the setup of this thesis is given. The domain of banking, the process of Know Your Customer, and the process of online banking at Rabobank are described in Section 1.1. The properties of fraud and challenges of detection are described in Section 1.2. The motivation leading to this project is described in Section 1.3. The goal for this project that follows from this motivation is described in Section 1.4. Finally in Section 1.5, a brief summary of this thesis is given.

1.1 Domain of banking

Domain experts have to learn about their customers from large volumes of data that are available. Every day, new relevant data becomes available via internal or external sources, e.g. transaction data or public news. From these data, domain experts make judgements on customers on different aspects, such as geographic risk, transaction risk, PEP-risk[1], and more. This process is also known as Know Your Customer (KYC). The scope of this project focuses primarily on the aspect of transaction risk. The remainder of this section describes the process of online banking at Rabobank.

There are many methods which customers of Rabobank can use to transfer money to other bank accounts via transactions. Each transaction involves many systems of Rabobank and transactions should be as automatic as possible, which makes fraud detection more difficult. With the introduction of online banking, the bank has taken many actions to improve its fraud detection systems.

To understand the process of online banking, we briefly describe a use case of a customer that signs in and transfers money to another bank account via a transaction. In this use case many connected components are involved, a global representation is visualized in Figure 1.1. The bank uses the same end-point of internet banking servers for all online banking scenarios. These internet banking servers handle all sessions of online banking for all platforms, e.g. mobile application[2], or internet webpage.

A customer has to sign in first before he can access his bank account. On the sign-in page, he has to enter his bank account number, card number, and a *signed message* (the login code). Customers can either sign messages by using a `Rabo Reader` or `Rabo Scanner`[3]. For both systems the customer has to authenticate himself by entering his pin-code. With the `Rabo Scanner` the customer has to scan an image presented on the sign-in page, or the he has to enter a number in case the older `Rabo Reader` is used. The EMV-chip[4] of the card then signs the message with its key and both `Rabo`-systems present this *signed message* on screen as a number. The customer then

[1] Politically Exposed Person (PEP): A person with a prominent public function and who has a higher risk for potential involvement in bribery or corruption.

[2] https://www.rabobank.nl/particulieren/apps/rabo-bankieren-app/

[3] https://www.rabobank.nl/particulieren/betalen/rabo-scanner/

[4] EMV stands for Europay, MasterCard, and Visa. It is a technical standard for smart payment cards.

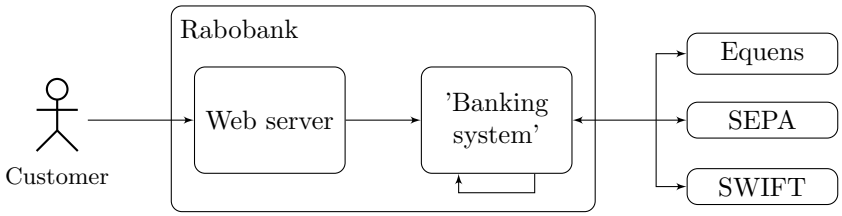


Figure 1.1: Interactions between customer and systems involved in online banking at Rabobank. The customer uses the internet to communicate with the web server of Rabobank. Using this web server, the customer may perform actions which are translated into actions for the other banking systems. For instance, a transaction action is processed by machines that handle transactions. Depending on the nature of this transaction, other systems could be involved as well (Equens, SEPA, SWIFT).

has to enter this number on the sign-in page. The online banking server verifies the signed message for the given bank account and card number. Customers can also use a login-pin in the mobile applications, which replaces the card number and login-code fields of the sign-in authentication.

Once the customer has signed in, he can view his bank accounts and all his transactions (Figure 1.2). The online banking system has to access other banking systems of Rabobank to retrieve this information. Furthermore, the customer can transfer money to other bank accounts via transactions. To proceed with the transaction, the customer again has to *sign a message* with either a `Rabo Scanner` or `Rabo Reader`. This procedure is very similar to the sign-in procedure, but in case of the older `Rabo Reader` the customer has to provide the system with additional information such as the amount of the transaction. The `Rabo Scanner` also displays information about the transaction on its screen to the customer. The internet banking server verifies the *signed message* and it communicates with other banking systems to proceed with the transaction.

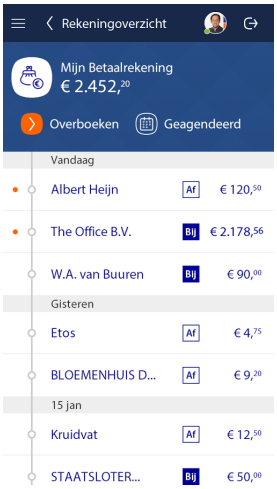


Figure 1.2: Example of transactions in the "Rabobank Bankieren" app (from Play Store)

Which other banking systems are involved depends on the nature of the transaction. For instance, if the creditor of the transaction is within Rabobank, then the transaction can be handled fast and locally by the same system. It can also be the case that the creditor is within another bank located in the Netherlands, then the transaction is handled by systems of a third party (Equens). Similarly for banks within Europe Union (SEPA) and the rest of the world (SWIFT).

1.2 Fraud

Before the problem can be described, we have to define what "fraud" is. From Oxford's dictionary follows that *fraud* can be described as "wrongful or criminal deception intended to result in financial or personal gain"[5]. Someone who commits fraud is called a *fraudster*.

In literature, fraud is distinguished into many types for different kind of domains. In domain of banking, we can distinguish offline and online fraud [Wei et al., 2013]. In case of *offline fraud*, the fraudster could have committed the crime by using a stolen card of the bank, e.g. a credit card. On the other hand we have *online fraud*, where a fraudster commits the crime via web or phone, e.g. by using malware that targets customers who use online banking. Due to the development of technologies during the past years, online fraud in domain of banking is increasing every year. This follows from a report[6] of the European Central Bank.

Customers of banks are victims of financial fraud which results to a large sum of loss. From the same report follows that the total value of fraudulent transaction conducted within SEPA amounted to €1.44 billion in 2013. It is often the case that the crime can only be detected once it has been reported by the customer. However, banks cannot always rely on customers to report the fraud [Dal Pozzolo et al., 2014]. In attempt to detect crimes committed by fraudsters, banks use the large amount of available data to identify patterns and to detect fraud using fraud detection systems. Such fraud detection system then blocks potential fraudulent transactions and generates alerts for these transactions. A domain expert then has to analyze the generated alert and act accordingly. For example, he could contact the owner of the bank account and unblock the transaction if it is actually a genuine transaction.

The transactions that are fraudulent according to domain experts are called the *positives*, which is also considered to be the ground truth. The other transactions are called the *negatives* or genuine transactions. In domain of banking, the data is imbalanced and there are significantly more genuine transactions. This makes fraud detection more difficult. Imbalanced data is not the only reason which makes fraud detection difficult. More challenges of fraud detection are described in Section 2.1. There are several classification problems that are applicable for fraud detection. We consider fraud detection as a binary classification problem, which means that a detector must determine whether a transaction is fraudulent or genuine. As all classification problems, four type of cases can be defined (Table 1.1). With the knowledge of the bank at a given time, these cases are defined as follows.

- The True Negatives (TNs) are negatives for which no alert has been generated. As mentioned before, the data is imbalanced and there are significantly more negatives than positives, and thus the chances are high that there are a lot of TNs.

- True Positives (TPs) are positives which have been detected by the detector. The system only has a brief moment to determine whether an incoming transaction is fraudulent, because the system cannot hinder normal transactions. In normal settings, there are only a few positives compared to the number of negatives. So there can also only be a few TPs compared to the negatives.

- False Negatives (FNs) are positives which are not detected by the detector. The cost of undetected fraudulent transactions can be high for the customers. A customer may notice the fraud themselves and report this to the bank. This feedback can then be used to improve the detection.

- False Positives (FPs) are negatives for which an alert has been generated. In domains of detection, these are also called false alerts. It is difficult to measure the cost of one FP. We assume that the cost of one FP to be equal to one phone call of an employee with the customer [Dal Pozzolo et al., 2014], which is close to €15. In this case, this cost can be lower than the cost for the customer of a missed fraudulent transaction. However, many incorrectly

[5] https://en.oxforddictionaries.com/definition/fraud
[6] Fourth report on card fraud https://www.ecb.europa.eu/pub/pdf/other/4th_card_fraud_report.en.pdf

blocked transactions could also lead to high loss for the customers. A lot FPs can also be expensive for a bank, because these false alerts require the time of domain experts.

Table 1.1: Classification cases in fraud detection. In this table, the rows represent the actual class and the columns represent the predicted class.

	No alert	**Alert**
Genuine	TN	FP
Fraud	FN	TP

In *online settings*, not all data (and knowledge) is available at a given time. Moreover, domain experts can change the actual class of a transaction over time, because more knowledge becomes available. This also changes the classification of the detector for these transactions. Consider the situation of a sequence of transactions for different time windows illustrated in Figure 1.3. Here transactions $T2$, $T3$ and $T4$ are transactions done by a fraudster. At time t_2, the transaction $T2$ is considered to be genuine. At the same time, a detector has a detection delay d_D in terms of sub-seconds to decide whether it could be fraudulent. In this example, the detector decides that the transaction is not fraudulent, which means that this classification is a TN at t_2. This might change after more time has passed and more knowledge becomes available. For instance, a domain expert has been in contact with the account holder, or more transactions were performed and analyzed. Then at time t_4, a domain expert labels this transaction $T2$ as fraud after labeling delay d_L. The detector decided at t_2 that $T2$ is not fraudulent. The classification of the detector changes at t_4 from TN to FN as well. The actual class and the classification of a transaction can thus change over time, because more knowledge of transactions becomes available. Similarly, a decision for a transaction being fraudulent could also be revised after more time has passed. In this project, we consider the problem in *offline settings*, which means that the data is already available. We also assume that labeling delay d_L is immediate such that the actual class cannot change over time.

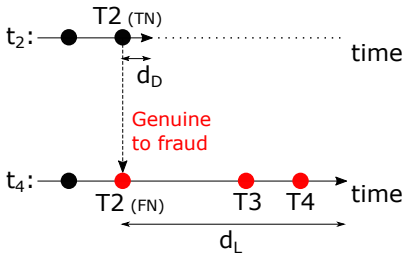


Figure 1.3: Change in actual class of transactions over time for some bank account. The positives are colored red here. At time t_2, transaction $T2$ is considered as a genuine transaction. A detector must be able to determine whether an alert should be generated within detection delay d_D. After more time has passed and more transactions are performed (t_4), the domain expert then considers $T2$ to be fraudulent. This changes the actual class of the transactions. In this example, the label delay d_L is the delay of labeling transaction $T2$.

A perfect fraud detection system has no FPs and no FNs, but this is very difficult to achieve in this domain. It is also difficult to define a cost measure in fraud detection, because it is difficult to measure the costs of FPs and FNs [Dal Pozzolo et al., 2014]. Furthermore, traditional metrics such as *accuracy* (1.1) and *precision* (1.2) do not work well with imbalanced data. This conclusion follows from the equations, because both equations use values from both the negative and positive instances. In this domain, there are only a few positives per a few million transactions each day.

Consider the situation of 5 positives and 15 million transactions. Then if a classifier would classify every transaction as "normal", the accuracy would be higher than 99%. This is obviously not a good or useful detector, because it detects no fraud at all. Now consider a classifier that is able to detect all positives and has one hundred FPs. The precision for this classifier is equal to 0.048, but the classifier could still be considered to be good.

Different techniques are necessary to evaluate a classifier in such settings. The evaluation techniques that are used in this project are discussed in Chapter 3.

$$accuracy = \frac{TP + TN}{total\ population} \quad (1.1) \qquad precision = \frac{TP}{TP + FP} \quad (1.2)$$

1.3 Problem description

The current rule-based fraud detection system of Rabobank generates a few alerts for an average of 15 million transactions per day in real-time. Moreover, the average of online transactions per day is around 1 million. In the rest of this thesis, we assume that 1 million is the number of transactions per day. The process of fraud detection with a rule-based detector is depicted in Figure 1.4. After the rule-based detector has generated an alarm, a small team of domain experts is tasked to analyze these alerts, and they have to handle each alert within a small time window. For each alert, the domain expert has to analyze a lot of attributes which are different for each type of fraud. At the end of the day, only a few or no alerts are true positives. All other alerts are false alerts of the detection system. These types of detection systems require, however, regularly interactions of a domain expert to update definitions of the detection model. For this project, we assume that there is a maximum threshold of 100 alerts per day that the current team of domain experts can handle.

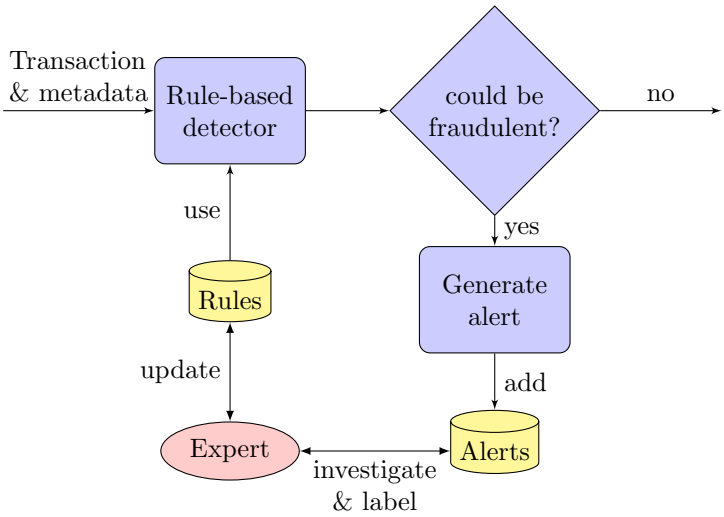


Figure 1.4: Process of fraud detection with a rule-based detector. The detector can generate alerts for transactions that satisfy rules defined by a domain expert. Then a domain expert has to investigate and label these generated alerts.

Whenever the detection system generates an alert for domain experts, it temporarily blocks the transaction. The detector must be able to block a potential fraudulent transaction within a very small window of time, because a transaction should be as automatic as possible and cannot be delayed. In case a detector is not fast enough, the transaction (and thus the money) will be gone. Therefore, the fraud detection system must be able to act fast. After the system has generated an alert, the domain experts investigate the transaction and they may unblock the transaction or decide to keep it blocked for further investigation.

In the past 10 to 20 years, internet banking has grown and some customers have been victims of hackers that have used malware designed to steal money from the customer's bank account. These hacks become difficult to detect as the malware can manipulate information that is visible to the customer and the hackers try to remain undetected. For example, some of the malware could hide the fraudulent transactions in the online banking environment from the customer. The chances that a customer reports this fraud are then much lower. Furthermore, fraudsters could try to "disguise" fraud as genuine transactions. The result of this is that new (small) patterns of these hacks become more difficult to be recognized by domain experts, and therefore, it becomes more difficult to define detection rules. This is where machines become useful as a Machine Learning (ML) based approach may be able to discover patterns that are not recognizable or unknown to the domain experts. These new patterns could then be used to help domain experts to define new detection rules for the rule-based system.

Rabobank also has a ML-based fraud detection system in development which currently operates on a daily basis at the end of the day. However, this detection system generates way more false alerts per day in comparison to the other detection system. The number of false alerts is not stable and it varies per model and per day, but it is often 100 times more than the current rule-based system. The team of domain experts would not be able to handle all of these generated alerts every day. Thus, this ML-based system is not yet ready for a deployment in production. The motivation for this project is formulated as follows.

> **Motivation:** *Rabobank's current ML-based fraud detection system is not suitable for deployment in production due to its fluctuating performance and the too high number of false alerts it generates.*

Running example

To understand the problem better, the following running example is used. Consider a fictional financial bank called AlbertBank. Bob is a customer of AlbertBank and owns one bank account. Unfortunately, his personal computer has been infected by malware developed by fraudster Chuck. Chuck has developed the malware with the intention to steal money from his victims' bank accounts.

Bob uses his personal computer to sign in on the online banking environment of AlbertBank. However, Bob does not know that his computer has been infected with malware of Chuck. This malware tricks Bob into performing the authentication and displays a message that the login was unsuccessful while it was not. Then the malware displays a similar authentication screen, but Bob does not know that this is an authentication to transfer money to a bank account of Chuck. Bob authenticates again and then Chuck is able to transfer money to his account. Bob does not notice this transaction, because the malware is able to filter the fraudulent transaction from the list of transactions on the infected PC.

Luckily, detection systems of AlbertBank have detected this anomaly and these systems have generated an alert for the domain experts of AlbertBank after blocking the transaction. Consider that the rule that has triggered this alert is defined as follows.

$$Amount \geqslant 500 \wedge CreditorIsNew \wedge CreditorCountryIsForeign, \tag{1.3}$$

where *Amount* is the value of the transaction, *CreditorIsNew* and *CreditorCountryIsForeign* are booleans that represent respectively whether the creditor of the transaction is new for Bob's bank account and whether the creditor's bank account is located in another country.

Unfortunately, these detection systems are not very accurate and often also generate more than a few thousands of false alerts. David, one of the few domain experts at AlbertBank, has to analyze this generated alert. In order to discover this fraud, he has to inspect a lot of attributes which are different for each type of fraud. Unfortunately, David does not have enough time to notice the anomaly in the data of the transaction. Thus he unblocks the transaction and Chuck receives Bob's money without Bob noticing.

Let us have a look at one of these many false alerts. Erin also has a bank account at AlbertBank and is currently staying in a different country than the bank. The latest and relevant transactions of Erin with a few attributes are given in Table 1.2. From transaction #501 follows that she has bought travel tickets via online banking. Furthermore, a few purchases have been made in a previously unknown country at 15:11 on 2017-01-22. A domain expert could conclude from this that Erin has bought tickets to travel to country Y. During her stay in this country, she accidentally damaged something valuable to Frank. Erin tries to repay the dealt damage to Frank with transaction #515. However, this transaction is automatically blocked by (the same rule of) the detection system of AlbertBank, because it is a large sum of money transferred to a new banking account in another country.

Table 1.2: The latest transactions of Erin. Here D/C stands for debtor or creditor respectively for a positive or negative amount of Euros. The country column represents the country from where the transaction has been performed. For example, the country where an ATM is located or the country of the IP-address that has been used in online banking. Furthermore, the suffix `new` between parenthesis is used to describe that the debtor or country is new for this bank account. The transaction with the identifier in red (#515) is the blocked legal transaction.

ID	D/C	Date	€	Desc.	Country
501	CheapTickets	2017-01-05 18:41	−100	Tickets 01-22	Z
502	Eve	2017-01-06 11:16	+59	Thank you	Z
511	AirportFood	2017-01-22 09:44	−9	-	Z
512	Market1 `(new)`	2017-01-22 15:11	−20	-	Y `(new)`
513	Market2 `(new)`	2017-01-22 19:14	−20	-	Y
515	Frank `(new)`	2017-01-23 07:01	−1000	Sorry!	Y

1.4 Project goal

Following from the motivation and problem described in previous section, the goal of this project is to improve the output of the ML-based fraud detection system of Rabobank by reducing the number of false alerts. In this project, we develop a "filter" that should discard false alerts generated by the fraud detection system. In this thesis, we use "discarded alerts" and "filtered alerts" to describe respectively the negative and positive predictions of a filter. Unfortunately, discarding false alerts could also lead to discarding true positives which is obviously not desirable, because then there will be no alerts for these fraudulent transactions. Thus it would be valuable to have a good trade-off between reducing FPs and TPs which is satisfiable enough for the bank. The number of FPs should be lower than 100 per day and the True Positive Rate (TPR) should be greater than or equal to 0.2. The project goal is formulated as follows.

> **Project goal:** *Develop and implement a technique that can be used as a filter to automatically discard false positives of the fraud detection system while conserving most of the true positives.*

Note that the goal of this project is to develop a technique that can reduce the number of false alerts automatically. Although this project does not focus on aiding the domain experts, some techniques or results may still be useful to domain experts. For instance, some of the techniques used in this project could also be used to rank the alerts, or provide insights to some characteristics of alerts.

1.5 Thesis summary

In this section a brief summary of this thesis is described. A background of fraud detection and related work of false alert reduction are given in Chapter 2. The research framework of this thesis is described in Chapter 3. It includes a description of the research questions that align with the goal of this project, a description of the evaluation framework for filtering experiments, and a description of the data that is used in the experiments.

A score-based filtering technique is applied to filter alerts (Chapter 4) using the probability that a transaction is fraudulent according to the detector. This filter takes the operational constraints of a maximum number of alerts per day into account. The filtering performance of this score-based filter is quite good, because it discards only a few TPs and it significantly reduces the number of FPs. However, it discards too many TPs for data outside the training period of the detector. This is because the detector is less certain about these cases, which leads to the lower score values.

An "Anomaly Description Mining Framework" is introduced (Chapter 5) to mine descriptions that have contrasting properties between FPs and TPs of the detector. In this thesis, mostly quantitative methods are used to analyze the performance of filters. However, this framework can be used to create visualizations for a qualitative analysis of descriptions. This framework is used to mine descriptions that have a low support for TPs and yet still have a reasonable high support for FPs. Similar descriptions are mined with switched classes.

These mined descriptions are then used to filter alerts in several experiments (Chapter 6). Good performance is achieved by a RandomForest model whose feature space is limited during training by information of the mined descriptions. This RF-based filter discards not as many alerts as the score-based filter of Chapter 4, but it discards a lot of FPs and maintains a high TPR.

The RF-based and score-based filters discard different alerts (Chapter 7). The former discards a lot of FPs that are not discarded by the score-based filter. These alerts thus have relatively high probabilities to be fraudulent according to the detector. A "layered filtering" approach is used to apply the RF-based filter before the score-based filter. The RF-based filter does not only maintain a high True Positive Rate (TPR), but it also discards many of those "high score FPs". The score-based filter is then able to filter different alerts that have lower scores. This "layered filtering" approach improves the performance of the score-based filter as it discards less TPs than without the RF-based filter as the first layer.

Our "layered filter" works quite well in discarding alerts. There are however still other possibilities for future research, which is described in Chapter 8.

Main contributions

In this thesis, we introduce novel techniques of reducing false alerts in domain of online banking, because there are not many papers about false alert reduction in public literature (Section 2.2).

Additionally, we introduce an "Anomaly Description Mining Framework" to mine descriptions that have contrasting properties between FPs and TPs of the detector. In particular, a methodology is introduced to analyze these descriptions with several visualizations.

Chapter 2

Background & Related Work

In this chapter, a background about fraud detection is given. Section 2.1 provides a background about fraud detection from literature. Lastly, related work of false alert reduction is given in Section 2.2.

2.1 Fraud detection

The purpose of this section is solely to provide a background about fraud detection. Detecting fraud is very difficult for several reasons. A few reasons have been described in the introduction of this thesis (Chapter 1). In particular, there are significantly more genuine cases than fraud cases. Another major problem to detect fraud in *online settings* is the phenomenon called *concept drift* [Gama et al., 2014]. Concept drift occurs due to change in data distribution over time in dynamically changing environments. The problem is then that models are less accurate and must be updated as well. Systems that require manual interactions may have problems to keep up. There are several types of concept drift [Gama et al., 2014]:

- Reconsider the situation described in the introduction (Section 1.2) where the actual class changes from genuine to fraud. In this situation a domain expert changes the label of the transaction from genuine to fraud after labeling delay d_L (Figure 1.3). Note that the distribution of the features does not change. This is also called *real concept drift*.
- On the other hand we have *virtual drift*, which happens if the distribution of the features change without affecting the label.

In both cases of concept drift, the models need to be updated in order to detect fraud. Next we describe a few characteristics and challenges of detecting fraud due to concept drift [Wei et al., 2013]:

- Patterns (and fraud patterns) change suddenly or over time. For example, the availability of new products in the market continuously changes. Moreover, once cases of fraud are well detected, the fraudster could change his methods to do fraud to stay undetected.
- Fraud behavior is dynamic, meaning that involved people are actively working on finding exploits of the system. According to Wei et al. malware accounts for the greater part of online banking. From an analysis performed by `securelist.com`[1] follows that many new malware has been discovered during the first three quartiles of 2016. Most of these malware target users who (actively) use online banking. From the same analysis follows that banking malware for mobile has also grown a lot the past year. Hackers exploit Android which had

[1] IT threat evolution Q3 2016 statistics https://securelist.com/files/2016/11/KL_Q3_Malware_Report_ENG.pdf

a world-wide marketshare of 86,6% in 2016Q3[2]. For instance, hackers modify and share trending & popular app packages[3] to distribute malware.

- Fraudsters disguise fraud as normal behavior which makes the decision making of detectors more difficult. Fraudsters often have to do this such that fraudulent activities are accepted by the primary systems of the company. This could then lead to more false alerts. The downside of false alerts is that more resources are required for banks.

We can describe a few more characteristics and challenges of detecting fraud [Wei et al., 2013]:

- There are too many unique cases to pursue, hence it is difficult to capture all of them with human-defined detection rules. A rule-based detector could for instance use rules to identify fraud, which probably leads to many FPs while detecting fraud. Similarly, a rule-based detector could use rules to identify genuine behavior of the system, which leads to a lower number of FPs while the detector also detects probably less fraud.
- Datasets are highly imbalanced, because there is only a handful of "fraud" samples per a few hundred thousand samples. Thus it is difficult to create and validate detection systems. Techniques such as *oversampling*, *undersampling*, and *cost-modifying* may improve the results of some ML methods [Japkowicz and Stephen, 2002].
- In domain of banking, fraud detection must be done in real-time such that a fraudulent transaction can be blocked before it has been completed. This means that a detection system must be very fast (in terms of sub-seconds) such that it is able to process a lot of transactions at a given time. In the introduction of this thesis, this detection delay is illustrated with d_D in Figure 1.3.

Machine learning techniques can address most of these characteristics of fraud, but it is very challenging to develop a single model to tackle all of these aspects [Wei et al., 2013]. Moreover, many machine learning techniques have challenges with reducing the number of false positives for imbalanced data [Japkowicz and Stephen, 2002]. Machine learning, or anomaly/outlier detection in particular, can be used to distinguish "abnormal" samples from "normal" samples [Bolton et al., 2001]. The next section (Section 2.1.1) describes a few techniques of anomaly detection that can be used to detect fraudulent transactions. Lastly, in Section 2.1.2 background about fraud detection in the domain of (online) banking is given.

2.1.1 Anomaly detection

As mentioned before, anomaly detection distinguishes "abnormal" samples from "normal" samples [Bolton et al., 2001]. We define an *anomaly* as an instance (or a pattern) in a dataset that does not conform to the expected behavior. Anomaly detection is also used by intrusion detection systems in the domain of network intrusion. In this domain, an intrusion detection system often uses signature/pattern based or anomaly detection techniques to detect intrusions.

Because there is often a lack of "fraud" labeled data available, generative methods are expected to be more robust for anomaly detection than discriminative methods. Discriminative methods try to optimize a decision rule that classifies data into categories. Such method does not model the relationships between the data and underlying system process. Generative methods on the other hand try to learn a model that describes the system process. A generative approach could be to employ a Hidden Markov Model (HMM) to learn sequential behavior of "normal" behavior [Srivastava et al., 2008]. If a new transaction is not accepted by the trained HMM with sufficiently high probability, then it is considered to be a fraudulent transaction.

Another method could be to use an *autoencoder* [An and Cho, 2015]. An autoencoder is a neural network that is trained to learn reconstructions close to the original input in a smaller hidden space. It consists of an encoder which encodes the input and a decoder which then decodes the encoded input back again (Figure 2.1). For anomaly detection, the autoencoder is trained on

[2]Smartphone OS Market Share https://www.idc.com/promo/smartphone-market-share/os
[3]https://blog.kaspersky.com/pokemon-go-malware/12953/

"normal" instances of the data. Then for new transactions, the model reconstructs the data and measures the reconstruction error, which is the difference between the original and reconstructed data. A transaction is considered to be fraudulent if the measured reconstruction error is higher than some threshold [An and Cho, 2015].

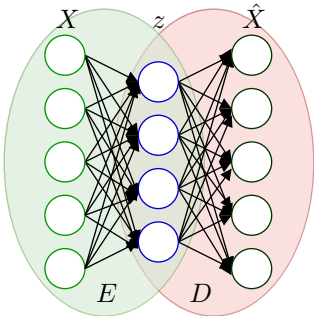


Figure 2.1: Example of an autoencoder. The encoder (E) encodes the input X to z with a smaller dimension, and the decoder (D) decodes the encoded input (z) back again to $\hat{X}$. An autoencoder is trained on "normal" instances of the data. Then for any new transaction, the model reconstructs the data and measures the reconstruction error, which is the difference between the original and reconstructed data. A transaction is considered to be fraudulent if the measured reconstruction error is higher than some threshold.

2.1.2 Fraud detection at banks

In domain of banking, credit card fraud is one of the most studied domains of fraud detection [Abdallah et al., 2016, Bhattacharyya et al., 2011, Bolton et al., 2001, Dal Pozzolo et al., 2014, Patidar et al., 2011, Sánchez et al., 2009, Sarno et al., 2015, Srivastava et al., 2008]. There are a few papers available about fraud detection in the domain of online banking [Kovach and Ruggiero, 2011, Wei et al., 2013]. Furthermore, a single transaction often provides not enough information to detect fraud [Bolton et al., 2001, Dal Pozzolo et al., 2014]. Therefore, the analysis should include aggregated measures [Dal Pozzolo et al., 2014]. In Figure 2.2 examples of two aggregated features for transaction T with different time windows are illustrated. For instance, these features could represent the number of performed transactions for the bank account within the past day or 2 days respectively for feature x and y. Then x is 2, because only transaction $T4$ and $T5$ are within the time window of feature x. Similarly for y, which contains 5 transactions within the time window of feature y.

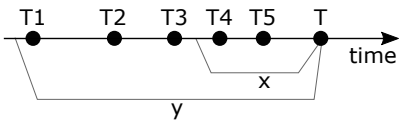


Figure 2.2: Illustration of aggregated features for transactions. Here x and y are aggregated features for transaction T. The time window of feature y covers five transactions. Feature x has a smaller time window than y and only covers the last two transactions.

Domain experts often use their knowledge to select features that can be used to identify fraudulent transactions. Then they use this knowledge to define rules which can then be used by rule-based detection systems [Abdallah et al., 2016]. The downside of such system is that it requires many interactions of experts with domain knowledge to maintain the rules.

Rules can also be discovered using association rule mining techniques. This has been applied to detect credit card fraud [Sánchez et al., 2009]. A hybrid combination of process mining techniques and association rules mining has been used to discover fraudulent behavior from event logs, and

from which then rules are discovered [Sarno et al., 2015]. The precision of their system improves after applying the association rules mining.

Another technique that has often been used are RandomForests [Bhattacharyya et al., 2011, Dal Pozzolo et al., 2014, Wei et al., 2013]. A RandomForest is an ensemble learning technique that constructs a multitude of decision trees at training time [Liaw and Wiener, 2002]. The performance of RandomForests was better than that of a Support Vector Machine (SVM) in a conducted comparative study for fraud detection [Bhattacharyya et al., 2011].

Artificial Neural Networks (ANNs) have been used to detect fraudulent activities [Dal Pozzolo et al., 2014, Patidar et al., 2011, Wei et al., 2013]. The problem of using ANNs is that there are no clear rules of which type and which configuration should be used. In traditional neural networks, the costs of FPs and FNs are treated equally. This has been changed in a detection framework designed for online banking such that the neural network model considers a higher impact of FNs in the training phase [Wei et al., 2013]. Furthermore, genetic algorithms have been combined with ANNs, because of the idea that if a real person is inherently very talented and if he is trained properly, then the chance of success is very high [Patidar et al., 2011]. However, the training phase of this hybrid approach takes way more time, because many generations of networks with small mutations are evaluated. This may be a disadvantage for some applications.

In domain of online banking, a fraud detection system has been designed that uses local and global observations of users' behavior [Kovach and Ruggiero, 2011]. This system uses differential analysis to obtain local evidence of fraud in individual accounts where a significant deviation from normal behavior indicates a potential fraud. Furthermore, the system performs a global analysis to detect whether there is an unusual number of accounts accessed by a single device. Both approaches are then combined in order to determine an overall score that may generate an alert for potential fraud. The disadvantage of this system is that it is difficult to identify devices, because it uses IP addresses for identifications which can change over time.

2.2 False alert reduction

In this section, a literature analysis about related work of false alert reduction is given. Unfortunately, there are not many papers about reducing false alerts in domain of banking. Research of false alert reduction is also done in other domains. Some of the proposed techniques in those domains may also be useful in the domain of banking. For example, one of the other domains is intrusion detection.

The ML-Detector of this project is considered as a "black box". It is therefore necessary to have a false alert reduction technique that is generic, and not specific for a fraud detection technique. In addition to this, false alert reduction techniques should not require resources of domain experts.

This section is structured as follows. Reduction of false alerts in domain of banking is described Section 2.2.1. Finally, in Section 2.2.2 an analysis about reduction techniques in domain of intrusion detection is given.

2.2.1 False alert reduction in banking domain

Unfortunately, there are not many papers available about reducing false alerts in fraud detection in domain of banking. Wei et al. have designed an ensemble framework to detect online banking fraud with imbalanced data [Wei et al., 2013]. This framework consists of three primary components and each component gives a risk score for an incoming event. The final risk score is a weighted combination of the individual's scores. A transaction is then considered to be fraudulent if this risk score is higher than some threshold. Each of these components is designed to perform a certain task. A RandomForest is used to detect the fraudulent transactions, and a neural network with a modified cost-function should reduce the false negatives. This cost-function is modified such that the model considers FNs to have a higher impact in the training phase. The third component employs a contrast pattern mining technique to minimize the false positives.

Contrast pattern mining can be used to identify contrasting itemsets. The definition of *Contrast Patterns* is defined as follows [Wei et al., 2013]. Let D_n and D_f be two datasets respectively of normal and fraudulent transactions, and let $supp(X, D_y)$ denote the support of itemset X in D_y. Furthermore, we define $\omega > 0$ as the *contrast coefficient* and θ as the *minimal detection rate*. Then the Contrast Patterns (CPs) can be defined as follows

$$CPs = \{X | supp(X, D_n) \leqslant \omega \cdot supp(X, D_f) \wedge supp(X, D_f) \geqslant \theta\} \tag{2.1}$$

The *contrast function* that denotes the support difference of an itemset X in the two datasets is defined as follows

$$F(X) = \frac{supp(X, D_f)}{supp(X, D_n)} \tag{2.2}$$

Unfortunately, finding the contrasting patterns is a *NP-hard* complexity problem. Therefore, the authors propose a new algorithm ContrastMiner, which is a modified version of the original Emerging Pattern Mining algorithm [Dong and Li, 1999, Wei et al., 2013]. In this modified version, the high number of found patterns is pruned before they are used by the classifiers. Secondly, the Border-Differ method of the original algorithm has been modified such that it no longer iterates the whole sub space of long itemsets. Lastly, the algorithm does not calculate pattern borders directly, but it uses an alternative approach.

In the proposed framework, the best union of pattern sets is selected. This is divided into parts, and each part is used to generate one model. These models are then used to generate the risk score for the contrast pattern mining component. The authors of the paper have compared results of their framework with that of a rule-based system used in major banks of Australia on real data. From the results followed that the majority of the alerts could be detected by both systems. Moreover, their system had a better accuracy during the tests. Furthermore, the modified version of the contrast mining algorithm is faster than the original. This approach could be combined with existing banking fraud detection systems [Wei et al., 2013]. Unfortunately, the authors have not evaluated the performance of the individual primary components.

Contrast pattern mining could be useful in this project to find contrasting patterns between negatives and positives. However, in this project we will use a different, but similar, technique (Chapter 5).

2.2.2 False alert reduction in other domains

More research about false alert reduction has been done in the domain of intrusion detection. In several papers, researches propose methods that use similarity techniques to filter false alerts [Grill et al., 2017, Spathoulas and Katsikas, 2010]. The authors of [Spathoulas and Katsikas, 2010] introduce and combine three filtering techniques to discard false alerts of a detector. Combining multiple techniques to discard false alerts is also applicable to our project. The "neighboring related alerts" technique is based on the fact that some network attacks involve multiple and similar connections. For example, a *port scan* attack involves many connection attempts as the attacker tries to find active ports of a host. The detection system then probably generates multiple alerts for the same attack. The second technique is similar to the first one, but it uses signatures of alerts instead of neighboring related alerts. The frequency of an alert is then defined as the number of alerts with the same signature for some time window. Then the assumption is made that an alert with a high frequency is more likely to be a TP of the detector. The third technique of this paper is based on the topology of a network with the assumption that some network clients periodically communicate with services of the host and cause the "usual false positives".

The other paper is a more recent paper. In this paper, the authors propose the local adaptive multivariate smoothing (LAMS) method to replace the output of the detector by an average score of similar events observed in the past [Grill et al., 2017]. The smoothing method effectively smooths the output of the detector and therefore reduces short-term noise events, which is a kind of false positives according to the authors. From the results with different intrusion detection systems follows that this method increases the AUC in most experiments.

These post-processing similarity techniques are not directly applicable to the domain of online banking, because of the differences in the domains. Moreover, the categories of FPs are most likely to be different. However, techniques that use information of previously observed similar events could still be applied.

Active learning with a domain expert in the loop has been used to improve the performance of intrusion detection systems [Pietraszek, 2004, Pietraszek and Tanner, 2005]. Such technique can improve the performance of the detection system. But it does require resources of a domain expert, and therefore it is not applicable to our project.

Chapter 3

Research Framework

In this chapter, the research framework of this thesis is described and is structured as follows. The research questions that align with the goal of our project and the research methodology used in this thesis are described in Section 3.1. Each filtering experiment is evaluated with the evaluation framework as described in Section 3.2. A brief exploration of the transactions dataset is described in Section 3.3. Finally, the process of generating the alerts that are used in the remainder of this thesis is described in Section 3.4.

3.1 Research questions & methodology

Our research questions and research methodology are described in this section. The goal of this project is to develop a technique that can be used as a filter to automatically discard FPs of the ML-Detector while conserving most of the true positives. In addition to this goal, the following research questions (RQ) are defined:

RQ 1 How do score values of the detector's model relate to the label, and how can this be used for automatic filtering? (Chapter 4)

RQ 2 Can feature descriptions distinguish FPs from TPs, and how well do they distinguish the classes? (Chapter 5)

RQ 3 How can these descriptions be used with automatic filtering, or improve other techniques? (Chapter 6)

RQ 4 Do filters discard different alerts, and does a combination improve the performance? (Chapter 7)

Each of these research questions can also be divided into smaller sub-questions. These sub-questions are listed in the chapter that covers the research question.

Four main research methodologies of this thesis are listed below. Note that they do not directly align to the structure of the thesis or the order of the research questions.

Analysis of transaction and alert data Properties of the large transaction dataset are analyzed and described in Chapter 3. It covers a small experiment of machine learning that uses all features and which results with a very small model that can predict fraud too well. Feature selection is then used to prevent this issue in our experiments. Properties of alerts are investigated in Chapter 5. Here biclustering and frequent itemset mining are applied to find local patterns in alerts that can distinguish the negatives from positives (RQ 2).

Alert model score values Model score values (predictions probabilities) for alerts are analyzed and used in reduction in Chapter 4 (RQ 1). The analysis includes an investigation of whether the model of the detector is a good ranker. Alerts that have highest scores are then filtered. This reduction technique respects operational constraints such that the number of alerts do not exceed any day.

Contrasting descriptions Descriptions with contrasting properties between FPs and TPs are mined with Frequent Itemset Mining and analyzed in Chapter 5 (RQ 2). The analysis includes analyzing characteristics of the descriptions with several heatmaps. These descriptions are then used to discard alerts in Chapter 6 (RQ 3). Descriptions are used to directly discard alerts, to limit the feature space during training of models, and to update the probability that a transaction is fraudulent.

Layered filtering Multiple filters are used as "layers" such that they together improve the performance of filtering alerts in Chapter 7 (RQ 4). Then each layer operates on a smaller subset of data for which the base rate of fraudulent transactions increases.

3.2 Evaluation framework

The evaluation framework of our filtering experiments is described in this section. The first part of this section describes a few definitions used in this thesis followed by a few measures that are used in the evaluation. The procedures of this evaluation are described in the final part of this section.

3.2.1 Definitions

Events Assume that an event F means that a transaction is fraudulent, and then we have that $\neg F$ is a genuine transaction. Furthermore, assume that event A represents an alarm for a transaction from the detector and thus $\neg A$ represents no alarm. Then the prediction of a detector that a transaction is fraudulent can be denoted by $P(F|X_i)$, where X_i represents a transaction. The rates can also be expressed as probabilities in terms of F and A. The False Positive Rate (FPR) can be expressed as $P(A|\neg F)$ and the TPR by $P(A|F)$. Similarly, we can express True Negative Rate (TNR) as $P(\neg A|\neg F) = 1 - FPR$ and False Negative Rate (FNR) as $P(\neg A|F) = 1 - TPR$.

3.2.2 Measures

Difference in FPs and TPs It is desired to know the difference in FPs and TPs after filtering the alerts. These differences are computed as described by (3.1) and (3.2), where the subscripts D and F denote respectively whether it is a measure of the detector or a filter that extends the detector. Note that this difference can only be negative or zero.

$$\Delta\text{FP} = \text{FP}_F - \text{FP}_D \tag{3.1}$$

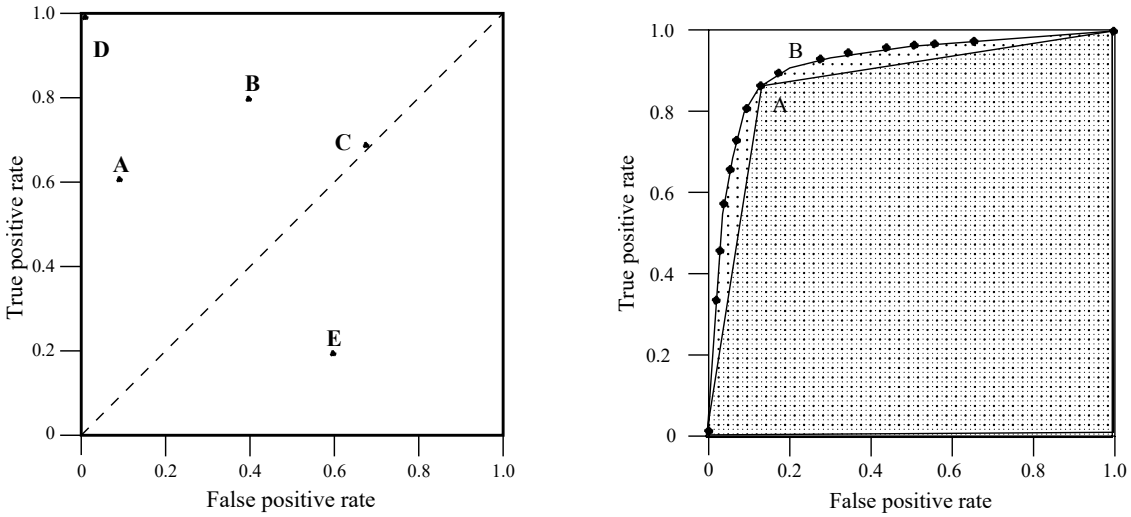
$$\Delta\text{TP} = \text{TP}_F - \text{TP}_D \tag{3.2}$$

New FPR and TPR The new FPR and TPR after filtering are computed as described by (3.3) and (3.4). Note that the denominator of these equations use the total number of negatives or positives of the detector. These equations yield rates as if the detector and the filter together form a single model.

$$\text{FPR}_{D+F} = \frac{\text{FP}_F}{\text{TN}_D + \text{FP}_D} \tag{3.3}$$

$$\text{TPR}_{D+F} = \frac{\text{TP}_F}{\text{TP}_D + \text{FN}_D} \tag{3.4}$$

ROC graph & TPR/FPR tradeoff A Receiver Operating Characteristic (ROC) graph is a plot that can be used to illustrate the performance of classifiers in domains with imbalanced data [Fawcett, 2006, Phua et al., 2004]. The X-axis of the plot represents the FPR and the Y-axis represents the TPR (Figure 3.1). These metrics do not depend on class distributions, because each dimension of the graph has a strict ratio. Therefore, the plot does not change when class distribution of the data changes [Fawcett, 2006]. ROC graphs are good to compare the performance of classifiers.



(a) Five discrete classifiers

(b) AUC for two classifiers

Figure 3.1: Two examples of ROC graphs (from [Fawcett, 2006]). The X-axis represents the FPR and the Y-axis represents the TPR. Figure 3.1a visualizes 5 discrete classifiers and a dashed diagonal from $(0, 0)$ to $(1, 1)$ which represents the random guesses. Classifier D represents the optimal classifier with $TPR = 1$ and $FPR = 0$. In Figure 3.1b the area under curve (AUC) is visualized for a binary classifier (A) and a scoring classifier (B).

Area Under Curve (AUC) Classifiers are easier to compare with one value. The Area Under Curve (AUC) is a common method to calculate the area under the ROC curve and transforms the ROC performance to a single scalar value [Fawcett, 2006]. The AUC is a portion of a unit square (Figure 3.1a), and therefore always in the range between 0 and 1.0. Note that the diagonal line that represents random guesses has an area of 0.5, and thus a realistic classifier should have more than 0.5 [Fawcett, 2006]. In Figure 3.1b the AUC is visualized of a binary classifier (A) and a scoring classifier (B). In our case, we want to focus on low FPRs, and it therefore makes sense to compute the partial AUC (or pAUC) of the region with low FPR [McClish, 1989].

3.2.3 Procedures

Each filtering experiment is conducted and evaluated as described in this section. The first part of this section describes how alerts are filtered. The final two parts of this section describe the measures and the ROC graph comparison that are used.

Filtering Consider the performance of a fictional detector shown in Figure 3.2. Assume that instance A of this detector has a prediction threshold δ_A. We consider three different approaches that a "filter" can use to improve the performance of this instance of the detector. These three approaches are illustrated with arrows x, y and z, and are described as follows:

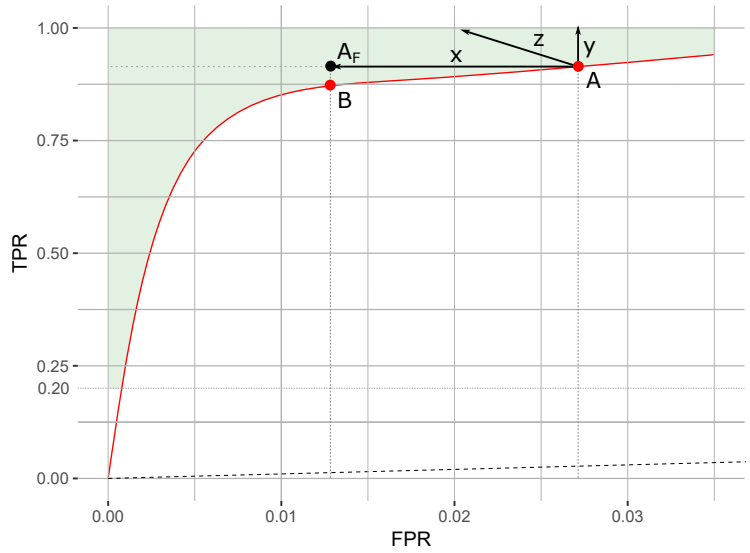


Figure 3.2: Example of ROC comparison. The X-axis represents the FPR and the Y-axis represents the TPR. The X-axis is cutoff at 0.035 to visualize the performance for smaller FPRs. The red curve represents the performance of a fictional detector. The green area above the red curve is considered to be an improvement of the detector. Arrows x, y and z represent different approaches for filtering to improve the performance of instance A of the detector.

- Approach x tries to decrease the FPR of A. This could be achieved by updating the predictions of A such that the negatives have lower predictions. A filter then operates on the predicted positives of A with the objective to decrease the number of FPs. This could however also decrease the number of TPs.
- Approach y tries to increase the TPR of A. This could be achieved by updating the predictions of A such that the positives have higher predictions. A filter then operates on the predicted negatives of A with the objective to increase the number of TPs. This could however also increase the number of FPs.
- Approach z is a mixed approach of approaches x and z.

Each filtering approach yields a new instance A_F with updated predictions. In this project, filtering approach x will be used, since it is our goal to decrease the number of FPs of the detector. Let B be an instance of the detector with prediction threshold $\delta_A \leqslant \delta_B \leqslant 1$ and the same FPR as filter instance A_F. The filter instance then needs to have a higher TPR than B such that it is an improvement of the detector.

Our first filtering approach uses the prediction probability of the detector such that the number of alerts per day does not exceed a threshold (Chapter 4). It is a simple approach to decrease the number of FPs, but note that it is different than another instance of the detector that has a higher prediction threshold. This score-based filter could be compared by an instance that has a continuously changing prediction threshold.

The general setup of filtering alerts is shown in Figure 3.3. Training and test sets are created from the large transactions dataset (Section 3.3). The training set is then used to train a model of the detector (Section 3.4). This new model is applied on all transactions to output a new dataset "alerts" (Section 3.4). This alerts dataset is the base of the filtering experiments.

Although the filter is a separate model, we consider the detector and the filter as one single model such that performance of a filter can be compared with the detector. For each alert, the filter updates the prediction of the corresponding transaction being fraudulent. The new prediction of

discarded alerts is 0 and similarly, the new prediction of filtered alerts is 1. Note that predictions of the predicted negatives of the detector are not changed.

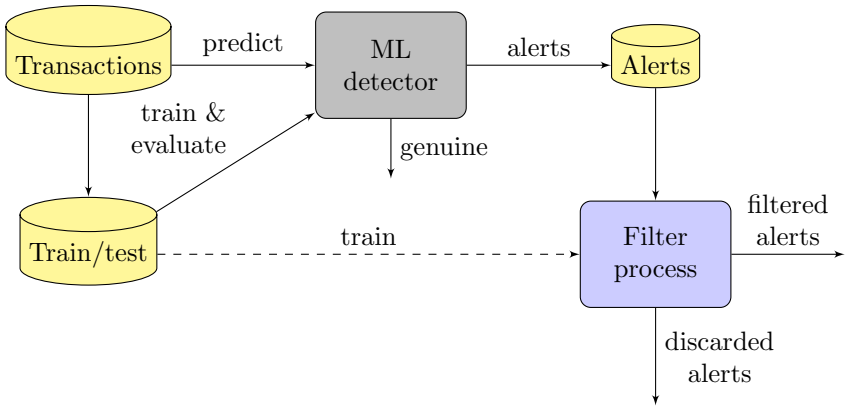


Figure 3.3: General setup of the process of filtering alerts. Here the dashed line represents an optional data flow, because not every "Filter process" requires to be trained. The "ML detector" is considered to be a "black box" and stores all alerts after execution. A "Filter process" then filters alerts for fraudulent transactions. In this thesis, the "Filter process" is different for each experiment.

Measures A quantitative analysis is performed to evaluate several measures of a filter. The following values are computed for each month-subset of the data (Section 3.3) and for the all data.

- The difference in FPs and TPs as defined by respectively (3.1) and (3.2).
- The new FPR and TPR as defined by respectively (3.3) and (3.4).
- The partial AUC (pAUC) for FPR $\leqslant 0.035$. This FPR threshold follows from Table 3.2 of the last section of this chapter.

ROC graph comparison The predictions for all months are used to visualize the performance of the detector and filters in a ROC graph. The same FPR threshold of 0.035 as for pAUC is used to visualize the graphs for smaller FPRs. An example of a ROC graph comparison is illustrated in Figure 3.4. In this figure, the red curve represents the performance of a fictional detector and the two black curves are different filters that extend this detector. The green area above the red curve is considered to be an improvement of the detector. In this example, filter A improves the performance as it achieves a higher TPR than the detector at the same FPR. Filter B on the other hand achieves a worse performance than the detector. One may argue that the filter should be a point in the ROC plane, but a curve is used because the filter is an "extension" of the detector.

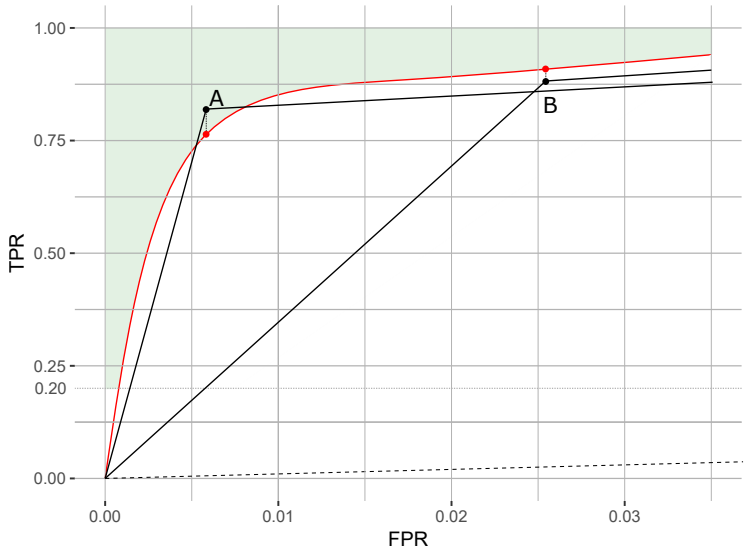


Figure 3.4: Example of ROC comparison. The X-axis represents the FPR and the Y-axis represents the TPR. The X-axis is cutoff at 0.035 to visualize the performance for smaller FPRs. The red curve represents the performance of a fictional detector and the two black curves are different filters that extend this detector. The green area above the red curve is considered to be an improvement of the detector. Filter A improves the performance as it achieves a higher TPR than the detector at the same FPR. Filter B on the other hand achieves a worse performance than the detector.

3.3 Transactions lab dataset

In this section, the dataset of transactions that will be used in this project is described. This dataset consists of real online transactions labeled by domain experts of Rabobank. As stated in the introduction (Section 1.2) we assume that these labels are labeled correctly. All transactions labeled as fraud are thus confirmed fraud. The transactions of the large remainder of the data are either confirmed legal or assumed to be genuine.

The bank has already pre-processed the raw data of online transactions and created a new large lab dataset during the start of this project. In this large lab dataset, names of features have been anonymized and their values have been binned. This binning process is not only applied to anonymize the values, but it also enables usage of algorithms that do not handle categorical values natively. Therefore, all feature values are in a numeric form, while there are some features that are actually categorical or nominal. In addition to these features, each sample also has a date of the transaction, an identification hash, and a binary label. Downsampling of genuine transactions has been used to decrease the very high number of genuine transactions. Different sample ratios have been used for January 2016 up to August 2016 and the remainder of the dataset. The performance of a model can be projected to real settings by the bank.

In an early stage of this project, an experiment has been performed on an older version of this transaction dataset. From this experiment followed that there are features that have good predictive properties to detect the fraudulent transactions. This can also be concluded from a single decision tree learned from the same data (Figure 3.5). It can be observed that the decisions made by this tree only includes two features ($X[685]$ and $X[429]$). These two *too good to be true* features have high predictive properties, and thus such features should not be used in our experiments. Feature selection is used to disregard such features.

In Table 3.1, an overview of sizes of the transaction data is given. The complete dataset that will be used in this project consists of a lot of features and 30 million samples in a time window from January 1st 2016 up to February 28th 2017 (14 months). A selection of features is made to prevent issues with the previously mentioned too good to be true features. This gives us a total

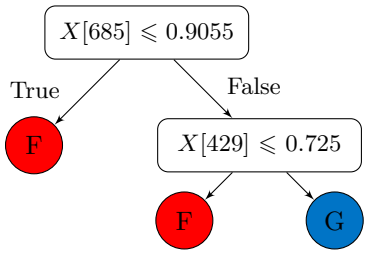


Figure 3.5: Simple decision tree trained with all features. This decision tree only uses two features to make a decision whether it is fraud or genuine. These *too good to be true* features have high predictive properties and should be excluded from experiments.

Table 3.1: Overview of sizes of the online transaction lab dataset. *Selected features* are the features left after excluding features with high predictive properties. This dataset has way more samples of genuine transactions than fraud cases.

Property	Count
Selected features	1013
Confirmed fraud	∼2000
Samples (all)	30 million

of 1013 features. There are thus a lot of features in the dataset that we will be using, and thus we will not engineer new features during this project. In the next section, the process of generating alerts is described.

3.4 Detector alerts

As described in previous section, the large lab dataset has been created by the bank at the start of this project. Thus small modifications of the software of the ML-based detection system have been performed such that it will work with this anonymized dataset. Then a model has been trained with a training set created from lab dataset of samples from January 2016 up to January 2017. This model is then used to generate alerts from all transactions for each day in January 2016 up to February 2017. Note that the "generated alerts" consists only of the predicted positive class, i.e. the FPs and TPs. From now on, we refer to the generated alerts with "alerts".

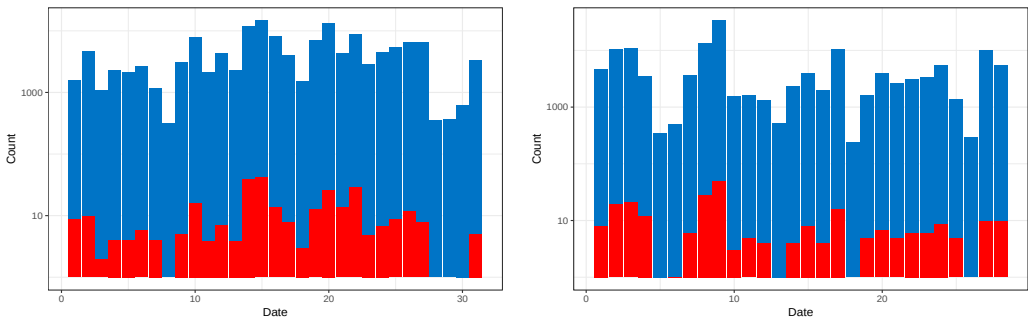
Instead of using a single model, a more reasonable (and realistic) approach would be to use a "streaming approach". Meaning, to split the data into chunks of smaller intervals, e.g. weeks. Then for each interval, we could train a model with data upto this interval and use this model on all historic data and on some of the next intervals. We then expect that new fraudulent cases appear over time. Moreover, newer models should be better in detecting these new cases. Unfortunately, training and applying a model takes quite some time. To simplify the problem, we only consider one model with a static prediction threshold and one interval in the experiments in this project.

In Table 3.2 an overview of the alerts per month is given. From this table follows that there are too many alerts on average per day. Moreover, the FPR is too high for each month. The detector has a very high TPR on average for all months. Most of the FPRs are below 0.035 and therefore we use this as the threshold to compute the pAUC and to visualize ROC graphs (Section 3.2).

The same conclusion can be made from the plots depicted in Figure 3.6. From these figures also follow that there are less alerts on weekends than on regular working days. This is because there are usually less transactions during the weekends. From these figures also follow that there are sometimes more TPs than on other days. The number of FPs of the detector's model is thus too high, as has been formulated in the problem of this thesis (Section 1.3). The remainder of this thesis describes a few methodologies to reduce this high number of FPs.

Table 3.2: Overview of the alerts per month. This table provides an overview of statistics of the generated alerts per month. The fourth column represents the mean (and standard deviation) per day of alerts. The average numbers of alerts per day are too high and exceeds the banks capacity, but the model achieves a very high TPR for all months. One may say that the values of FPRs are low. But recall that there are a lot of genuine transactions (Table 3.1), and therefore the low values of FPRs are still too high. The partial AUC (pAUC) is computed for FPR $\leqslant 0.035$.

Month	Total TP	Total FP	Mean alerts (sd)	FPR	TPR	pAUC
January 2016	51	33,657	1087.4 (531.7)	0.0112	0.9808	0.0342
February 2016	47	83,221	2871.3 (774.8)	0.0284	0.9592	0.0321
March 2016	93	112,702	3638.5 (1368.7)	0.0338	0.9894	0.0340
April 2016	126	129,163	4309.6 (1115.0)	0.0343	0.9921	0.0340
May 2016	144	145,964	4713.2 (1303.4)	0.0357	0.9600	0.0327
June 2016	165	141,350	4717.2 (1352.2)	0.0356	0.9706	0.0320
July 2016	83	136,628	4410.0 (1167.4)	0.0344	1.0000	0.0339
August 2016	100	55,563	1795.6 (714.0)	0.0339	0.9709	0.0320
September 2016	47	6,450	216.6 (88.0)	0.0194	0.9216	0.0321
October 2016	110	15,466	502.5 (190.0)	0.0255	0.9821	0.0334
November 2016	165	15,952	537.2 (161.4)	0.0237	0.9880	0.0340
December 2016	173	17,213	560.8 (190.4)	0.0236	0.9774	0.0341
January 2017	311	15,241	501.7 (136.1)	0.0225	0.9936	0.0346
February 2017	254	13,872	504.5 (162.6)	0.0223	0.8789	0.0277
All months	1,869	922,442	2180.0 (1958.0)	0.0304	0.9649	0.0325



(a) January 2017 (b) February 2017

Figure 3.6: Number of alerts in 2017 per day. The X-axis represents the date and Y-axis represents the number of alerts with a logarithmic scale. The colors red and blue are used to represent fraudulent and genuine cases respectively. Moreover, there are less alerts during weekends. Larger version of these images are depicted in Figure A.1 of the appendix.

Chapter 4

False Alert Reduction With Model Scores

In this chapter, an experiment is conducted to reduce false alerts by solely using the probability that a given transaction is fraudulent according to the model of the detector. We refer to this probability as the "**model score**" of an alert. This score-based filter also takes operational constraints into account. In particular, the maximum number of alerts that can be handled by domain experts per day. We have assumed that this maximum is 100 (Section 1.3). It is a simple approach to decrease the number of FPs, but note that it is different than another instance of the detector that has a higher prediction threshold. This approach could be compared by an instance that has a continuously changing prediction threshold.

The research objective of this chapter corresponds to RQ 1 and is as follows.

- We want to discard alerts by solely using the model scores such that the number of alerts satisfy the operational constraints.

Moreover, for this chapter we consider the following sub-questions of RQ 1.

RQ 1.1 How are these score values distributed per class per month, and is it possible to filter FPs with a score-based filter technique?

RQ 1.2 How well can alerts be filtered using a maximum alert score-based filter technique for alerts inside and outside the training period?

RQ 1.3 How does this compare to the performance of the detector without filtering?

This chapter is structured as follows. The model score for predicted positives is analyzed in Section 4.1. Filtering techniques that are used in the experiment of this chapter are described in Section 4.2, and the experiment itself is described in Section 4.3. In Section 4.4 results of this experiment are described. And finally, a brief conclusion of this chapter is given in Section 4.5.

4.1 Model score

Recall from Chapter 3 that detection systems provide a risk score or the probability that a transaction is fraudulent. An alert is then generated by the detector if and only if the score of the transaction is greater than or equal to some threshold $0 \leqslant \delta \leqslant 1$. We denote this probability score or model score as $P(F|X_i)$ where X_i represents a transaction and event F represents whether this transaction is really fraudulent. A higher score for a transaction then means that it is more likely to be fraudulent.

We want to know how these score values are distributed per class and per month (RQ 1.1). Values of fraudulent transactions could be less accurate for transactions outside the model's training window. To investigate this, we have a look at the boxplots illustrated in Figure 4.1. In this

figure, boxplots are illustrated for the distribution of scores per class for different months. The score value of each TP is also projected onto the figure. We have not done this for the FPs, because the result would be a filled black rectangle from top to bottom. Recall that the training period of the detector was from January 2016 up to January 2017 (Section 3.4), and thus the last month of this figure (February 2017) lies outside the training period. Furthermore, from Table 3.2 followed that for most months there are more than 100 TPs.

We can make a few conclusions from Figure 4.1. First of all, the scores of FPs are in the range from the detection threshold (0.5 in this case) up to 1.0. Moreover, more than 75% (all below the top of the rectangle) of the FPs have scores lower than 0.8 for each month. However, there are still a lot of FPs that have scores greater than 0.8. This is slightly less than 25% (above the rectangle) for each month.

Secondly, for all months in the training period, the scores of the majority of the TPs are greater than 0.9. For these same months, some TP outliers can be observed with lower scores.

Finally, the scores of the TPs are much lower during February. This can be concluded from the fact that score values of the TPs are more distributed in February, and only slightly more than 75% of the TPs have scores greater than 0.8. This is because the model of the detector has been trained on data before this month. Thus during this month, there are unknown fraudulent transactions. The scores for the TPs in this month are thus less reliable than for the other months.

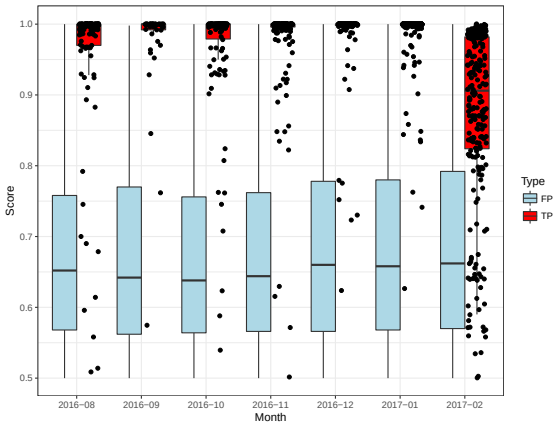


Figure 4.1: Distribution of scores per class for different months. The scores of more than 75% FPs are lower than 0.8 for each month, and there are still slightly less than 25% of the FPs with scores greater than 0.8. Most of the scores of the TPs before February are greater than 0.9, and there are also some outliers with lower scores during these months. However, the scores of TPs in February are much lower. This will make a score-based filtering technique more difficult.

From this we can conclude that the model of the detector is good ranker, because the TPs have high model scores and most of the FPs have lower values. There are however still many FPs with higher values. This means that a score-based filtering technique would be able to discard many of the FPs. However, this could also lead to discarding TPs that have lower scores, and especially outside the time window on which the detector has been trained. Scores of positives observed after this training period are lower, and thus a score based filtering technique can be less effective there.

4.2 Filters

In this section, two types of filters are described. The first type of filter is a score-based filtering technique that filters at most k alerts per day that have the highest score values. The other filter filters at most k alerts randomly.

4.2.1 "Score top" filter

As described earlier, the model score of a transaction is denoted by $P(F|X_i)$ where X_i represents a transaction and event F represents whether this transaction is really fraudulent. Furthermore, an alert is then generated by the detector if and only if the score of the transaction is greater than or equal to some threshold $0 \leqslant \delta \leqslant 1$. We have thus that $P(F|X_i) \geqslant \delta$ for all generated alerts, because an alert is only generated if the score is greater than or equal to the threshold δ.

As mentioned before, from the perspective of banks there is a limit of available resources and thus a bank can only handle a certain amount of alerts per day. So a naive approach to filter the alerts would be to sort (rank) the alerts such that the first alert has the highest probability to be fraudulent according to the detector. Then the top k alerts can be filtered for the final set of alerts Z, such that the expected score of the filtered alerts is bounded by $\delta' \leqslant E[F|Z] \leqslant 1$ where $\delta' \geqslant \delta$ is the score of the latest filtered alert in Z, i.e. $\delta' = P(F|Z_k)$. This alert has the lowest score of all filtered alerts, because the alerts were ordered by score. If we would filter k alerts to output Z' randomly, then the expected score of the filtered alerts is bounded by $\delta \leqslant E[F|Z'] \leqslant 1$. So taking the first k ranked alerts improves the performance more likely better than taking k alerts randomly.

The *score-top-k* filter is implemented as follows. For each day, the filter sorts all alerts of that day by the model score into descending order. It is possible that some of the alerts have an equal model score. Therefore, the filter also sorts all alerts by the date of the transaction into ascending order. Then this filter takes the first k alerts as final output (the the filtered alerts). This is achieved by replacing the model score with a new score value to update the predictions. The filter uses a score value of 1.0 for the first k alerts, and 0.0 for all other alerts.

Note that our implementation of this *score-top-k* filter does not work directly in streaming settings. A proper streaming algorithm has to be designed to apply it in such scenario. An alternative approach would be to discard alerts below some prediction threshold and use knowledge of historic alerts to maintain this threshold.

4.2.2 Baseline

In the first experiment, we also use a baseline to compare the performance of other filter. This baseline is also called the *random-k* filter. The implementation is very similar to that of the score-based filter. For each day, the filter takes k alerts randomly as final output.

4.3 Experiment goal & setup

The goal of this experiment is that to discard alerts on a daily basis solely using a score-based filter technique (RQ 1.2) and compare it to the performance of the detector (RQ 1.3), and to that of an approach that filters alerts randomly.

This experiment looks as follows. For each day from January 2016 to February 2017, we apply two different types of filters. The first one is a *score-top-100* filter which selects at most 100 alerts per day that have highest values of scores. This filter is compared with a baseline. This baseline is a *random-100* filter which selects at most 100 alerts per day randomly. We expect that the former has a better performance than the latter as described in Section 4.2.1.

4.4 Results

In this section, the results of the experiment of filtering alerts by solely using the model scores are described. The reduction of the alerts by the score-based filter for each month is presented in Table 4.1. All FPRs and TPRs can be found in Table B.5. From this table follows that the score-based filter is able to discard a lot of FPs. However, there is also a reduction of TPs for all months. The reduction of TPs is very high in February 2017. The reason for this higher reduction is that score values of TPs are much lower during this month.

Both filters achieve in reducing the FPR for all days in January 2017 (Figure 4.2). The random filter (baseline) also never discarded more of the FPs than the score filter. Less FPR reduction occurs during weekends, this is because there are also less alerts on these days (Section 3.4). Unfortunately, the score-based filter also discards some of the TPs but not as much as the baseline (random filter).

This reduction can also be observed from the ROC curves of the filters compared to that of the original fraud detector. In Figure 4.3a this is depicted for January 2017. From this figure follows that performance of the score-based filter is close to that of the detector. The performance of the random filter is much worse and also closer to the diagonal of random guesses. Moreover, not only the TPR of the random filter is worse in comparison with the other curves, but also the FPR.

The performance of the filters in February is depicted in Figure 4.3b. During this month, the performance of the detector is not as good as during the other months. This can also be concluded from a comparison of the partial AUC (Table 4.1). There are fraudulent cases during February which are unknown to the detector, which makes the detector a less good ranker. The score-based filter therefore also performs worse here. Moreover, it is notable that the random filter has a slightly better performance here than during January.

Recall that FPR and TPR for the detector are respectively 0.0304 and 0.9649 for all months (Table 3.2). The *score-top-100* filter has a FPR and TPR of respectively 0.0013 and 0.8296 (Table B.5). So the FPR is reduced by a factor of ∼23 while the TPR is still high. The performance of this score-based filter and that of the original detector are presented in a ROC plane in Figure 4.4. From this figure follows that the TPR after filtering is slightly higher than that of the detector at FPR of the filter. The pAUC on the other hand is much lower after filtering. This is because the TPR after filtering is lower than that detector could achieve with lower prediction thresholds (Figure 4.4).

Table 4.1: Reduction of alerts by *score-top-100* filter per month. The partial AUC (pAUC) is computed for the region of FPR $\leqslant 0.035$. The score-based filter is able to discard a lot of FPs, and discards a total of 262 TPs. Many of the TPs are discarded in February 2017, which did not belong to the training period of the model.

	Before			After		
Month	FP	TP	pAUC	FP (ΔFP)	TP (ΔTP)	pAUC
January 2016	33,657	51	0.0342	3,055 (−30,602)	45 (−6)	0.0304
February 2016	83,221	47	0.0321	2,862 (−80,359)	38 (−9)	0.0271
March 2016	112,702	93	0.0340	3,014 (−109,688)	86 (−7)	0.0317
April 2016	129,163	126	0.0340	2,890 (−126,273)	110 (−16)	0.0302
May 2016	145,964	144	0.0327	2,980 (−142,984)	120 (−24)	0.0282
June 2016	141,350	165	0.0320	2,883 (−138,467)	117 (−48)	0.0244
July 2016	136,628	83	0.0339	3,034 (−133,594)	66 (−17)	0.0275
August 2016	55,563	100	0.0320	3,014 (−52,549)	86 (−14)	0.0285
September 2016	6,450	47	0.0321	2,937 (−3,513)	46 (−1)	0.0282
October 2016	15,466	110	0.0334	2,800 (−12,666)	102 (−8)	0.0299
November 2016	15,952	165	0.0340	2,841 (−13,111)	159 (−6)	0.0316
December 2016	17,213	173	0.0341	2,933 (−14,280)	167 (−6)	0.0318
January 2017	15,241	311	0.0346	2,794 (−12,447)	306 (−5)	0.0323
February 2017	13,872	254	0.0277	2,641 (−11,231)	159 (−95)	0.0191
All months	922,442	1,869	0.0325	40,678 (−881,764)	1,607 (−262)	0.0289

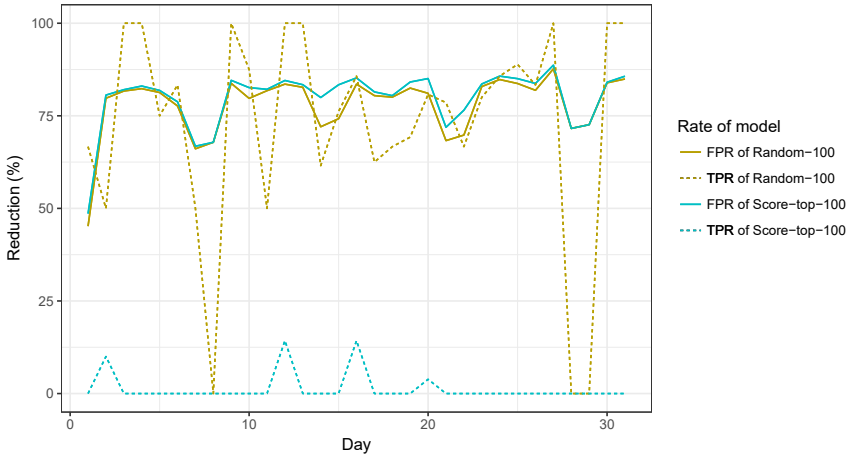
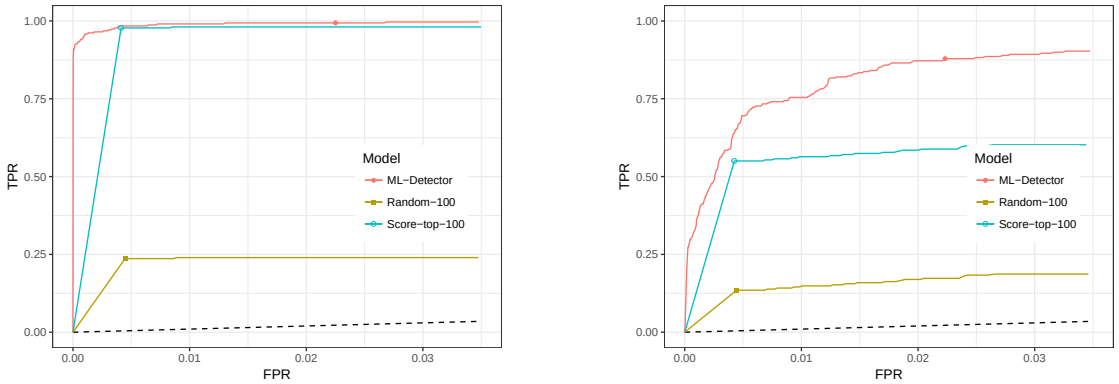


Figure 4.2: Rate reduction in percent of filters in January 2017. The X-axis represents the day of the month and the Y-axis represents the reduction of the rate in percent. From the FPR plots can be observed that the filters reduce a lot of the FPR per day, and with less reductions during weekends. This reduction has an average of 75% per day for each filter. The score-based filter only discards a few TPs on a few days.



(a) ROC curves in January 2017

(b) ROC curves in February 2017

Figure 4.3: Performance of fraud detector and filters in ROC plane per month. The X-axis (FPR) is cut off to only visualize the curve in the range of low FPRs. The dashed line represents the diagonal of random guesses. The performance of the detector is worse in February (Figure 4.3b) than in January (Figure 4.3a).

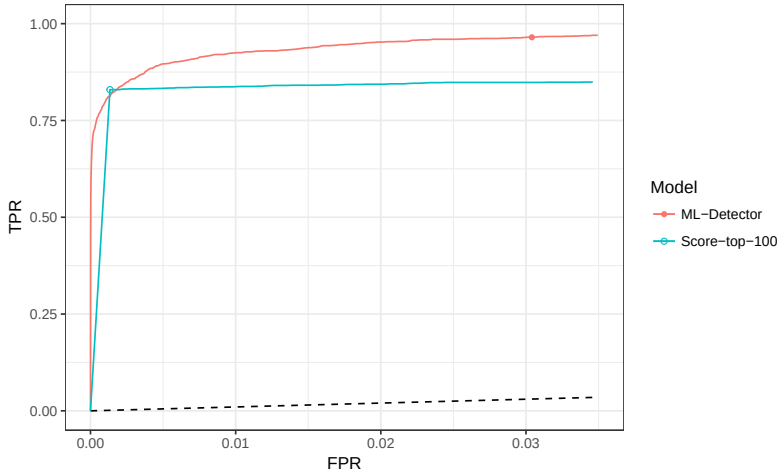


Figure 4.4: Performance of fraud detector and filters in ROC plane for all months. The X-axis (FPR) is cut off to only visualize the curve in the range of low FPRs. The dashed line represents the diagonal of random guesses. The TPR after filtering with the score-based filter for all months is slightly higher than that of the detector at FPR of the filter (∼0.0013).

4.5 Conclusion

From these results can be concluded that the tested filters are able to discard the FPs with the penalty of also discarding TPs. The random filter (baseline) has worse performance than the others as expected. The score-based filter was able to reduce the FPR by a factor of ∼23 while maintaining a high TPR on average for all months. From a comparison of the TPRs of the detector and the score-based filter at the FPR of the filter followed that the TPR of the filter is higher than that of the detector.

The score-based filter is however model-agnostic, because it relies on the detector's model being a good ranker. Meaning, if there are many alerts and if the model is uncertain of the prediction and thus the score of a positive (TP) is closer to the alert threshold, then the chance that the score-based filter discards this alert is much higher.

A filter that solely uses the model scores is thus not good enough. Different techniques to filter the alerts are necessary. In the next chapter, we use the data of the alerts to mine descriptions that could be used to discard alerts.

Chapter 5

Anomaly Description Mining Framework

In this chapter, the alerts generated by the ML-based fraud detector are explored. Moreover, a framework is introduced to mine descriptions that can make a distinction between the alerts. In previous chapter (Chapter 4) a reduction technique that solely uses the probability scores of the detector has been used. From the results followed that such technique is not good enough, as it discards too many TPs in some cases. Especially for February 2017 for which the model of the detector is a less good ranker. The research objective of this chapter corresponds to RQ 2 and is as follows.

- We want to know whether there are descriptions of features and values in the alerts, such that a distinction between the classes of alerts can be made.
- We want to be able to obtain such descriptions of features and values using a framework, such that they can be applied in filtering techniques.
- We want to be able to obtain insights of characteristics of these descriptions, such that we can argue whether the descriptions are valuable.

This chapter is structured as follows. A few definitions used in this chapter and in the remainder of this thesis are given in Section 5.1. Techniques that are used in the experiments are described in Section 5.2. A summary of a few options to find descriptions that could distinguish the classes of alerts are described in Section 5.3. A small experiment is described in Section 5.4. The Anomaly Description Mining Framework and its purpose are described in Section 5.5. The experiment that uses this framework is described in Section 5.7, and followed by a description of the results in Section 5.8. Finally, a conclusion of this chapter is given in Section 5.9.

5.1 Definitions

A few definitions that are used in the remainder of this thesis are described in this section.

Description We define a *description* as a set of distinct $(feature, value)$ pairs. The *length* of a description is the number of feature-value pairs. For example, the description {V_0330=0.795, C_0069=0.136} has length two. We say that a description d *describes* an alert p if and only if all feature-value pairs are satisfied for this alert. This is formally defined as follows

$$describes(d, p) = \bigwedge_{(f,v) \in pairs(d)} value(f, p) \equiv v, \tag{5.1}$$

where $pairs(d)$ is a set of all feature-value pairs of d, and $value(f, p)$ is the value of feature f for alert p.

Subgroup We define a *subgroup* $G(d, T)$ with some description d as the set of all alerts of an alerts dataset T that can be described by this description d. Formally, this is defined as follows.

$$G(d, T) = \{p \in T \mid describes(d, p)\} \tag{5.2}$$

Support The *support* of a description d is basically an indication of how frequently the description describes an alert in the alerts dataset T. This metric is defined as follows.

$$supp(d, T) = \frac{|G(d, T)|}{|T|}, \tag{5.3}$$

where $|X|$ is the size of set X.

5.2 Techniques

Techniques that are used in, or related to, the remainder of this chapter are described in this section. The first technique is used in a small experiment and the other two techniques are used by our framework.

5.2.1 Biclustering

Biclustering is used in our first experiment and it is different than other clustering techniques. A biclustering algorithm simultaneously clusters the rows and columns of a matrix. Then these clusters of rows and columns are the *biclusters* of the data.

There are many biclustering algorithms proposed in literature. In this experiment, we use Spectral biclustering by Kluger et al. to find biclusters in feature data of the alerts. This algorithm has been designed in the domain of bioinformatics with the purpose to identify "marker genes" that are differentially expressed in particular sets of "conditions" [Kluger et al., 2003]. Then this algorithm is used to simultaneously cluster genes and conditions to find distinctive "checkerboard" patterns in data. Then in the context of cancer, these checkerboards correspond to genes that are significantly up- or downregulated in patients with particular types of tumors.

5.2.2 Frequent itemset mining

Frequent Itemset Mining (FIM) is used in the remainder of this chapter to mine frequent itemsets. An *itemset* is basically a collection of items and can be represented by a single description. An itemset is *frequent* if it has a support that is greater than the *minimum support* threshold. There are many algorithms for FIM available in literature. In this project, the optimized Eclat algorithm by Christian Borgelt is used [Borgelt, 2003]. The optimizations by the author achieve better performance for both execution time and memory usage compared to the basic original Eclat algorithm. This algorithm uses intersection operations to determine the support of an itemset along with a depth-first search.

5.2.3 Contrast set mining

Contrast set mining is a method to mine *contrasting sets* [Bay and Pazzani, 2001]. Contrasting sets are descriptions that differ meaningfully in their distribution across groups of data. For instance, groups that represent male or female students. In our case it could be negatives vs. positives. The algorithm of the authors mines contrasting sets that satisfy the following equation

$$|supp(d, T_i) - supp(d, T_j)| \geqslant \delta, \tag{5.4}$$

where d is a description, T_i and T_j are groups of data, and δ is a minimum support difference threshold defined by the user.

Note that this method is different than Contrast Pattern Mining proposed in [Wei et al., 2013] (Section 2.2.1), which is an improved version of the Emerging Pattern Mining (EPM) algorithm [Dong and Li, 1999]. The difference is that EPM aims at mining itemsets that have a significantly increase in support from one dataset to another. While on the other hand, Contrast Set Mining aims at finding discriminating characteristics in groups of data.

5.3 Contrasting set options

Several options for contrasting sets can be considered to find contrasting descriptions that could be useful to distinguish the FPs from the TPs of the detector. For instance, we could consider to use *all data* or to only use the *output of the detector* (data of alerts). We consider the following options:

- *All negatives vs. all positives* yields contrasting descriptions that can distinguish the two classes. These are very interesting descriptions because they could help the initial classification problem (Section 1.2). There are however a lot of negatives (also more TNs than FPs) which makes this option more difficult to experiment with.
- *False positives vs. all positives* yields contrasting descriptions that can distinguish the false alerts from the positives. This option is interesting, because these descriptions will likely be useful with distinguishing the FPs from TPs of the detector.
- *False positives vs. true positives* yields contrasting descriptions that can distinguish the false alerts from correctly detected positives. The previous option includes more information of the positives, but this is option relates more to our problem. Also these descriptions will likely be useful with distinguishing the FPs from the TPs of the detector.

Each of these options is reasonable and applicable to this project. In the remainder of this chapter, we consider the option of *FPs vs. TPs*, because we consider the output of the detector as our input data (Chapter 3).

A small experiment that uses biclustering to identify relations between features is described in the next section.

5.4 Biclustering on feature data of the alerts

The goal of this experiment is to explore whether there exist relations between features locally in data of alerts per day. Moreover, it addresses the first sub-question of RQ 2, which is defined as follows.

RQ 2.1 In a time window of days, are there biclusters that can separate negatives from positives in data of alerts, and how do they project model scores?

Furthermore, we want to know what effect these relations have on the classes of alerts, and on the model scores.

Spectral biclustering is performed on a data matrix defined as *Alerts* $\times$ *Features* on each day of January 2017. The algorithm tries to find four biclusters in the data of the alerts for each day. The found biclusters can then be used to rearrange the rows and columns of the data matrix such that the biclusters are grouped together. Furthermore, we can project the model scores to analyze whether these biclusters also rearrange the model scores. The figures in this section also visualize which alerts are positives with a (small) horizontal red line on both sides of the figure.

On each day, the algorithm finds three biclusters of data with similar feature values, and the remainder forms the fourth bicluster. The figures of the biclusters in this section always visualize this remainder bicluster in the top left corner. We therefore start at the bottom right of the figure to number each bicluster. We discuss our observations with figures of a few days in this section.

One of the first observations that we made is that biclusters are able to separate the positives from the negatives or group model score values on some of the days (Figure 5.1). This can be concluded from the visualization of the data values (Figure 5.1a) and the projection of model score values (Figure 5.1b, Figure 5.1c). A reason of these "anomalies" could be that there were just many cases of the same fraud on these days, which resulted with similar values for multiple features. From the corresponding features names we can also conclude that many of the involved features may indeed be involved in the decision making of an expert.

Furthermore, many feature values are often close to equal for the same feature. This can be concluded from the data matrix depicted in Figure 5.1a, where many columns (features) often have the same color for many rows. This also creates the vertical lines that can be observed from the same figure.

From this small experiment follows that there are indeed relations between features in data of alerts per day. In a few cases, these relations separate the positives from the negatives, or even nicely group the model score values. However, this approach cannot be used to filter alerts, because the observations made in this section only apply for a few cases. A different method should therefore be used to obtain more useful feature descriptions. An "Anomaly Description Mining Framework" is proposed and described in the next section.

5.5 Framework

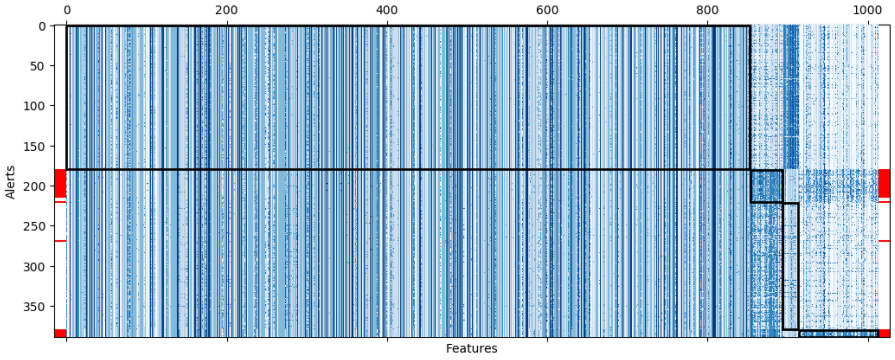
In this section, a framework is described that uses Frequent Itemset Mining (FIM) to mine descriptions of itemsets with a low minimum support in FPs and TPs. A user can then provide a few parameters to obtain descriptions that relatively describe more TPs or FPs, i.e. contrasting descriptions of alerts that can make a distinction between the classes of alerts. Moreover, the user can investigate a few visualizations created by the framework to observe properties of the descriptions for a given time window.

Recall from the introduction that this project does not focus on aiding the domain experts (Section 1.4). This framework can, however, be used to aid the domain experts. For instance, the visualizations can be used to provide insights to a few characteristics of the descriptions. The purpose of these visualizations is to argue whether the descriptions could be useful.

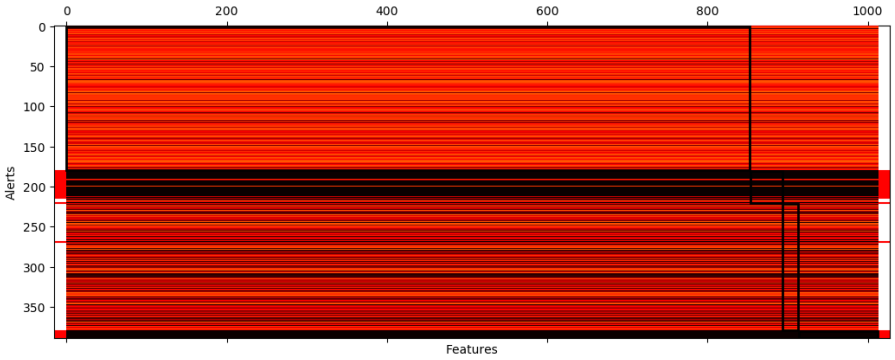
This framework uses FIM on FPs and TPs separately. We denote the resulting descriptions of FIM with D_{FP} and D_{TP} respectively for the executions on FPs and TPs. Furthermore, we have that the set $D = \{d_j \mid d_j \in D_{\text{FP}} \vee d_j \in D_{\text{TP}}\}$ consists of all mined descriptions. The support for an itemset with description $d_j \in D$ is denoted by $supp_{\text{FP}}(d_j)$ and $supp_{\text{TP}}(d_j)$ respectively for FPs and TPs. Note that the support is actually defined as $supp_{\text{FP}}(d_j) = supp(d_j, \text{FPs})$, and similarly for the TPs. Furthermore, the *support difference* of a description d_j is computed by the following formula.

$$supp_{\text{diff}}(d_j) = supp_{\text{FP}}(d_j) - supp_{\text{TP}}(d_j) \tag{5.5}$$

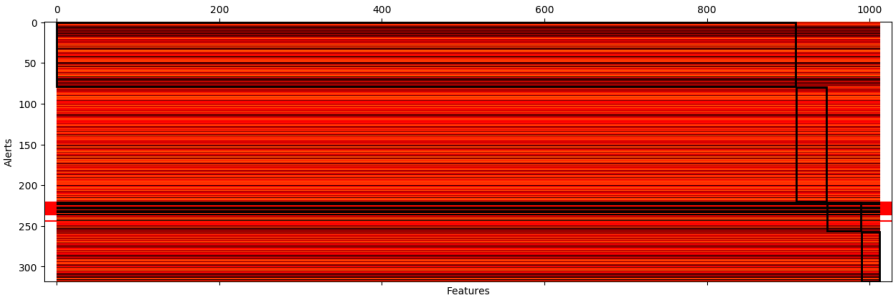
Note that our support difference (5.5) is different than that of work described in [Bay and Pazzani, 2001] (Section 5.2.3). The reason for this is that we want to take into account whether the support for TPs is greater than that for FPs. Furthermore, note that not every description can be found in the result of the other class when its support value is below the minimum support threshold of the algorithm. For such itemset, we assume that the support for the other class is slightly below the minimum support parameter of the FIM algorithm. For instance, if we have that $d_j \in D_{\text{TP}} \wedge d_j \notin D_{\text{FP}}$ and the minimum support of FP was 0.01, then we assume that $supp_{\text{FP}}(d_j) = 0.00999$.



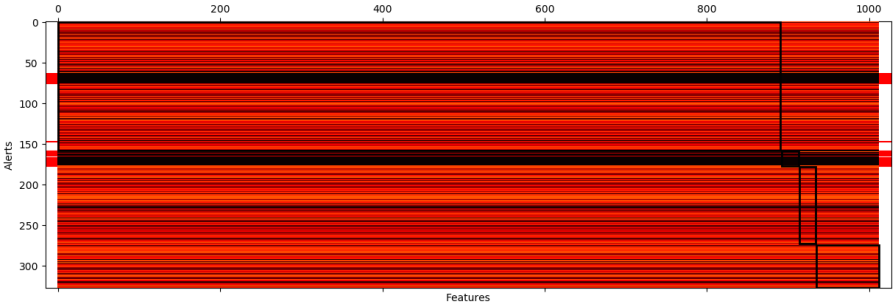
(a) Data matrix values on January 15th 2017



(b) Model scores projection on January 15th 2017



(c) Model scores projection on January 21st 2017



(d) Model scores projection on January 22nd 2017

Figure 5.1: Biclusters separate alerts and model score values on different days. The rows and columns have been rearranged to show the biclusters. In both types of figures, the columns represent the features. In the data matrix (Figure 5.1a), the rows represent the data of the alert in a color mapping from white to blue for the values in the range of $[0, 1]$. Similarly, the rows in the model scores projection matrix (Figure 5.1b) represent the scores values of the corresponding alerts in a color mapping from red to black respectively for low and high scores. The horizontal red lines on both sides of the figure are used to illustrate the positives. On January 15th, bicluster 1 and 3 consist almost of all positives of the alerts as can be observed from both figures. Furthermore, the model scores projections show that the biclusters also group the alerts on similar model score values. Similar observations can be made for other days, for instance for January 21st (Figure 5.1c) and 22nd (Figure 5.1d).

Using these definitions, and a few additional parameters, the framework can obtain two kind of descriptions. These kind of descriptions are obtained as follows.

1. Select descriptions that have a support difference greater than some threshold, and the support for TPs must be lower than another threshold. Hence, all descriptions $d_j \in D \wedge supp_{\text{diff}}(d_j) \geqslant \alpha_1 \wedge supp_{\text{TP}}(d_j) \leqslant \alpha_2$ for some thresholds α_1 and α_2. This should result with descriptions that support as many FPs as possible and support as few TPs as possible. We call these descriptions FP-descriptions.

2. Same as previous, but now with swapped classes and inverted support difference. Hence, all descriptions $d_j \in D \wedge supp_{\text{diff}}(d_j) \leqslant \beta_1 \wedge supp_{\text{FP}}(d_j) \leqslant \beta_2$ for some thresholds β_1 and β_2. We call these descriptions TP-descriptions.

Furthermore, the framework can be used to create a few visualizations that can be used to qualitatively analyze properties of the descriptions for a some time window. These properties correspond to the following sub-questions of RQ 2 which are described as follows.

RQ 2.2 Are there many descriptions with a high support difference, and are these descriptions unique or similar?

RQ 2.3 Do the descriptions work well to describe alerts and when?

RQ 2.4 How do the descriptions relate to score values of alerts?

RQ 2.5 Do the descriptions describe different alerts, or do they have "common alerts"?

An overview of the Anomaly Description Mining Framework is illustrated in Figure 5.2. FIM is used to mine descriptions D from the alerts that have a low minimum support during a large specified time window. FP-descriptions and TP-descriptions can then be obtained by specifying the parameters $\alpha_1, \alpha_2, \beta_1$ and β_2. Furthermore, the framework can be used to create a few visualizations for these descriptions. These visualization methods are described in the next section.

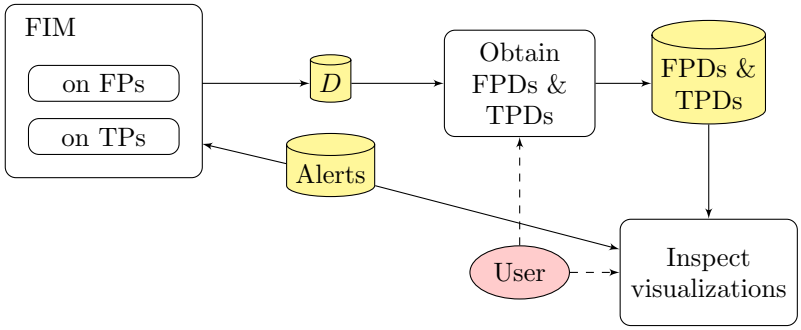


Figure 5.2: Anomaly Description Mining Framework. Here "FPDs" and "TPDs" are respectively FP-descriptions and TP-descriptions, and D is the set of all mined descriptions. Here the dashed lines represent user interactions with the framework. The first step is that FIM is used on FPs and TPs separately. Then a user can obtain FP-descriptions and TP-descriptions from the mined descriptions by specifying a few thresholds. These descriptions can then be analyzed by inspecting a few visualizations.

5.6 Visualization methods

5.6.1 Description coverage heatmap

For a given set of descriptions D and a set of time intervals I, we want to investigate which of them work well to describe the alerts and when (RQ 2.3). This is done by visualizing the percentage of

the alerts that can be described by one description for some some time interval. To visualize this *description coverage heatmap* per time interval, an $I \times D$ matrix is defined for some time window. We denote the coverage of single matrix value with $C_y(t_i, d_j)$ where y is the type of alert (e.g. TP), $t_i \in I$ is a time interval, and $d_j \in D$ is a description. Then all values of the matrix are computed as follows

$$C_y(t_i, d_j) = \frac{|G(d_j, A_y(t_i))|}{|A_y(t_i)|}, \tag{5.6}$$

where $A_y(t_i)$ are the alerts of type y for time interval t_i. This formula basically defines that a single cell of the matrix is equal to the number of alerts of type y described by d_j during t_i normalized by the number of alerts of type y during the same time interval t_i. Hence, the percentage of the alerts that can be described by d_j during t_i.

Using this matrix, a description coverage heatmap can then be created for all alerts, FPs, and TPs. Note that images of these heatmaps could use a different mapping for values to colors, because the maximum value of coverage is not the same in all experiments. The color mapping that is used to render a heatmap is also included in the visualization.

5.6.2 Common description heatmap

For a given set of descriptions D, we want to investigate which of them describe the same alerts (RQ 2.5). From now on, we call this the *common alerts* of two descriptions. This investigation is done by visualizing the percentage of the alerts that can be described by both descriptions for one of the descriptions. To visualize this *common description heatmap* per time interval, a $D \times D$ matrix is defined for some time window. We denote the common description coverage of single matrix value with $CD_y(d_i, d_j)$ where y is the type of the alert, and $d_i, d_j \in D$ are descriptions. Then all values of the matrix are computed as follows

$$CD_y(d_i, d_j) = \frac{|G(d_i, A_y) \cap G(d_j, A_y)|}{|G(d_i, A_y)|}, \tag{5.7}$$

where A_y are the alerts of type y for some time interval. This formula basically defines that a single cell of the matrix is equal to the number of alerts described by d_i and d_j divided by the number of alerts described by d_i. Hence, the percentage of alerts that are common with description d_j (column) for description d_i (row). Note that $CD_y(d_i, d_j) = 1.0$ if $d_i = d_j$, and thus the diagonal is always 1.0.

Using this matrix, a common description heatmap can then be created for all alerts, FPs, and TPs. The color mapping that is used to render a heatmap is also included in the visualization.

5.7 Experiment goal & setup

The goal of this experiment is to use the Anomaly Description Mining Framework to obtain descriptions that can distinguish the positives from the negatives of the alerts, i.e. the TPs from the FPs. In these experiments, FIM is used on alerts of August 2016 up to January 2017 with all features to mine itemsets of length 2 and with a minimum support of 0.1 and 0.05 respectively for data of FPs and TPs. Our framework is used to obtain FP-descriptions and TP-descriptions from these descriptions. Moreover, visualizations of these descriptions are created by the framework to answer sub-questions RQ 2.3 up to RQ 2.5 for both kind of descriptions. Alerts from week nr. 40 of 2016 up to week nr. 8 of 2017 are used to generate these visualizations.

5.8 Results

One of the first observations we could make from a comparison of class supports is that descriptions with a low/high support for TP (or FP) also have a low/high support for the other class. We are interested in the descriptions that have a high absolute support difference. We therefore want

to know whether there are many descriptions that have a high absolute support difference (RQ 2.2). In Figure 5.3 the absolute support different for each description is illustrated. From this figure follows that most of the descriptions have an absolute support difference of less than 0.1. There are not so many descriptions that have a high absolute support difference. We can also observe that some descriptions of itemsets that are very similar in name of features, can also have similar support values (Table B.1). In the next sections, the results of FP-descriptions and TP-descriptions are described. These descriptions can be found in Section B.1 and each description has a unique identifier (ID).

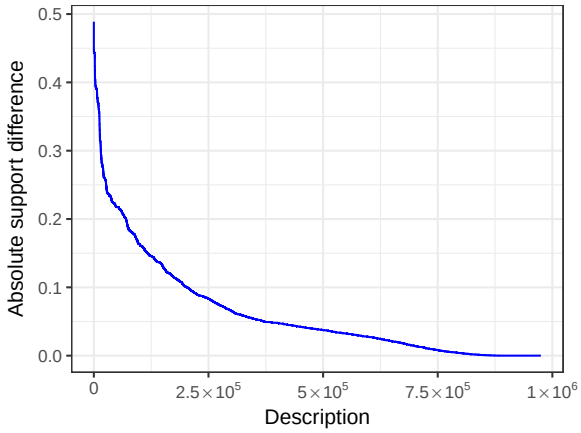


Figure 5.3: Absolute support difference for each description. The descriptions are sorted by the absolute support difference into descending order. Most descriptions have an absolute support difference of less than 0.10, and thus the two supports of those descriptions are close. Not a high percentage of the descriptions have a high absolute difference in support.

5.8.1 Results FP-descriptions

The FP-descriptions are descriptions with a low support for TPs and a high support difference. We obtain 19 different itemsets (Table B.1) with a TP support lower than 0.06 and a support difference greater than 0.19. Images of the corresponding heatmaps are depicted in Figure 5.4. Note that the descriptions have been sorted on the support difference in descending order. This can also be observed from the color gradient effect in the heatmap for FPs (Figure 5.4a). First we investigate the description coverage (RQ 2.3) and the distribution of the model scores per description (RQ 2.4). From the FP heatmap and corresponding legends can be concluded that the first descriptions describe more than 30% of the FPs per week. For weeks before February, these descriptions describe not too many TPs per week. The TP description coverage of the first descriptions is however again much higher for weeks in February (the last three rows). These descriptions also have higher model score values (Figure 5.4c). The description coverage of FPs is between 20% and 30% for most of the descriptions for each week. The description coverage of TPs is 5% or lower for most descriptions for almost all weeks. All descriptions (except the first two) also describe about 70% of alerts that have score values lower than 0.7, and with the median around 0.6.

Next we use the *common description heatmap* visualization method to investigate whether descriptions describe the same alerts (RQ 2.5). The common description heatmap of all alerts for these descriptions is depicted in Figure 5.4d. A few observations can be made from this figure. First of all, the descriptions in the middle (FP6 up to FP11) describe the same alerts during November to February. This also forms the large white rectangle in the center, and this is because these descriptions have the feature-value pair C_1055=0.124 in common (Table B.1). Similarly,

there is a smaller white rectangle for FP3 up to FP5. According to a domain expert this is because the different descriptions describe the same scenario. Furthermore, the last descriptions (FP14 and FP15) do not have relatively many alerts that are common with other descriptions. These descriptions also had a relatively low description coverage (Figure 5.4a). Other descriptions however do have around 50% of the alerts they describe in common with these two descriptions. Last but not least, most descriptions relatively describe quite some alerts that are also described by another description.

A domain expert has looked at translations of these anonymized descriptions. He could observe a few "unexpected" combinations of features and values for usual situations. Some of the descriptions even described the same scenario while they are not the same. This could also be observed from our common description heatmap. Lastly, some descriptions just did not make any sense at all.

Although some descriptions did not make any sense at all to a domain expert, this method could mine and obtain descriptions that are more likely to describe a FP than a TP of the detector. Most of these descriptions have a reasonable low description coverage for TPs during the evaluated time window. Moreover, most of these descriptions described alerts with low score values.

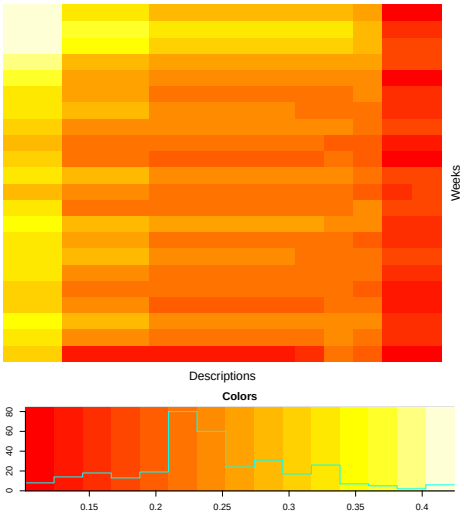
5.8.2 Results TP-descriptions

A similar analysis can be done for descriptions with a low FP support and high negative support difference. We obtain 33 different descriptions (Table B.2) with a FP support lower than 0.13 and a support difference lower than -0.29. Images of the corresponding heatmaps are depicted in Figure 5.5. First we investigate the description coverage (RQ 2.3) and the distribution of the model scores per description (RQ 2.4). From the TP heatmap (Figure 5.5b) and corresponding legend can be concluded that the descriptions often describe between 20% and 60% of the TPs, and even with higher percentages during the weeks of December and January. All descriptions also describe most of the time about 15% of the FPs per week. Unfortunately, the number FPs per week (∼450 FPs) is still very high compared to the number of TPs. Also, the percentage of TPs varies per description and per week. This follows from the observation that the heatmap for the TPs is more "chaotic". This is because the description coverage of a description is not similar for other weeks. From the distribution of model scores (Figure 5.5c) we can observe that all of these descriptions describe alerts with score values in the range of 0.5 to 1.0. Furthermore, the median is around 0.7 for most descriptions.

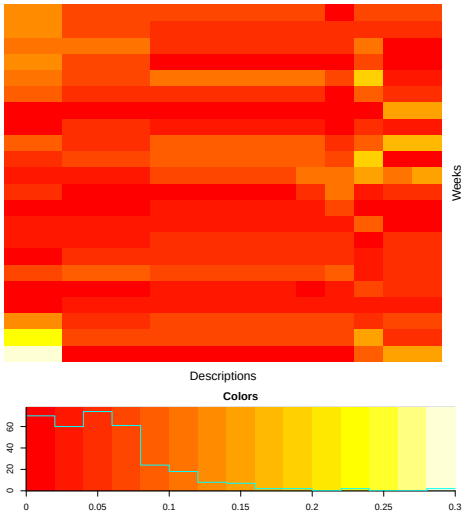
Next we investigate whether the descriptions describe the same alerts (RQ 2.5). The common description heatmap of all alerts of November to February for these descriptions is depicted in Figure 5.5d. A few observations can be made from this figure. First of all, there are a few descriptions that only have relatively few alerts in common with all other descriptions. This can be concluded from the dark red horizontal and vertical lines. The percentage of alerts that other descriptions have in common varies between 30% and 80% for each description. Hence, relatively many of the alerts a TP-description describes are also described by another TP-description.

A domain expert has also looked at translations of these anonymized descriptions. He concluded that these descriptions are nice, but they are too general in terms of describing fraud patterns. This is because each description only has two features. A greater description length of four or five would be better.

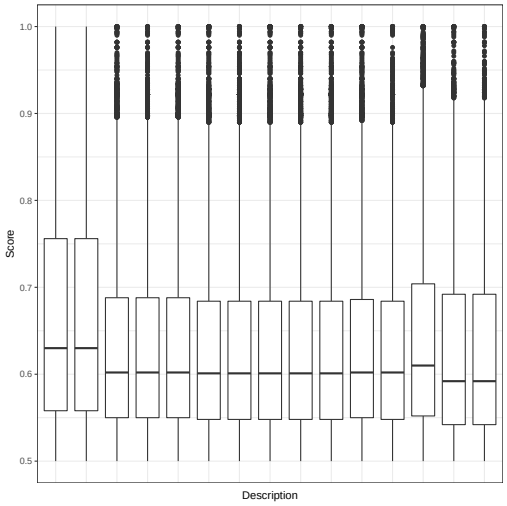
So from this experiment we can conclude that a high support for TP does not yield a very low support for FPs. This is not only because of the fact there are way more FPs than TPs, but also because the length of the descriptions may be too small. In order to find descriptions for itemsets with a very low support for FPs, the FIM algorithm must run with a lower minimum support threshold. However, this will yield a lot descriptions as there are a lot of features and data possible values. For the same reason it is more difficult for our framework to mine descriptions with a greater length. With this method, we can mine and obtain descriptions that can often describe some but not all positives. However, these descriptions still describe many FPs of the detector. The score values of alerts described by these descriptions are in the range from 0.5 to 1.0, and the median is around 0.7 for most descriptions.



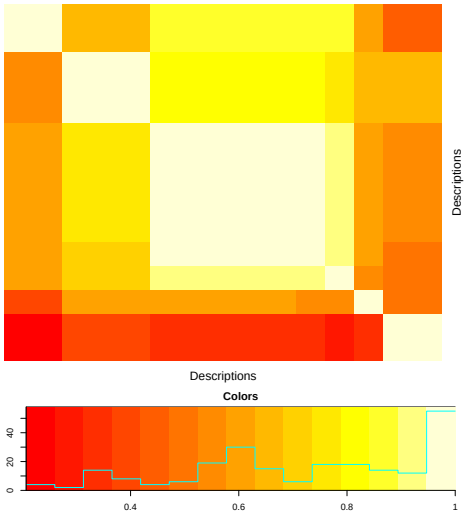
(a) Description coverage of FPs per week



(b) Description coverage of TPs per week

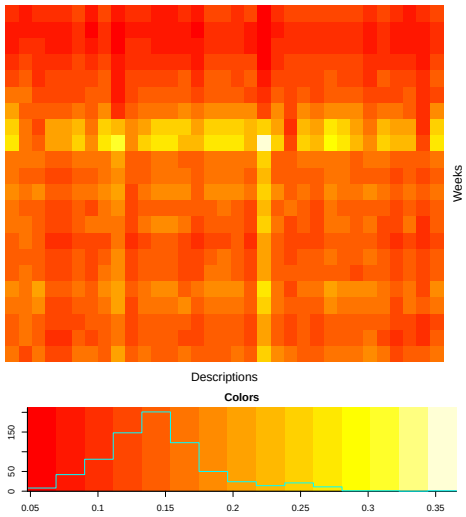


(c) Distribution of scores per description

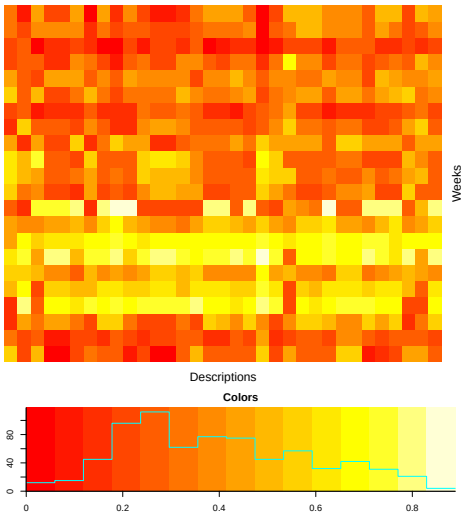


(d) Common description heatmap of alerts

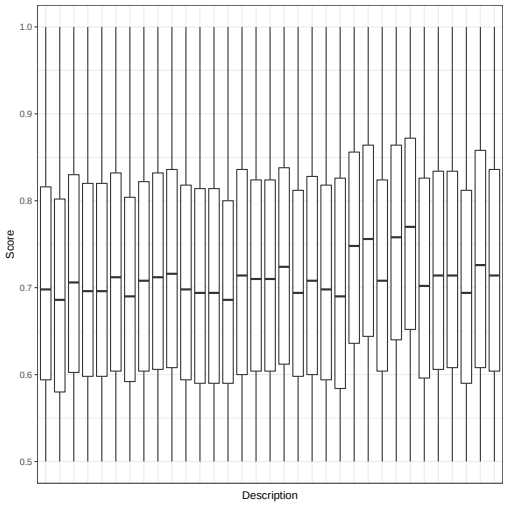
Figure 5.4: Visualizations for itemsets with low TP support and high support difference in August to January. The time interval of these visualizations are weeks from November 2016 up to February 2017. The description coverage of FPs is between 20% and 30% for most of the descriptions for each week (Figure 5.4a). The description coverage of TPs is 5% or lower for most descriptions for almost all weeks (Figure 5.4b). Moreover, all descriptions (except the first two) describe about 70% of alerts that have score values lower than 0.7, and with the median around 0.6 (Figure 5.4c). Many descriptions describe around 50% of the alerts in common (Figure 5.4d). There are also descriptions that describe the same alerts, which can be concluded from the white rectangles.



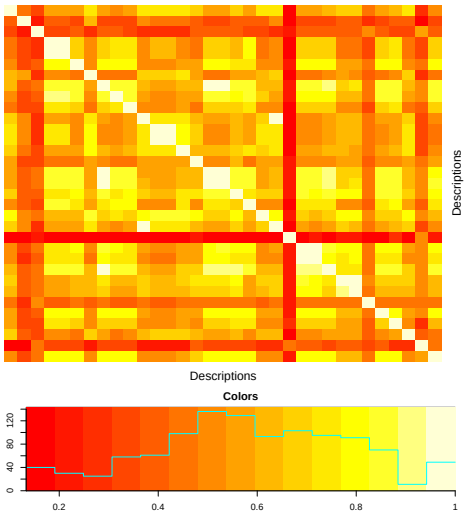
(a) Description coverage of FPs per week



(b) Description coverage of TPs per week



(c) Distribution of scores per description



(d) Common description heatmap of alerts

Figure 5.5: Visualizations for itemsets with low FP support and high negative support difference in August to January. The time interval of these visualizations are weeks from November 2016 up to February 2017. From the description coverage can be concluded that the descriptions often describe between 20% and 60% of the TPs (Figure 5.5b), and even with higher percentages during the weeks of December and January. Moreover, all descriptions describe most of the time about 15% of the FPs per week (Figure 5.5a). Unfortunately, the number FPs per week (∼450 FPs) is still very high compared to the number of TPs. Also, the percentage of TPs varies per description and per week. From the distribution of model scores we can observe that all of these descriptions describe alerts with score values in the range of 0.5 to 1.0 (Figure 5.5c). Furthermore, the median is around 0.7 for most descriptions. There are a few descriptions that only have relatively few alerts in common with all other descriptions (Figure 5.5d). This can be concluded from the dark red horizontal and vertical lines of the common description heatmap. The percentage of alerts that other descriptions have in common varies between 30% and 80% for each description. Hence, relatively many of the alerts a TP-description describes are also described by another TP-description.

5.9 Conclusion

Some properties of alerts have been discovered with a few experiments described in this chapter. Using biclustering, we have found that there are relations between features and values of alerts that are able to separate positives from the negatives of alerts, or even nicely group model score values, on some days of January (Section 5.4). However, the method applied in this experiment is not good enough to obtain useful feature descriptions, because the made observations only applied for a few cases.

An Anomaly Description Mining Framework has been introduced in Section 5.5 to mine descriptions using FIM that can distinguish the classes of alerts. The framework has been used to obtain FP-descriptions and TP-descriptions (Section 5.8). The former are descriptions that have a low support for TPs and still have a relatively high support difference. Similarly, the TP-descriptions are descriptions that have a low support for FPs and still have a relatively high negative support difference. From this experiment also followed that there are not many descriptions that have a high absolute support difference. Using a visualization technique to visualize the *description coverage heatmap* of descriptions (Section 5.6), we have seen that the FP-descriptions describe not too many TPs. According to a domain expert some of these descriptions did not make any sense at all. The coverage of TPs varied per week for each TP-description, and these descriptions still described a few hundred FPs per week. This is because the TP-descriptions may be too general to describe the fraud, which was also a comment of a domain expert. The model scores for alerts described by FP-descriptions are below 0.7 for 75% of the alerts. On the other hand, we have that the model scores of alerts described by TP-descriptions are much higher. Some sets of FP-descriptions described a same scenario with different features. These descriptions therefore described common alerts as could be observed from a visualization of the *common description heatmap*. The remainder of the FP-descriptions described around 50% alerts in common. The TP-descriptions on the other hand, they often described more percent of alerts in common.

Our framework has a few limitations and difficulties. First of all, it has difficulties with mining larger descriptions from many data. Data pre-processing techniques may be able to decrease the search space used by FIM. Furthermore, the FP-descriptions and TP-descriptions could be obtained more efficiently. Secondly, there is no user friendly interaction between the user and the framework. The user must specify thresholds and invoke plotting of figures manually. It would be better for a user to select a "region of descriptions" which automatically invokes plotting of new figures, but this is not in the scope of this project.

In this thesis, mostly quantitative methods are used to analyze the performance of filters. With our framework to obtain descriptions, qualitative methods were used to analyze properties of these descriptions. In the next chapter, the descriptions obtained with our framework are used in experiments to discard alerts.

Chapter 6

False Alert Reduction Using Descriptions

In this chapter, false alert reduction is performed using the descriptions mined by our Anomaly Description Mining Framework (Chapter 5). A filtering technique that solely uses model scores has been applied in Chapter 4. However, from these results followed that such technique is not good enough, as it reduces too many TPs. The research objective of this chapter corresponds to RQ 3 and is as follows.

- We want to use the descriptions mined by our framework (Chapter 5) to automatically filter alerts, or to improve other filtering techniques.

Moreover, for this chapter we consider the following sub-questions of RQ 3.

RQ 3.1 What is the performance of filtering alerts using filters that solely use the mined descriptions?

RQ 3.2 What is the performance of a RandomForest model that has been trained on data with a feature space limited by the descriptions, and how do the predictions of this model compare to the model of the detector?

RQ 3.3 Using the descriptions, what effect does altering score values have on the distribution of the score values, and on the performance of the *score-top-100* filter?

This chapter is structured as follows. First a general description of the experiments described in this this chapter is given in Section 6.1. Each experiment is described in Section 6.2 up to Section 6.4. Finally, conclusions of this chapter are given in Section 6.5.

6.1 Experiments setup

The format of describing an experiment and its results is slightly different in this chapter. The results of each experiment experiment is described in the same section as the setup and goal of that experiment.

The descriptions of itemsets obtained with our framework (Section 5.8) can be used in different approaches to discard false alerts. Multiple descriptions are used to form ensembles in Section 6.2. The information of the descriptions are used to limit the feature space during training of a RandomForest model in Section 6.3. Furthermore, the descriptions can also be used to modify model scores. This is described in Section 6.4.

6.2 Filtering with descriptions

In this section, an experiment is described and performed that uses the FP-descriptions and TP-descriptions (Section 5.8) to filter alerts. The goal of this experiment is to analyze the performance of using multiple descriptions together to filter alerts (RQ 3.1). Moreover, in this experiment we have the following description-based filters.

1. An *ensemble-FPDs* filter, where multiple FP-description-based filters are combined to form an ensemble with an equal voting mechanism. A FP-description-based filter is a filter that discards all alerts that can be described by the FP-description. A manual selection of FP-descriptions has been made, because some of these descriptions share a common feature-value pair. For this ensemble, we use the FP-descriptions FP1, FP3, FP7, FP13, and FP14 (Table B.1).
2. A *not-TPDs* filter, which discards all alerts that cannot be described by any TP-description (Table B.2), because these alerts are more likely to be a positive.

Each of these description-based filters then filters alerts from January 2016 up to February 2017.

Results & Conclusion The results of these filters, and also that of some other filters, can be found in Table B.5. A few observations can be made from these results. First of all, the results of the *not-TPDs* filter are not good, as it discards a lot of TPs. This is because these TP-descriptions do not describe all "groups" of TPs. The average TPR drops from 0.96 to 0.71 with this filter. Similarly, the FPR drops from 0.030 to 0.011. This also follows from the ROC graph of this description-based filter and that of the original detector shown in Figure 6.1. The *not-TPDs* filter has a worse performance than the model of the detector has. The pAUC for this filter therefore also decreases by 0.01 to 0.0211.

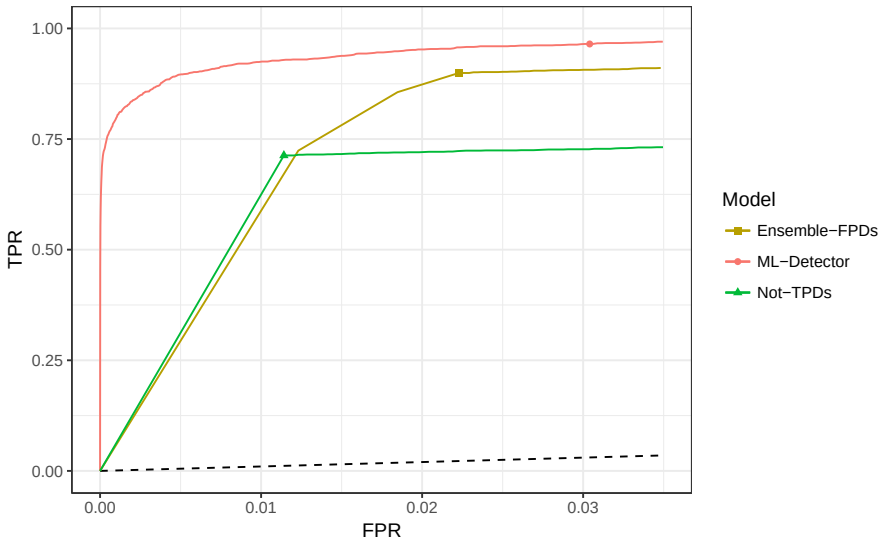


Figure 6.1: Performance of fraud detector and description-based filters in ROC plane. The X-axis (FPR) is cut off to only visualize the curve in the range of low FPRs. The dashed line represents the diagonal of random guesses. The curves of the description-based filters are below the curve of the detector.

The *ensemble-FPDs* filter on the other hand, discards less alerts than the previous filter. This ensemble is still able to discard a few thousands of FPs per month. Moreover, this filter only incorrectly discards a few TPs in for each month. The average TPR drops from 0.96 to 0.90 with this filter. Similarly, the FPR drops from 0.030 to 0.022. This filter has a worse performance

than the *score-top-100* filter (Chapter 4), because it discards more or a similar amount of TPs for most months, while it discards way less FPs. This filter still achieves a higher TPR than the score-based filter, but the FPR of the score-based filter is 16 times smaller. Although the TPR is high for this filter, it also has a worse performance than the model of the detector as shown in Figure 6.1. The pAUC for this filter therefore also decreases by 0.008 to 0.0242, which is still higher than that of the other description-based filter.

So these filters perform not very good compared to the score-based filter and the detector. Recall that the description mining period was from August 2016 up to January 2017 (Section 5.7). The performance of these filters is also different for the months before this mining period. First of all, the *not-TPDs* filter discards way more TPs during these months. The reason for this could be that characteristics of these TPs are different. Hence, there is no TP-description that could describe these different alerts. Furthermore, the *ensemble-FPDs* filter is still able to discard many FPs, but for instance, it also discards 22 TPs in July 2016 (Table B.5). It is also notable that this filter discards almost no alerts in January 2016.

So the conclusion of this experiment is that a filter that solely uses the mined descriptions do not perform very well compared to the score-based filter. The description-based filters also have a worse performance than the model of the detector. Furthermore, the TP-descriptions should not directly be used to discard alerts, because these descriptions cannot describe all TPs. This then leads to discarding many TPs. Finally, we have observed that the performance of these filters is different for alerts before the mining period. The mined descriptions should be used differently to achieve better performance.

6.3 RandomForest

In this section, an experiment is described that uses a RandomForest trained with information of the FP-descriptions and TP-descriptions. The goal of this experiment is to train a RF-based filter by limiting the feature space to the information of the mined descriptions (RQ 3.2). If a description is good, then the model will most likely include this in its decision making. Moreover, if a description is less good, then the algorithm compensates this with other trees. For these reasons, the model should have a better performance than the filters of previous section.

The RandomForest is trained on the same training dataset as the detector (Chapter 3) and is limited to only use the features of the descriptions. Moreover, cross validation with 10 folds is used to partition the dataset. This RF-based filter is then used to filter alerts from January 2016 up to February 2017.

Results & Conclusion The results of this *RF-Description* (or *RF-Desc*) filter, and also that of some other filters, can be found in Table B.5. A few observations can be made from these results. First of all, this filter is able to filter many FPs for each month. Moreover, only a few or none TPs are discarded for each month before February 2017. For these months, the performance is better than that of the filters of previous section.

This RF-based filter still has a good TPR compared to that of the detector. The average TPR drops from 0.96 to 0.93 with this filter. Similarly, the FPR drops from 0.030 to 0.010. The TPR of this filter is higher than that of other filters. Moreover, the FPR is also lower than that of the filters described in previous section. The performance of this RF-based filter and that of the detector is illustrated in Figure 6.2. From this figure follows that the filter achieves a slightly better performance for lower FPRs. The TPR is however slightly higher for FPRs closer to the instance of the detector that has been extended by this filter. This also follows from a comparison of the pAUC, which is 0.0003 lower than that of the detector (Table B.5). Although the performance of this filter is close to that of the detector, it is a different model and therefore also has different predictions. We can also conclude that this filter has better performance than the filters of previous section.

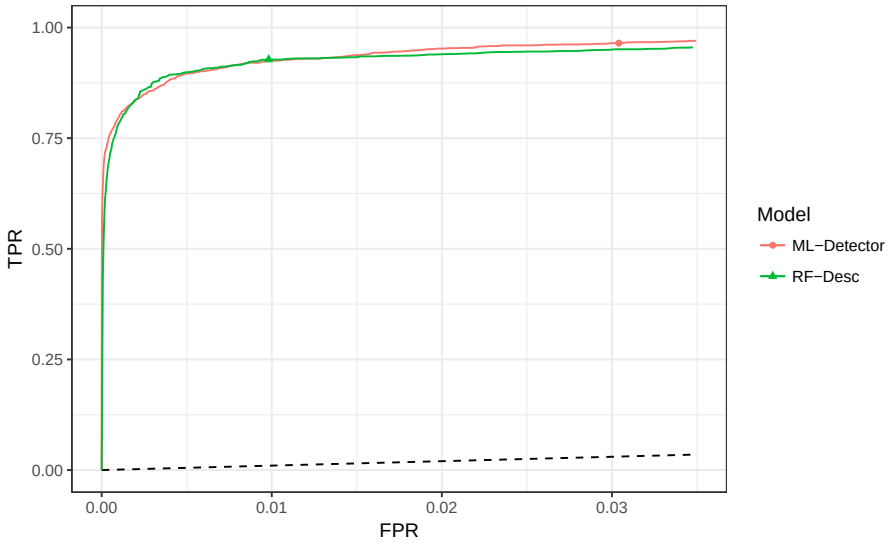
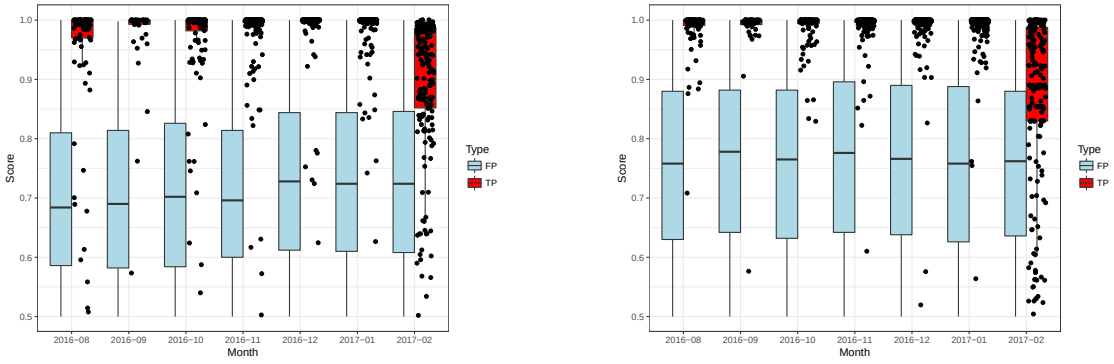


Figure 6.2: Performance of fraud detector and RF-Desc filter in ROC plane. The X-axis (FPR) is cut off to only visualize the curve in the range of low FPRs. The dashed line represents the diagonal of random guesses. The RF-based filter has a performance close to that of the detector.

This model of RandomForest also provides a probability that a given alert is fraudulent according to the model. The distribution of model scores of the filtered alerts are visualized in Figure 6.3 with boxplots for the ML-detector and the *RF-Desc* filter. From a comparison of the distribution before filtering (Figure 4.1) with the distribution of model scores after filtering (Figure 6.3a) follows that distribution of model scores for FPs has increased. Thus most of the filtered FPs have low model scores. Furthermore, it can be observed that the model scores of the *RF-Desc* filter are higher than that of the ML-detector for most alerts. The model scores of TPs in February 2017 are also lower than other months for the RF-based filter. This is because the model of the RF-based filter has not been trained on data of February 2017. For the same reason we have that this filter discards 52 TPs (Table B.5), which is a lot more than for the other months.



(a) ML-detector

(b) RF-Desc filter

Figure 6.3: Distribution of model scores of filtered alerts per class for different months. Also the model scores of TPs in February are much lower for the RF-based filter than other months (Figure 6.3b). Moreover, these score values are also slightly lower than that of the ML-detector (Figure 6.3a). The RF-based filter has slightly higher model scores for the other alerts.

So the conclusion of this experiment is that RandomForest is quite able to learn to distinguish negatives from positives of alerts for already observed positives if the feature space is limited to that of the mined descriptions. Furthermore, this RF-based filtering has a better performance than the other description-based filters of previous section. Hence, the mined descriptions were very effective for filtering alerts in this experiment.

6.4 Changing model score values

The experiment of this section is different than the experiments of previous sections. In this experiment, the descriptions are used to change the model score values such that the performance of a score-based filter also changes. The goal of this experiment is to analyze the performance of a score-based filter after changing the scores using the mined descriptions (RQ 3.3).

In this experiment, two *score modifiers* are used to change the model score values. These score modifiers are defined as follows.

1. One score modifier is used to alter scores of alerts that can be described by FP-descriptions. This modifier is denoted by M_{FP}.

2. Similarly, the other score modifier (M_{TP}) is for TP-descriptions.

Then for each description d_j, we alter the score of the alerts that it can describe by subtracting the product of the corresponding modifier and support difference of the description. This computation can be described by the following formula

$$\hat{P}(F|X_i) = P(F|X_i) - \Big(\sum_{d_j \in D(X_i)} M(d_j) \cdot supp_{\text{diff}}(d_j) \Big), \tag{6.1}$$

where $P(F|X_i)$ is the model score for alert of transaction X_i, $\hat{P}(F|X_i)$ is the modified score, $D(X_i)$ is a set of descriptions that can describe X_i, and $M(d_j)$ is the score modifier that corresponds with d_j. The computation is finalized by "clipping" the values such that all values do not exceed the boundaries 0.0 and 1.0.

After computing new scores for all alerts, the *score-top-100* filter (Chapter 4) can use the new score values to filter alerts. This yields in a change of number of FPs and TPs compared to a situation without modified scores values. These changes are respectively denoted by ΔFP and ΔTP. For these metrics, it is desired that ΔFP is negative and ΔTP is positive. We apply this filtering technique on each month from August 2016 to February 2017. Furthermore, we introduce a new metric $\sum_{\Delta\text{TP}}$ that is the sum of ΔTP for all months for which the same score modifiers have been used. To find an optimal configuration for these score modifiers, a grid search is applied such that $\sum_{\Delta\text{TP}}$ is maximal.

Results & Conclusion All results of a large grid search for score modifiers in the last three months is given in Table B.3. From these results follow that M_{TP} should be relatively low to prevent a high increase of reduction in TPs. Similarly, the score modifier for FPs M_{FP} should also be relatively low for the same reasons. However, this modifier seems to have less effect on the change in alert reduction. Now that we know this, we can choose a smaller grid to perform the search for optimal values of score modifiers.

The optimal results of this smaller grid search are given in Table 6.1. Furthermore, all results of this grid search can be found in Table B.4. Note that for this grid search a larger time window has been used, which is from August 2016 up to February 2017. From these results follow that with $M_{\text{FP}} = 0 \wedge M_{\text{TP}} = 0.02$, we have that the sum is maximal with $\sum_{\Delta\text{TP}} = 13$. Furthermore, there are 21 more TPs in February 2017 after applying the *score-top-100* filter than without modifying the scores. Without modifying the model score, the score-based filter discarded 95 TPs (Table B.5), and thus in this approach only 74 TPs are discarded. Unfortunately, there is an additional small reduction in TPs for most of the other months, but the total sum is 13.

Table 6.1: Best results of modified score filtering with small grid search for score modifiers. For both configuration of modifiers, the score-based filter discards 21 less TPs in February, and it filters a total of 13 more TPs for August to February. The complete table of all results can be found in Table B.4.

Month	ΔFP	ΔTP	M_{FP}	M_{TP}	$\sum_{\Delta\text{TP}}$
August 2016	1	-1	0	0.02	13
September 2016	0	0			
October 2016	1	-1			
November 2016	2	-2	⋮	⋮	⋮
December 2016	1	-1			
January 2017	3	-3			
February 2017	-21	21			
August 2016	1	-1	0.01	0.02	13
September 2016	0	0			
October 2016	1	-1			
November 2016	2	-2	⋮	⋮	⋮
December 2016	1	-1			
January 2017	3	-3			
February 2017	-21	21			

A 3D visualization of $\sum_{\Delta\text{TP}}$ and the two score modifiers of this smaller grid search is depicted in Figure 6.4. From this figure we can also conclude that the score modifier M_{TP} has more impact on the sum $\sum_{\Delta\text{TP}}$ than the score modifier M_{FP}. Moreover, it is visible that the red region around $M_{\text{TP}} = 0.02$ is maximal for this sum.

This approach changes the distribution of scores of the alerts. These changes in distribution are depicted in Figure 6.5 per class for different months. A few observations can be made from this figure. First of all, the lowest score value for FPs slightly decreases for all months, while this is less the case for the TPs. This change of score values is the effect of the score modifier M_{FP} for FP-descriptions. All boxplots also shift to the top of the figure. For February this is especially notable for the shift of score values for the TPs. This change is the effect of the score modifier M_{TP} for TP-descriptions. A more difficult observation to make is that some TPs have no change in score values. It could be that there is no description that describes these alerts. The *score-top-100* filter discards less TPs in February, because 75% of these alerts now have a score value above 0.875, while this used to be 0.825 in the original situation.

Results of this filter, and also that of some other filters, can be found in Table B.5. Again, recall that the description mining period was from August 2016 up to January 2017 (Section 5.7). During the months before this mining period, this approach has worse performance than when the scores are not changed. For instance, we have that ΔTP is -33 and -15 for respectively March and April 2016. For all months, we have that $\sum_{\Delta\text{TP}} = -113$. If we compare the average rates with that of the ML-Detector, we observe that TPR drops from 0.96 to 0.77 with this filter. Similarly, the FPR drops from 0.030 to 0.001. This filter still achieves a better performance than the *Not-TPDs* description-based filter. Although the pAUC for February 2017 of this filter is higher than that without modifying the score values, it is still lower than the other filters (Table B.5). Since the TPR drops for this filter, it now has a worse performance than the model of the detector as shown in Figure 6.6. The pAUC for this filter therefore also decreases by 0.007 to 0.0269, which is still higher than that of the description-based filters described in Section 6.2.

So the conclusion of this experiment is that changing the model score values does improve the performance of a score-based filter during February 2017. However, it also significantly decreases the performance for months before the description mining period. One reason could be that the detector is already a reasonable ranker for these months (Chapter 4). Another reason could

be that the descriptions are too general and describe too large subgroups. Lastly, the positives (and negatives) are likely to be different before the mining period. Furthermore, using the FP-descriptions to change the model scores did not have much effect on the performance of the score-based filter.

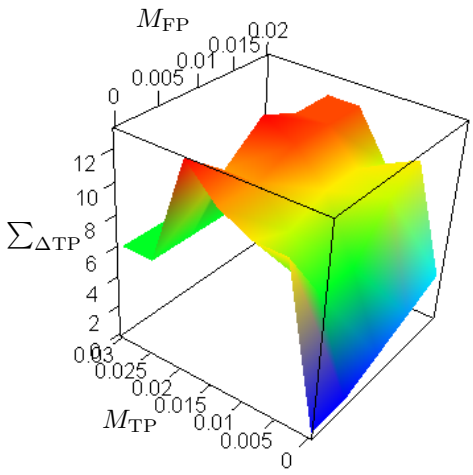


Figure 6.4: Effect of score modifiers on $\sum_{\Delta TP}$. This is a 3D visualization of the grid search results presented in Table B.4. The X-axis represents the score modifier M_{TP}, the Y-axis represents the score modifier M_{FP}, and the Z-axis represents the sum $\sum_{\Delta TP}$. Score modifier M_{FP} has little effect on $\sum_{\Delta TP}$, while on the other hand score modifier M_{TP} must be relatively low.

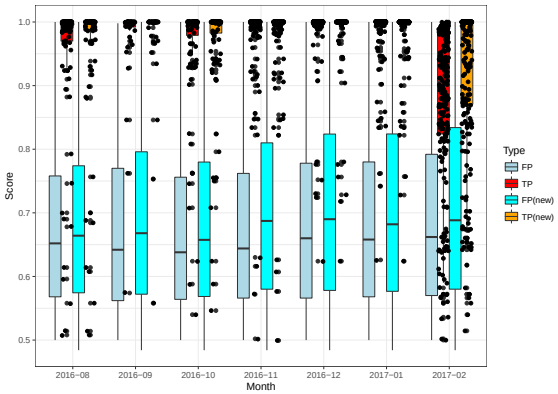


Figure 6.5: Changes in distribution of scores per class for different months. For each month, the original distribution (Figure 4.1) is depicted next to the new distribution of score values. The lowest score value for FPs slightly decreases for all months. Moreover, all boxplots shift to the top of the figure. A larger version of this image is depicted in Figure A.2 of the appendix.

6.5 Conclusion

In this chapter, we have used the FP-descriptions and TP-descriptions to discard alerts (Section 6.2). We have combined several FP-descriptions to discard alerts together and we observed that this filter has a worse performance than the score-based filter and the detector. Moreover, a filter that discarded all alerts that could not be described by any TP-description has not a good

performance. This filter still discarded many TPs, because the TP-descriptions cannot describe all cases of TPs. So we should not use a description-based filter that solely uses TP-descriptions. Furthermore, a description length of two is probably too small to perform filtering, because these descriptions may be too general. This was also a comment of a domain expert (Section 5.8).

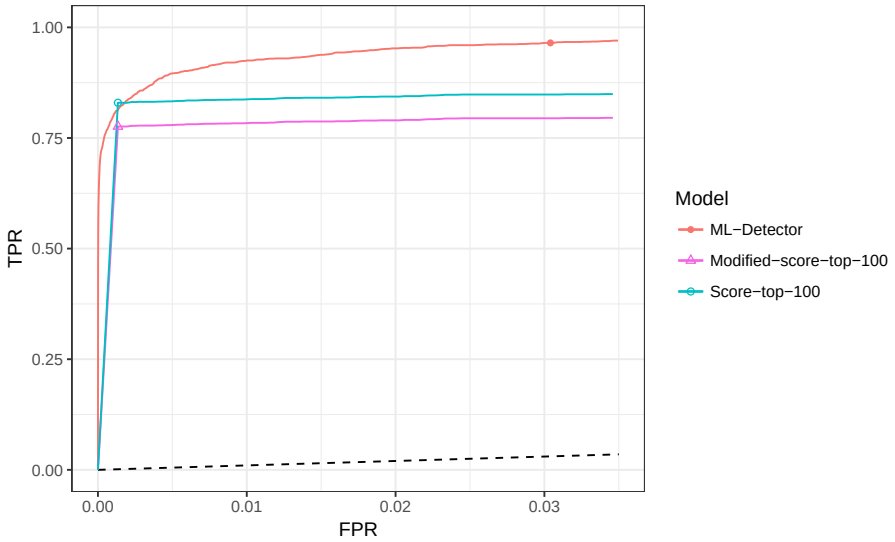


Figure 6.6: Performance of fraud detector and score-based filters in ROC plane. The X-axis (FPR) is cut off to only visualize the curve in the range of low FPRs. The dashed line represents the diagonal of random guesses. The *score-top-100* filter has a slightly better TPR than the detector at the same FPR. Due to changing score values, the new *modified-score-top-100* filter has an average TPR which is lower than that of without modifying these scores. This also yields a filter which has a worse performance than the detector.

The mined descriptions have also been used to limit the feature space during the training of a RandomForest-based filter (Section 6.3). This filter discarded many FPs and none or a few TPs for most months. However, quite a lot of TPs were discarded in February 2017, because this month is not included during training. Furthermore, most of the FPs discarded by this filter had relatively low model scores. This filter only drops the TPR by ∼0.03 compared to the ML-Detector, while the FPR also drops by ∼0.02. From a comparison of models in the ROC plane followed that the performance of this RF-based filter is very close to the performance of the detector, while it is a different model and ranker. This RF-based filter is thus quite good in filtering the alerts.

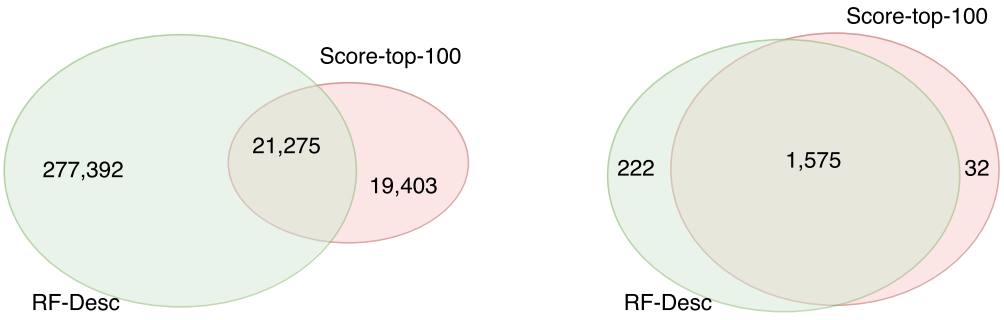
In the last experiment (Section 6.4), we have used the descriptions to compute new score values using two *score modifiers*, and by subtracting values for alerts that could be described by a description. We have used grid search to find the configuration of these score modifiers for which the *score-top-100* filter has the highest increase of TPs. From this followed that the FP-descriptions do not have much effect on the change in performance. Furthermore, the score modifiers should have relatively low values. With this approach, the distribution of score values per class has changed for each month. In February, which has low model score values for TPs, the new score values are slightly increased for the TPs. With this approach, the performance of the score-based filter slightly improved by discarding 13 less TPs for alerts from August up to February (Table 6.1). However, the filter does discard way more TPs before this period. This is because the descriptions were mined from data of August up to January. Hence, the descriptions should be mined from alerts with a greater time window, or all historic positives (or TPs) should be used to mine itemsets from.

Chapter 7

Layered Filtering

So far a few filtering techniques had a good performance in the experiments, because they discarded a lot of FPs while maintaining a high TPR. The RandomForest-based filter that has been used in previous chapter had a good performance for most months. This filter discarded many FPs and maintained a high TPR. The score-based filter of Chapter 4 has a similar filtering performance (Table B.5) but it discards more alerts.

We want to know whether these filters discard different alerts (RQ 4). To investigate this, two Venn diagrams are constructed for all filtered FPs and TPs. Hence, the alerts that are not discarded by a filter from January 2016 to February 2017. These Venn diagrams are depicted in Figure 7.1. From the Venn diagram of FPs (Figure 7.1a) follows that the RF-based filter discards 19,403 FPs that are not discarded by the score-based filter. Hence, these FPs have relative high model scores. Moreover, the RF-based filter also filters 277,392 FPs. These FPs are discarded by the score-based filter because of relatively low scores. Unfortunately, the RF-based filter discards 32 TPs that were not discarded by the score-based filter (Figure 7.1b). However, the RF-based filter discards $222 - 32 = 190$ less TPs. The relatively low FPR of the RF-based filter is 0.0098, but it is still 0.0085 higher than the score-based filter (Table B.5). Similarly, the relatively high TPR of the RF-based filter is 0.9277, which is 0.0981 higher than the score-based filter.



(a) False positives (b) True positives

Figure 7.1: Filtered alerts of *score-top-100* and *RF-Desc* filters. The RF-based filter discards 19,403 FPs that are not discarded by the score-based filter. These FPs have thus relatively high score values. Moreover, the RF-based filter filters 277,392 FPs that have relatively low score values, because these are discarded by the score-based filter. Unfortunately, the RF-based filter discards 32 TPs that were not discarded by the score-based filter, but it discards 190 less TPs.

From this analysis follows the research objective of this chapter. The expectation is that the performance of the score-based filter is better when the RF-based filter is performed first, because the RF-based filter discards a lot of FPs that were not discarded by the score-based filter. Thus there should be less FPs that had relatively high model scores. The research objective of this

chapter correspond to RQ 4 and is as follows.

- We want to filter alerts using a RF-based filter before the score-based filter to improve the performance.

Moreover, for this chapter we consider the following sub-question of RQ 4.

RQ 4.1 What improvements does layered filtering with *RF-Desc* have for the *score-top-100* filters?

The *layered filtering* approach is described in Section 7.1. A description of the setup of the experiment is given in Section 7.2, and followed by the results presented in Section 7.3. Finally, conclusions of this chapter about layered filtering are given in Section 7.4.

7.1 Layered filtering

Pokrywka describes a layered filtering approach that uses multiple detection systems in sequence to reduce the number of FPs [Pokrywka, 2008]. Layered filtering is often used to purify water or air by using at least two filters in sequence. Each filter is then responsible for filtering different type of molecules and together they achieve clean water or air. In our case, each layer is a different model and has its own predictions for a transaction to be fraudulent. Each layer then tries to discard different negatives and together they reduce more FPs of the detector while trying to keep all positives.

For fraud detection the layered filtering approach consists of a sequence of n layered detectors $Ld_1, ..., Ld_n$. Each detector Ld_i has one input for the transactions and two output streams. One output stream represents the filtered genuine transactions, we call this the n-stream. The a-stream is the output stream that represents anomalous transactions according to the detector. Then each detector passes its output a-stream to the input stream of the next detector in the sequence. In Figure 7.2 an example of such layered filtering system is illustrated.

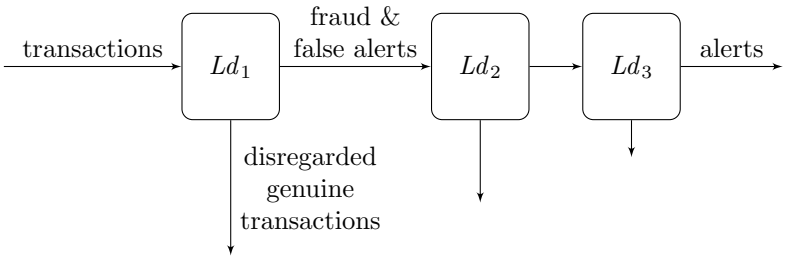


Figure 7.2: Representation of a layered filtering system. Each detector Ld_i disregards transactions that are normal according to the detector. Then the fraud and false alerts are propagated to the next detector Ld_{i+1}.

The authors describe that the difficulty of improving the effectiveness of anomaly detectors is due to base-rate fallacy phenomenon. This has also been described in work by Axelsson [Axelsson, 2000]. This base-rate fallacy relates to the Bayes' theorem. Recall from Section 3.2 that an event F means that a transaction is fraudulent, and then we have that $\neg F$ is a genuine transaction. Furthermore, recall that event A represents an alarm for a transaction from the detector and thus $\neg A$ represents no alarm. Then rates can be expressed as probabilities in terms of F and A. The FPR can be expressed as $P(A|\neg F)$ and the TPR by $P(A|F)$. Similarly, we can express TNR as $P(\neg A|\neg F) = 1 - FPR$ and FNR as $P(\neg A|F) = 1 - TPR$.

The authors have also defined the two probabilities $P(F|A)$ and $P(\neg F|\neg A)$ which they call respectively Bayesian Detection Rate (BDR) and Bayesian True Negative Rate (BTNR). The former represents the probability that an alarm of the system really indicates a fraudulent transaction. BTNR represents the probability that the absence of an alarm of the system indicates

there is no fraud present in the input. We can define $P(F|A)$ and $P(\neg F|\neg A)$ using the extended form of Bayes' theorem as follows

$$P(F|A) = \frac{P(F) \cdot P(A|F)}{P(F) \cdot P(A|F) + P(\neg F) \cdot P(A|\neg F)} \tag{7.1}$$

$$P(\neg F|\neg A) = \frac{P(\neg F) \cdot P(\neg A|\neg F)}{P(\neg F) \cdot P(\neg A|\neg F) + P(F) \cdot P(\neg A|F)} \tag{7.2}$$

The goal in anomaly detection is then to maximize these probabilities. Note that this is different than maximizing the accuracy of predictions $P(F|X_i)$ of a model, where X_i represents a transaction. Maximizing these two probabilities is however difficult for a detection system, because a high BDR requires a low FPR due to the base-rate fallacy. This follows from the equation of $P(F|A)$. In normal settings, the base rate $P(\neg F)$ is very high compared to $P(F)$, because the classes are high imbalanced. Since $P(F)$ is very low, we have that the numerator of (7.1) is also very low. Then the product of $P(\neg F) \cdot P(A|\neg F)$ must be very low in order to maximize $P(F|A)$. As mentioned before, $P(\neg F)$ is very high, therefore the FPR (or $P(A|\neg F)$) must be very low to maximize $P(F|A)$.

In the layered filtering approach, the detector Ld_i operates on a set of alerts with significantly changed base rates of fraudulent and genuine transactions. This reduces the base-rate fallacy influence on the FPR of the complete system. The base rate probabilities ($P(F)$ and $P(\neg F)$) for the next detector in the sequence are as follows

$$P(F)_{i+1} = P(F|A)_i \tag{7.3}$$

$$P(\neg F)_{i+1} = 1 - P(F|A)_i \tag{7.4}$$

We illustrate this with the following example. Assume that we have 1 million transactions of which 10 transactions are fraudulent. Then $P(F) = 10^{-5}$ and $P(\neg F) = 0.99999$. Moreover, assume that we have three layers of different detectors and each layer is able to discard some of the negatives. The classifications for each layer and of the complete system are presented in Table 7.1. Moreover, the corresponding rates for each layer can be found in Table 7.2. Each imaginary layer is able to disregard many genuine transactions with only a few TN in total. Furthermore, the base rate $P(F)$ significantly increases for each layer. This yields a much higher BDR for the last layer and a lower FPR for the complete system.

Table 7.1: Classification of each layer for layered filtering example. The input are 1 million transactions of which 10 are fraudulent. Each imaginary layer is able to disregard many genuine transactions with only 1 TN.

Layer	TN	FN	FP	TP
Layer 1	949,999	1	49,991	9
Layer 2	43,999	1	5,992	8
Layer 3	5,899	1	93	7
System	999,897	3	93	7

Table 7.2: Rates of each layer for layered filtering example. The input are 1 million transactions of which 10 are fraudulent. The rates correspond to the classifications presented in Table 7.1. The new base rate $P(F)$ and the BDR increase for each imaginary layer.

Layer	$P(F)$	$P(\neg F)$	FPR	TPR	BDR	BTNR
Layer 1	0.00001	0.99999	0.04999	0.90000	0.00018	0.99999
Layer 2	0.00018	0.99982	0.11986	0.88889	0.00133	0.99998
Layer 3	0.00133	0.99867	0.01552	0.87500	0.07000	0.99983
System	0.00001	0.99999	0.00009	0.70000	0.000001	0.99999

7.2 Experiment goal & setup

The goal of this experiment is to improve the filter performance of the *score-top-100* filters by using a different filter first (RQ 4.1). The experiment is setup as follows. For all alerts, first the *RF-Desc* filter (Section 6.3) is used to filter the alerts. Then the output of this filter is the input for the *score-top-100* filter (Section 4.2) and the *modified-score-top-100* filter (Section 6.4). The score-based filters use the scores of the ML-Detector, because this model has a lower distribution of model scores for FPs than the RF-based filter (Figure 6.3).

The *score-top-100* filter allows a fixed number of alerts. A layered filtering approach could therefore not change the FPR a lot. Since there are not many positives, the layered approach could change the TPR a lot compared to the score-based filtering without a first layer. This setup could for instance increase the TPR a lot, but it could also decrease the TPR.

Different prediction thresholds could be used for the RF-based filter. A higher threshold then yields an instance with a lower FPR and a relatively high but lower TPR (Figure 6.2). This instance could then discard potentially more FPs of the detector that were otherwise not discarded by the score-based filter. However, in this experiment the prediction threshold that corresponds with the rates of the RF-based filter listed in Table B.5 will be used.

The names of the layered filters are a combination of filter names. Hence, the layered filter without modifying the model scores is called the *RF-Desc_Score-top-100* filter, and the other one is called the *RF-Desc_Mod-score-top-100* filter. An overview of this setup (for one layered filter) is illustrated in Figure 7.3. One may say that the "ML detector" is actually the first layer of this setup. In this project our focus is on the detector's output, therefore we consider the RF-based filter as the first layer.

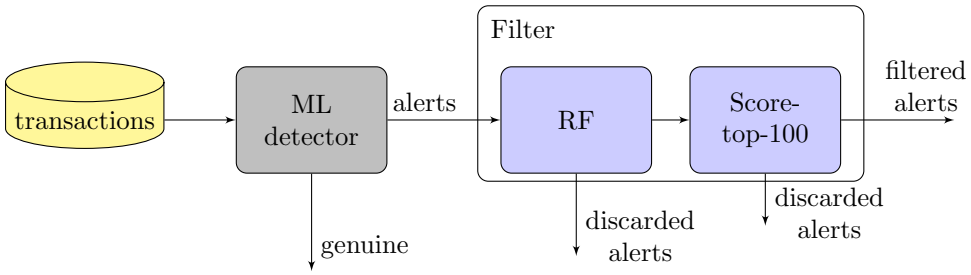


Figure 7.3: Overview of layered filtering approach experiment. The RF-based filter is the first layer and propagates the filtered alerts to the score-based filter.

7.3 Results

A Venn diagram of the filtered TPs of a layered and non-layered filter can be constructed again as earlier. This has been depicted in Figure 7.4 for the *score-top-100* filter. From this follows that

$74 - 32 = 42$ more TPs are filtered when *RF-Desc* is performed first. From a comparison with Figure 7.1b follows that the second layer discards $222 - 74 = 148$ more TPs.

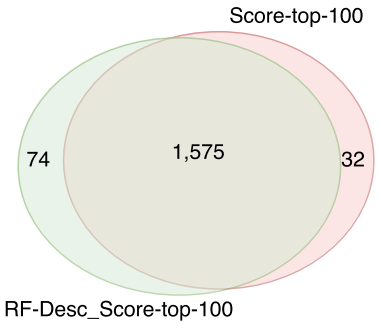


Figure 7.4: Filtered TPs of *score-top-100* filter with and without layered filtering. With the layered approach 32 TPs are not filtered anymore, because the first layer has discarded these alerts. However, the layered approach still filters 42 more TPs than without the first layer.

The results of these layered filters compared to their counterpart without layering can be found in Table 7.3 (and Table B.5) for each month. A few observations can be made from these results. First of all, without modifying the model score values, a sum of 42 more TPs are filtered compared to that without applying the *RF-Desc* filter first. This also means that a total of 42 more FPs are discarded, because the *score-top-100* filter allows a fixed number of alerts. Moreover, for this layered filter it is never the case that less TPs are filtered, since ΔTP in Table 7.3 is always non-negative. The FPR is slightly decreased, but more importantly, the average TPR increases to 0.85 (Table B.5).

The layered filtering also yields 18 more TPs for the score-based filter that uses modified score values (Table 7.3). Recall that the score-based filter discarded 113 more TPs for all months with modified scores (Section 6.4). This means that this layered approach still discards 93 more TPs in total for all months than the filter of Chapter 4.

The classification and corresponding rates of each layer of the first layered system are listed in respectively Table 7.4 and Table 7.5. From these tables follow that the second layer discards a total 148 TPs and 259,427 FPs. The base rate of fraudulent transactions is slightly higher for the second layer. The BDR is increased by more than a factor of 4, but it is still very low. This is because the FPR is still too high compared to the base rates. The BTNR is slightly lower with the second layer because this layer discarded more TPs.

Similar observations can be made for the second layered system, where the second layer uses changed model score values as described in Section 6.4. The classification and corresponding rates of each layer of this layered system are listed in respectively Table 7.6 and Table 7.7. In this layered system, the second layer discards more TPs than without modifying scores (Table 7.4). The second layer of this layered system has therefore also a lower BDR and BTNR.

The performance of the fraud detector and that of several filters in a ROC plane are shown in Figure 7.5. The filters that use modified scores are not included in this figure to improve readability of this figure. The curves of these filters are slightly below the curve of the filter without modifying model scores. Each filter in this figure achieves a slightly better TPR at their corresponding FPR than the ML-Detector at the same FPR. Finally, the layered approach has a better performance than its counterpart without the first layer. This also follows from a comparison of the pAUCs (Table B.5). The layered approaches slightly increase the pAUC of its counterpart without the first layer. Although the layered approach for the modified score-based filtering slightly improves the performance, it still has a worse performance than the filter of Chapter 4.

From this experiment we can conclude that the layered filtering approach works well to improve the performance of another filter. It is obvious that the first layers should have a high TPR.

Table 7.3: Filtered TPs with and without layered filtering per month. Here, the columns "Separate" and "Layered" represent respectively whether the corresponding filter is deployed without or with a first layer. In both layered filtering approaches, the first layer is a *RF-Desc* filter before the score-based filter. The third column (ΔTP) represents the gain of TPs by the layered approach. Note that this ΔTP is different than the measure described in the Evaluation Framework of this thesis (Section 3.2). Both score-based filters improve their performance in a layered filtering approach.

	TP (ΔTP) of *score-top-100*			TP (ΔTP) of *modified-score-top-100*		
Month	Separate	Layered	ΔTP	Separate	Layered	ΔTP
January 2016	45 (−6)	47 (−4)	+2	45 (−6)	47 (−4)	+2
February 2016	38 (−9)	39 (−8)	+1	36 (−11)	37 (−10)	+1
March 2016	86 (−7)	87 (−6)	+1	53 (−40)	54 (−39)	+1
April 2016	110 (−16)	114 (−12)	+4	95 (−31)	97 (−29)	+2
May 2016	120 (−24)	122 (−22)	+2	89 (−55)	93 (−51)	+4
June 2016	117 (−48)	121 (−44)	+4	81 (−84)	84 (−81)	+3
July 2016	66 (−17)	72 (−11)	+6	57 (−26)	58 (−25)	+1
August 2016	86 (−14)	88 (−12)	+2	85 (−15)	85 (−15)	0
September 2016	46 (−1)	46 (−1)	0	46 (−1)	46 (−1)	0
October 2016	102 (−8)	104 (−6)	+2	101 (−9)	103 (−7)	+2
November 2016	159 (−6)	162 (−3)	+3	157 (−8)	162 (−3)	+5
December 2016	167 (−6)	169 (−4)	+2	166 (−7)	165 (−8)	−1
January 2017	306 (−5)	309 (−2)	+3	303 (−8)	307 (−4)	+4
February 2017	159 (−95)	169 (−85)	+10	180 (−74)	174 (−80)	−6
All months	1,607 (−262)	1,649 (−220)	+42	1,494 (−375)	1,512 (−357)	+18

Table 7.4: Classification of each layer of *RF-Desc_Score-top-100*. The first layer discards many FPs of the detector and it has not too many FNs. The second layer discards even more of the FPs. The complete layered system discards in total a lot of negatives.

Layer	TN	FN	FP	TP
RF-Desc	623,784	72	298,658	1,797
Score-top-100	259,427	148	39,231	1,649
System	883,211	220	39,231	1,649

Table 7.5: Rates of each layer of *RF-Desc_Score-top-100*. The rates correspond to the classifications presented in Table 7.4. The new base rate $P(F)$ and the BDR increase for the second layer, but these are still very low. Moreover, the second layer has a slightly lower BTNR because it discards some positives.

Layer	$P(F)$	$P(\neg F)$	FPR	TPR	BDR	BTNR
RF-Desc	0.0020	0.9980	0.3238	0.9615	0.0060	0.9999
Score-top-100	0.0060	0.9940	0.1313	0.9176	0.0403	0.9994
System	0.0020	0.9980	0.0425	0.8823	0.0403	0.9997

Table 7.6: Classification of each layer of *RF-Desc_Mod-score-top-100*. The first layer discards many FPs of the detector and it has not too many FNs. The second layer discards even more of the FPs. The complete layered system discards in total a lot of negatives, but also more positives than the other layered system (Table 7.4).

Layer	TN	FN	FP	TP
RF-Desc	623,784	72	298,658	1,797
Modified-score-top-100	259,299	285	39,359	1,512
System	883,083	357	39,359	1,512

Table 7.7: Rates of each layer of *RF-Desc_Mod-score-top-100*. The rates correspond to the classifications presented in Table 7.6. The new base rate $P(F)$ and the BDR increase for the second layer, but these are still very low. Moreover, the second layer has a slightly lower BTNR because it discards some positives.

Layer	$P(F)$	$P(\neg F)$	FPR	TPR	BDR	BTNR
RF-Desc	0.0020	0.9980	0.3238	0.9615	0.0060	0.9999
Modified-score-top-100	0.0060	0.9940	0.1318	0.8414	0.0370	0.9989
System	0.0020	0.9980	0.0427	0.8090	0.0370	0.9996

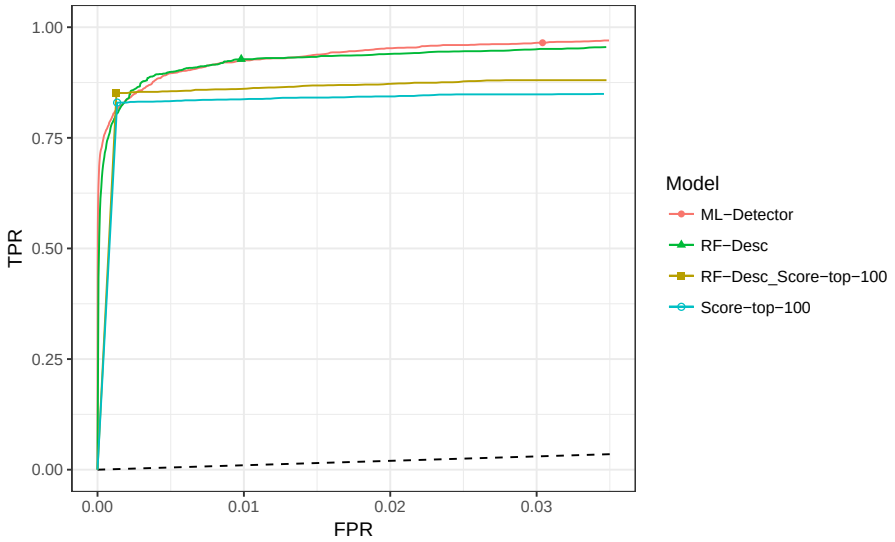


Figure 7.5: Performance of fraud detector and filters in ROC plane. The X-axis (FPR) is cut off to only visualize the curve in the range of low FPRs. The dashed line represents the diagonal of random guesses. The filters that use modified scores are not included in this figure to improve readability of this figure. The curves of these filters are slightly below the curve of the filter without modifying model scores (Figure 6.6). Each filter achieves a slightly better TPR at their corresponding FPR than the ML-Detector at the same FPR. The layered approach also has a better performance than its counterpart without the first layer.

7.4 Conclusion

In this chapter we have analyzed whether our two best filters discarded the same alerts. From this followed that the *RF-Desc* filter (Section 6.3) filtered more TPs than the *score-top-100* filter (Section 4.2). This RF-based filter also discards a lot of FPs that were not discarded by the score-based filter. These alerts thus have relatively high model scores. The score-based filter is able to filter other alerts that have relatively high model scores because of this.

This chapter discusses the use of a layered filtering approach such that output of a layer is used as the input of the next layer (Section 7.1). Then each layer operates on a set of alerts with significantly changed base rate of non-fraudulent transactions.

In the experiment of this chapter, the *RF-Desc* filter is used as the first layer and it propagates its output to the *score-top-100* filter (Section 7.2). Similarly, the *modified-score-top-100* filter (Section 6.4) is applied after the same RF-based filter. From the experiment (Section 7.3) followed that both score-based filters achieve a better performance as a layer for most months. To summarize the results, in the layered approach, the *score-top-100* filter discards 42 less TPs in total. This filter also never discards more TPs for any month than without the first layer. The improvement for the *modified-score-top-100* filter are only 18 additional TPs. The average FPR slightly decreases and the average TPR increases in both layered systems.

A layered filtering approach is thus a good method to improve the filtering performance with imbalanced data. The first layers should obviously have minimal FNs. The experiment of this chapter was good, because the first layer (*RF-Desc* filter) has a relatively high TPR for each month (Table B.5), and yet it still discards many FPs of the detector. Furthermore, it has a performance close to that of the detector while it also has different predictions. This reduction of FPs included reduction of FPs that were not discarded by the *score-top-100* filter, i.e. alerts that have relatively high model scores. The score-based filter could therefore include different alerts in its decision making as the second layer. In our experiments, the BDR is still very low due to the high FPR. It is still difficult to maximize the BDR. The performance may improve by adding additional layers of filters before the last layer.

Chapter 8

Conclusions & Future Work

The goal of this project was to develop a technique that can be used as a filter to automatically discard FPs of the ML-Detector while conserving most of the true positives. In Section 8.1 we summarize how this goal has been achieved and our main contributions are described in Section 8.2. Threats to validity of this research are described in Section 8.3. The possibilities for future research are described in Section 8.4.

8.1 Conclusions

For large dataset with 30 million samples, the ML-Detector has detected 1,869 positives and it has also a lot of FPs (922,442). Moreover, it has an average high TPR of 0.96 and an average FPR of 0.0304 for the 14 months of data which is too high. Several techniques have been used to reduce this high number of FPs.

Our first filtering approach used the probability that a transaction is fraudulent such that the number of alerts per day does not exceed a threshold (Chapter 4). It is a simple approach to decrease the number of FPs, but note that it is different than another instance of the detector that has a higher prediction threshold. This score-based filter could be compared by an instance that has a continuously changing prediction threshold. The filter had a slightly better TPR than the detector could achieve for the same FPR. From Table B.5 follows that on the provided benchmark dataset we shall expect one less detected positive for every reduction of ∼3,365 FPs with this score-based filter approach.

Since this score-based filter depends on the detector being a good ranker, we have explored the data for patterns that are useful in filtering. We have proposed an "Anomaly Description Mining Framework" that uses Frequent Itemset Mining (FIM) to mine descriptions of features and values with a low minimum support. This framework can then be used to obtain "FP-descriptions" and "TP-descriptions" that can distinguish classes of alerts by providing a few thresholds. It uses the *support difference* similar to [Bay and Pazzani, 2001] to find these descriptions with contrasting properties between the classes of alerts. Moreover, the framework creates a few visualizations that can be used to qualitative analyze a few characteristics of the descriptions. We have seen that there are not many descriptions with a high absolute support difference. This framework has been used to obtain several descriptions. Our FP-descriptions describe only a few TPs, but some of the descriptions made no sense to a domain expert. The TP-descriptions on the other hand did make sense to a domain expert. However, their too short description length makes them too general to describe fraud. A few optimalizations have to be made in order to find longer and more specific descriptions. This is described in more detail in the last section of this chapter (Section 8.4).

These descriptions have been used in several experiments to reduce false alerts (Chapter 6). The filters that directly used the descriptions to discard alerts had worse performance than the detector could achieve. The method to alter predictions of the detector before performing the score-based filter of Chapter 4 improved the filtering performance for February 2017, for which

the detector is a worse ranker compared to the other months. This method however decreased the overall performance of the score-based filter and has a lower TPR than the detector has for the same FPR. These experiments could be unsuccessful because of several reasons. First of all, the descriptions are too general as stated before. This leads to larger subgroups that consists of both classes of alerts. Moreover, the TP-descriptions descriptions together describe not all cases of fraud. This leads to discarding the other "groups" of the positives. Lastly, the descriptions have been mined from data with a smaller time window, which could lead to discarding more positives that lie outside the mining period.

A RandomForest has been trained for which the feature space was limited by the information of the descriptions during the training of the model. We consider it a good filter, because it was able to discard a lot of FPs while maintaining a very high TPR. This filter had a similar prediction performance as the model of the detector. More than half of all TPs discarded by this filter belonged to February 2017 (Table B.5), which lies outside the training period. It does not discard as many FPs as our first score-based filter, but it has a very high TPR. From Table B.5 follows that on the provided benchmark dataset we shall expect one less detected positive for every reduction of ∼8,664 FPs with this RF-based filter approach.

From an analysis followed that the RF-based filter discarded FPs that were not discarded by the score-based filter. A layered filtering approach has been used to combine the RF-based filter and the score-based filter as a sequence of filters (Chapter 7). Each filter then propagates its output as input to next filter in the sequence. The next filter then has a higher base rate for fraud to operate on, because the filter before discards a lot of negatives. From Table B.5 follows that on the provided benchmark dataset we shall expect one less detected positive for every reduction of ∼4,015 FPs with this layered filtering approach. This is an improvement compared to our first filtering approach.

The goal of this project has been achieved with several filters. In particular, the RF-based filter and the layered filter of RF-based followed by the score-based filter are our best filters. There are however still opportunities for improvements, which are described in the last section of this chapter.

8.2 Main contributions

In this thesis, we have introduced novel techniques of reducing false alerts in domain of online banking, because there are not many papers about false alert reduction in public literature (Section 2.2).

Additionally, we have introduced an "Anomaly Description Mining Framework" to mine descriptions that have contrasting properties between FPs and TPs of the detector. In particular, a methodology was introduced to analyze these descriptions with several visualizations.

8.3 Threats to validity

There are however a few threats to validity of our research. First of all, we have assumed that the delay of labeling is immediate (Section 1.2), while in reality a sample is labeled by domain experts after an investigation (as shown in Figure 1.3).

In order to simplify the problem, only one instance of a model of the detector has been used in *offline settings* (Section 3.4). In *online settings* several models are used or a model is incrementally updated to learn new behavior as more information becomes available. Moreover, in these situations the data arrives in chunks of streams.

Finally, there are few threats regarding the dataset. The genuine transactions of the dataset used in this project have been downsampled. Reduction in practice may discard more of the positives to achieve the same number of alerts per day. Furthermore, our experiments use one source of data, i.e. the output of the detector (Section 3.2). This could introduce overfitting.

8.4 Future work

Tasks for future work of this research are described in this section. These tasks are grouped and ordered by the structure of this thesis. A few tasks can be described with respect to the evaluation framework and the score-based filter:

- A single instance of the detector with a fixed prediction threshold has been used in this project. Research might be performed for different thresholds such that the detector has less FPs and still a relatively high TPR. This does not affect the filtering performance of our score-based filter as long as the threshold is not too high. However, the description-based filters are affected and FIM could yield different supports for descriptions from the data of alerts.
- As described in previous section, the data arrives in chunks in *online settings*. Our implementation of the *score-top-k* filter does not work in these "streaming settings" as described in Chapter 4. It is possible to design a streaming algorithm that uses the knowledge of historic data to maintain a prediction threshold.
- The filters update the predictions of alerts with either a 0 or a 1. This results with ROC curves that have a low slope and a fast increasing FPR, because many of the false alerts are not discarded and have a new prediction of 1. In addition, the AUC is therefore also lower. Filters might use a different prediction scheme to allow a prediction threshold and a better analysis.
- The score-based filter is model-agnostic, because it relies a model that is a good ranker (Chapter 4). A comparison of different models with the score-based filter might be researched.

Furthermore, several tasks can be described to research with respect to our Anomaly Description Mining Framework:

- As stated in the conclusion of Chapter 5, our framework has difficulties with larger descriptions for large volumes of data. Data pre-processing techniques might be able to decrease the search space used by the mining algorithms. Research has been done to speed up mining algorithms. For instance, association rule mining has been improved with a new data structure [Luna et al., 2016]. A different paper describes several algorithms that significantly decreased the number of frequent itemsets [Siebes et al., 2006].
- In this project, we have only used *false positives vs. true positives* to mine contrasting descriptions (Section 5.3). Different options might be explored in future research. For instance, *negatives vs. positives* could be a good option to learn contrasting descriptions.
- It has been stated in the conclusion that the TP-descriptions are nice but too general to describe fraud. The information of these descriptions might be used to decrease the search space of another FIM-iteration to mine longer descriptions.
- Our descriptions are obtained using a *support difference*. Different metrics of interestingness might be used or combined to find different descriptions. For instance, Emerging Pattern Mining (EPM) (Section 2.2.1) might find interesting descriptions.
- Exceptional Model Mining (EMM) [Duivesteijn et al., 2016] might be used to find descriptions of small and interesting groups of alerts that could be useful to filtering.

Finally, several tasks can be described to research with respect to the layered filtering approach:

- The RF-based filter of our filtering experiments uses the default prediction threshold. Different prediction thresholds could be used, which then yields an instance with a lower FPR and a relatively high but lower TPR. This might improve the performance of the layered filtering approaches.

- The two layers in our layered filtering experiments were manually selected and compared with a Venn diagram to analyze the *disagreement* between the filters. A learning approach to select and evaluate combinations of filters that have a high disagreement might be researched.

Bibliography

Aisha Abdallah, Mohd Aizaini Maarof, and Anazida Zainal. Fraud detection system: A survey. *Journal of Network and Computer Applications*, 68:90–113, 2016. 11

Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. 2015. 10, 11

Stefan Axelsson. The base-rate fallacy and the difficulty of intrusion detection. *ACM Transactions on Information and System Security (TISSEC)*, 3(3):186–205, 2000. 50

Stephen D Bay and Michael J Pazzani. Detecting group differences: Mining contrast sets. *Data mining and knowledge discovery*, 5(3):213–246, 2001. 30, 32, 57

Siddhartha Bhattacharyya, Sanjeev Jha, Kurian Tharakunnel, and J Christopher Westland. Data mining for credit card fraud: A comparative study. *Decision Support Systems*, 50(3): 602–613, 2011. 11, 12

Richard J Bolton, David J Hand, et al. Unsupervised profiling methods for fraud detection. *Credit Scoring and Credit Control VII*, pages 235–255, 2001. 10, 11

Christian Borgelt. Efficient implementations of apriori and eclat. In *FIMI03: Proceedings of the IEEE ICDM workshop on frequent itemset mining implementations*, 2003. 30

Andrea Dal Pozzolo, Olivier Caelen, Yann-Ael Le Borgne, Serge Waterschoot, and Gianluca Bontempi. Learned lessons in credit card fraud detection from a practitioner perspective. *Expert systems with applications*, 41(10):4915–4928, 2014. 3, 4, 11, 12

Guozhu Dong and Jinyan Li. Efficient mining of emerging patterns: Discovering trends and differences. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 43–52. ACM, 1999. 13, 31

Wouter Duivesteijn, Ad J Feelders, and Arno Knobbe. Exceptional model mining. *Data Mining and Knowledge Discovery*, 30(1):47–98, 2016. 59

Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006. 17

João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, 46(4):44, 2014. 9

Martin Grill, Tomáš Pevnỳ, and Martin Rehak. Reducing false positives of network anomaly detection by local adaptive multivariate smoothing. *Journal of Computer and System Sciences*, 83(1):43–57, 2017. 13

Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449, 2002. 10

Yuval Kluger, Ronen Basri, Joseph T Chang, and Mark Gerstein. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome research*, 13(4):703–716, 2003. 30

Stephan Kovach and Wilson Vicente Ruggiero. Online banking fraud detection based on local and global behavior. In *Proc. of the Fifth International Conference on Digital Society, Guadeloupe, France*, pages 166–171, 2011. 11, 12

Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R news*, 2(3): 18–22, 2002. 12

José María Luna, Alberto Cano, Mykola Pechenizkiy, and Sebastián Ventura. Speeding-up association rule mining with inverted index compression. *IEEE transactions on cybernetics*, 46 (12):3059–3072, 2016. 59

Donna Katzman McClish. Analyzing a portion of the roc curve. *Medical Decision Making*, 9(3): 190–195, 1989. 17

Raghavendra Patidar, Lokesh Sharma, et al. Credit card fraud detection using neural network. *International Journal of Soft Computing and Engineering (IJSCE)*, 1(32-38), 2011. 11, 12

Clifton Phua, Damminda Alahakoon, and Vincent Lee. Minority report in fraud detection: classification of skewed data. *Acm sigkdd explorations newsletter*, 6(1):50–59, 2004. 17

Tadeusz Pietraszek. Using adaptive alert classification to reduce false positives in intrusion detection. In *International Workshop on Recent Advances in Intrusion Detection*, pages 102–124. Springer, 2004. 14

Tadeusz Pietraszek and Axel Tanner. Data mining and machine learningtowards reducing false positives in intrusion detection. *Information security technical report*, 10(3):169–183, 2005. 14

Rafał Pokrywka. Reducing false alarm rate in anomaly detection with layered filtering. In *International Conference on Computational Science*, pages 396–404. Springer, 2008. 50

Daniel Sánchez, MA Vila, L Cerda, and José-María Serrano. Association rules applied to credit card fraud detection. *Expert systems with applications*, 36(2):3630–3640, 2009. 11

Riyanarto Sarno, Rahadian Dustrial Dewandono, Tohari Ahmad, Mohammad Farid Naufal, and Fernandes Sinaga. Hybrid association rule learning and process mining for fraud detection. *IAENG International Journal of Computer Science*, 42(2):59–72, 2015. 11, 12

Arno Siebes, Jilles Vreeken, and Matthijs van Leeuwen. Item sets that compress. In *Proceedings of the 2006 SIAM International Conference on Data Mining*, pages 395–406. SIAM, 2006. 59

Georgios P Spathoulas and Sokratis K Katsikas. Reducing false positives in intrusion detection systems. *computers & security*, 29(1):35–44, 2010. 13

Abhinav Srivastava, Amlan Kundu, Shamik Sural, and Arun Majumdar. Credit card fraud detection using hidden markov model. *IEEE Transactions on dependable and secure computing*, 5(1):37–48, 2008. 10, 11

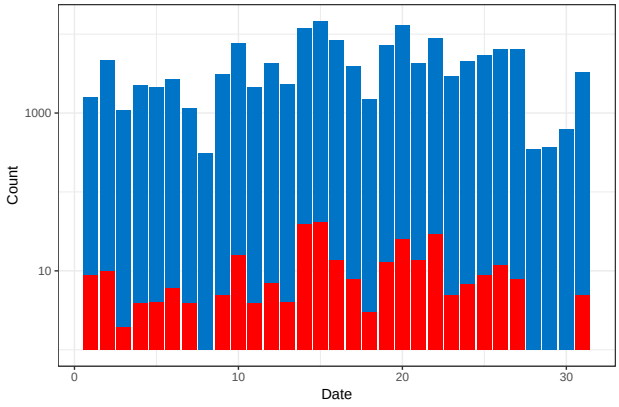
Wei Wei, Jinjiu Li, Longbing Cao, Yuming Ou, and Jiahang Chen. Effective detection of sophisticated online banking fraud on extremely imbalanced data. *World Wide Web*, 16(4): 449–475, 2013. 3, 9, 10, 11, 12, 13, 31

Appendix A

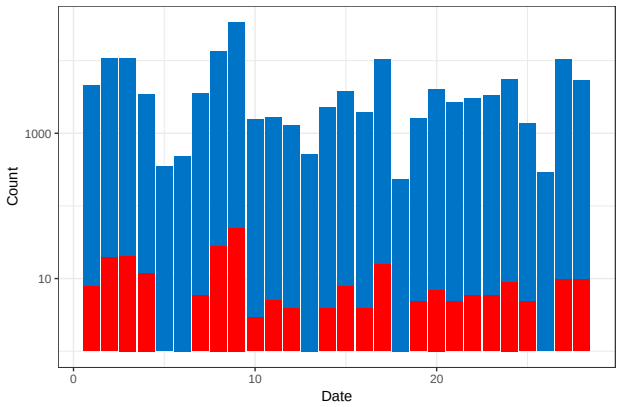
Figures

In this thesis, several figures have been shown. However, it is not always the case that the small images are readable. Those small images are also visualized here with a greater size.

A.1 Figures in Chapter 3



(a) January 2017



(b) February 2017

Figure A.1: Number of alerts in 2017 per day. The X-axis represents the date and Y-axis represents the number of alerts with a logarithmic scale. The colors red and blue are used to represent fraudulent and genuine cases respectively. Moreover, there are less alerts during weekends.

A.2 Figures in Chapter 6

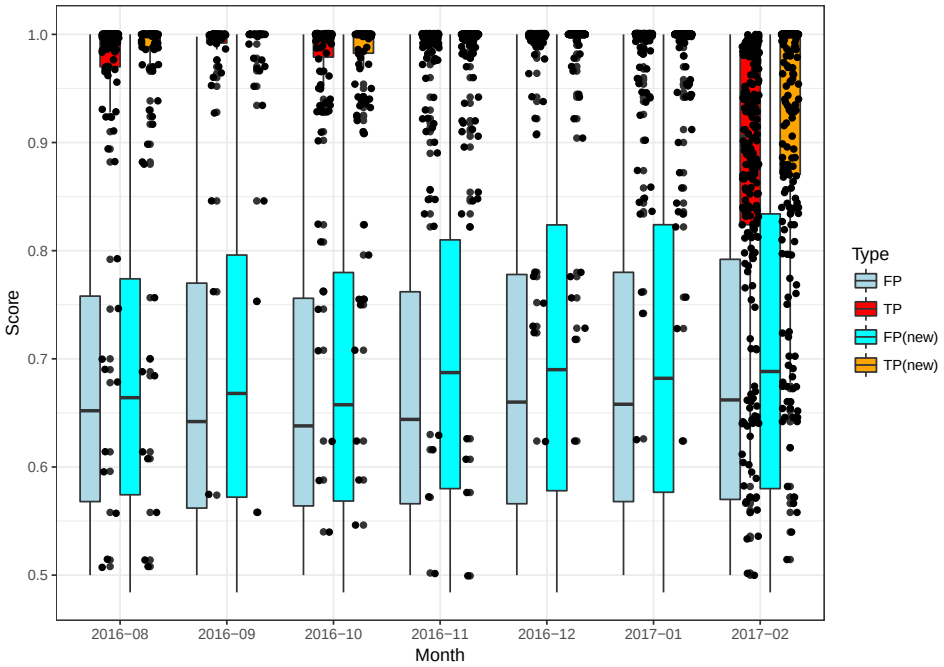


Figure A.2: Changes in distribution of scores per class for different months. For each month, the original distribution (Figure 4.1) is depicted next to the new distribution of score values. The lowest score value for FPs slightly decreases for all months. Moreover, all boxplots shift to the top of the figure.

Appendix B

Results

B.1 Mined descriptions

Descriptions obtained with the Anomaly Description Mining Framework experiments described in Section 5.8 are listed here with corresponding values for supports.

B.1.1 FP-descriptions

Table B.1: Descriptions with low support for TPs and high support difference. These descriptions are obtained with our Anomaly Description Mining Framework and the corresponding results are described in Section 5.8.1.

Description	ID	$supp_{\text{FP}}$	$supp_{\text{TP}}$	$supp_{\text{diff}}$
{C_1055=0.124, C_0082=0.167}	FP1	0.359	0.0574	0.3016
{C_1055=0.124, C_1023=0.701}	FP2	0.359	0.0574	0.3016
{C_0116=0.104, C_0908=0.15}	FP3	0.2997	0.0508	0.2489
{C_0116=0.104, C_1012=0.351}	FP4	0.2997	0.0508	0.2489
{C_0116=0.104, C_0296=0.701}	FP5	0.2997	0.0508	0.2489
{C_1055=0.124, C_0807=0.701}	FP6	0.2887	0.0519	0.2368
{C_0228=0.5, C_1055=0.124}	FP7	0.2887	0.0519	0.2368
{C_1055=0.124, C_0501=0.767}	FP8	0.2887	0.0519	0.2368
{C_0231=0.834, C_1055=0.124}	FP9	0.2887	0.0519	0.2368
{C_0238=0.501, C_1055=0.124}	FP10	0.2887	0.0519	0.2368
{C_1055=0.124, C_0182=0.701}	FP11	0.2871	0.0552	0.2319
{C_1055=0.124, C_0887=0.3}	FP12	0.28	0.0552	0.2248
{V_0425=0.229, C_0485=0.628}	FP13	0.257	0.0596	0.1974
{V_0330=0.795, C_0068=0.129}	FP14	0.2443	0.0519	0.1924
{V_0330=0.795, C_0069=0.136}	FP15	0.2448	0.0541	0.1907

B.1.2 TP-descriptions

Table B.2: Descriptions with low support for FPs and high negative support difference. These descriptions are obtained with our Anomaly Description Mining Framework and the corresponding results are described in Section 5.8.2.

Description	ID	$supp_{\text{FP}}$	$supp_{\text{TP}}$	$supp_{\text{diff}}$
{C_0487=0.678, C_0556=0.5}	TP1	0.1287	0.4194	−0.2907
{C_0809=0.104, C_0447=0.701}	TP2	0.1145	0.4062	−0.2917
{C_0082=0.3, C_0317=0.3}	TP3	0.1287	0.4205	−0.2919
{C_0201=0.701, C_0191=0.3}	TP4	0.1073	0.4018	−0.2945
{C_0201=0.701, C_1033=0.3}	TP5	0.1073	0.4018	−0.2945
{C_0201=0.701, C_1028=0.107}	TP6	0.1234	0.4205	−0.2971
{C_0433=0.3, C_0556=0.5}	TP7	0.1052	0.404	−0.2987
{C_0201=0.701, C_1085=0.3}	TP8	0.1294	0.4305	−0.301
{C_0201=0.701, C_0644=0.11}	TP9	0.1103	0.4117	−0.3014
{C_0201=0.701, C_0573=0.3}	TP10	0.1071	0.4095	−0.3024
{C_1068=0.701, C_0556=0.5}	TP11	0.1169	0.4205	−0.3037
{C_0191=0.3, C_0556=0.5}	TP12	0.1286	0.4338	−0.3052
{C_1033=0.3, C_0556=0.5}	TP13	0.1286	0.4338	−0.3052
{C_1018=0.3, C_0556=0.5}	TP14	0.1218	0.4272	−0.3053
{N_1059=0.95, C_0556=0.5}	TP15	0.1052	0.4117	−0.3065
{C_1083=0.3, C_0201=0.701}	TP16	0.1287	0.436	−0.3073
{C_1040=0.3, C_0201=0.701}	TP17	0.1289	0.4393	−0.3104
{C_0201=0.701, C_0556=0.5}	TP18	0.1207	0.4316	−0.3108
{C_0201=0.701, C_1018=0.3}	TP19	0.1158	0.4283	−0.3125
{C_0644=0.11, C_0556=0.5}	TP20	0.1274	0.4404	−0.313
{C_1070=0.701, C_0556=0.5}	TP21	0.1179	0.4338	−0.3159
{C_0151=0.701, C_0653=0.701}	TP22	0.1164	0.4327	−0.3163
{C_0201=0.701, C_1049=0.701}	TP23	0.1226	0.4426	−0.32
{C_0201=0.701, C_1046=0.701}	TP24	0.1168	0.4371	−0.3203
{C_0201=0.701, V_0331=0.422}	TP25	0.129	0.4503	−0.3214
{C_1049=0.701, C_0556=0.5}	TP26	0.1279	0.4503	−0.3224
{C_1046=0.701, C_0556=0.5}	TP27	0.1201	0.4481	−0.328
{C_0082=0.3, C_0235=0.701}	TP28	0.1289	0.4592	−0.3303
{C_0201=0.701, N_0263=0.95}	TP29	0.1246	0.4592	−0.3346
{C_0201=0.701, C_0653=0.701}	TP30	0.1057	0.4448	−0.3391
{C_0299=0.3, C_0556=0.5}	TP31	0.1156	0.4547	−0.3391
{C_0151=0.701, C_1023=0.3}	TP32	0.1039	0.4437	−0.3398
{C_0201=0.701, V_1060=0.379}	TP33	0.1296	0.4724	−0.3428

B.2 Score modifier grid search

Table B.3: Modified score filtering with a wide grid search for score modifiers. The score modifier M_{TP} has more impact on the score-based filter than the score modifier M_{FP}. A larger grid search with smaller values for the score modifiers is necessary to find better values.

Month	ΔFP	ΔTP	M_{FP}	M_{TP}	$\sum_{\Delta\text{TP}}$
December 2016	0	0	0.0	0.0	0
January 2017	0	0	⋮	⋮	⋮
February 2017	0	0			
December 2016	111	−111	0.0	0.1	−386
January 2017	151	−151	⋮	⋮	⋮
February 2017	124	−124			
December 2016	1	−1	0.1	0.0	−4
January 2017	4	−4	⋮	⋮	⋮
February 2017	−1	1			
December 2016	111	−111	0.1	0.1	−380
January 2017	152	−152	⋮	⋮	⋮
February 2017	117	−117			
December 2016	4	−4	0.2	0.0	−13
January 2017	9	−9	⋮	⋮	⋮
February 2017	0	0			
December 2016	111	−111	0.2	0.1	−379
January 2017	152	−152	⋮	⋮	⋮
February 2017	116	−116			
December 2016	8	−8	0.3	0.0	−19
January 2017	12	−12	⋮	⋮	⋮
February 2017	−1	1			
December 2016	112	−112	0.3	0.1	−379
January 2017	152	−152	⋮	⋮	⋮
February 2017	115	−115			
December 2016	9	−9	0.4	0.0	−26
January 2017	16	−16	⋮	⋮	⋮
February 2017	1	−1			
December 2016	112	−112	0.4	0.1	−377
January 2017	150	−150	⋮	⋮	⋮
February 2017	115	−115			

Table B.4: Modified score filtering with a narrow grid search for score modifiers. The score modifier M_{TP} has more impact on the score-based filter than the score modifier M_{FP}. A relatively low M_{TP} yields a positive $\sum_{\Delta\text{TP}}$ during August up to February. The sum is 13 and maximal for $M_{\text{FP}} = 0 \wedge M_{\text{TP}} = 0.02$.

Month	ΔFP	ΔTP	M_{FP}	M_{TP}	$\sum_{\Delta\text{TP}}$
August 2016	0	0	0.0	0.0	0
September 2016	0	0			
October 2016	0	0			
November 2016	0	0	⋮	⋮	⋮
December 2016	0	0			
January 2017	0	0			
February 2017	0	0			
August 2016	−1	1	0.0	0.005	10
September 2016	0	0			
October 2016	0	0			
November 2016	0	0	⋮	⋮	⋮
December 2016	0	0			
January 2017	1	−1			
February 2017	−10	10			
August 2016	0	0	0.0	0.01	10
September 2016	0	0			
October 2016	1	−1			
November 2016	1	−1	⋮	⋮	⋮
December 2016	0	0			
January 2017	2	−2			
February 2017	−14	14			
August 2016	1	−1	0.0	0.015	11
September 2016	0	0			
October 2016	1	−1			
November 2016	2	−2	⋮	⋮	⋮
December 2016	1	−1			
January 2017	3	−3			
February 2017	−19	19			
August 2016	1	−1	0.0	0.02	13
September 2016	0	0			
October 2016	1	−1			
November 2016	2	−2	⋮	⋮	⋮
December 2016	1	−1			
January 2017	3	−3			
February 2017	−21	21			
August 2016	6	−6	0.0	0.025	6
September 2016	0	0			
October 2016	1	−1			
November 2016	3	−3	⋮	⋮	⋮
December 2016	1	−1			
January 2017	4	−4			
February 2017	−21	21			

Month	ΔFP	ΔTP	M_{FP}	M_{TP}	$\sum_{\Delta\text{TP}}$
August 2016	7	−7	0.0	0.03	6
September 2016	0	0			
October 2016	1	−1			
November 2016	4	−4	⋮	⋮	⋮
December 2016	1	−1			
January 2017	5	−5			
February 2017	−24	24			
August 2016	0	0	0.005	0.0	0
September 2016	0	0			
October 2016	1	−1			
November 2016	0	0	⋮	⋮	⋮
December 2016	0	0			
January 2017	1	−1			
February 2017	−2	2			
August 2016	0	0	0.005	0.005	6
September 2016	0	0			
October 2016	1	−1			
November 2016	0	0	⋮	⋮	⋮
December 2016	0	0			
January 2017	1	−1			
February 2017	−8	8			
August 2016	0	0	0.005	0.01	9
September 2016	0	0			
October 2016	1	−1			
November 2016	1	−1	⋮	⋮	⋮
December 2016	0	0			
January 2017	2	−2			
February 2017	−13	13			
August 2016	0	0	0.005	0.015	11
September 2016	0	0			
October 2016	1	−1			
November 2016	2	−2	⋮	⋮	⋮
December 2016	1	−1			
January 2017	3	−3			
February 2017	−18	18			
August 2016	1	−1	0.005	0.02	11
September 2016	0	0			
October 2016	1	−1			
November 2016	2	−2	⋮	⋮	⋮
December 2016	1	−1			
January 2017	3	−3			
February 2017	−19	19			
August 2016	6	−6	0.005	0.025	6
September 2016	0	0			
October 2016	1	−1			
November 2016	3	−3	⋮	⋮	⋮
December 2016	1	−1			
January 2017	4	−4			
February 2017	−21	21			

Month	ΔFP	ΔTP	M_{FP}	M_{TP}	$\sum_{\Delta\text{TP}}$
August 2016	7	−7	0.005	0.03	6
September 2016	0	0			
October 2016	1	−1			
November 2016	4	−4	⋮	⋮	⋮
December 2016	1	−1			
January 2017	5	−5			
February 2017	−24	24			
August 2016	0	0	0.01	0.0	1
September 2016	0	0			
October 2016	1	−1			
November 2016	0	0	⋮	⋮	⋮
December 2016	0	0			
January 2017	1	−1			
February 2017	−3	3			
August 2016	0	0	0.01	0.005	6
September 2016	0	0			
October 2016	1	−1			
November 2016	0	0	⋮	⋮	⋮
December 2016	0	0			
January 2017	1	−1			
February 2017	−8	8			
August 2016	0	0	0.01	0.01	10
September 2016	0	0			
October 2016	1	−1			
November 2016	1	−1	⋮	⋮	⋮
December 2016	0	0			
January 2017	2	−2			
February 2017	−14	14			
August 2016	0	0	0.01	0.015	12
September 2016	0	0			
October 2016	1	−1			
November 2016	1	−1	⋮	⋮	⋮
December 2016	1	−1			
January 2017	3	−3			
February 2017	−18	18			
August 2016	1	−1	0.01	0.02	13
September 2016	0	0			
October 2016	1	−1			
November 2016	2	−2	⋮	⋮	⋮
December 2016	1	−1			
January 2017	3	−3			
February 2017	−21	21			
August 2016	6	−6	0.01	0.025	6
September 2016	0	0			
October 2016	1	−1			
November 2016	3	−3	⋮	⋮	⋮
December 2016	1	−1			
January 2017	4	−4			
February 2017	−21	21			

Month	ΔFP	ΔTP	M_{FP}	M_{TP}	$\sum_{\Delta\text{TP}}$
August 2016	7	−7	0.01	0.03	6
September 2016	0	0			
October 2016	1	−1			
November 2016	4	−4			
December 2016	1	−1	⋮	⋮	⋮
January 2017	5	−5			
February 2017	−24	24			
August 2016	0	0	0.015	0.0	2
September 2016	0	0			
October 2016	1	−1			
November 2016	0	0			
December 2016	0	0	⋮	⋮	⋮
January 2017	1	−1			
February 2017	−4	4			
August 2016	0	0	0.015	0.005	9
September 2016	0	0			
October 2016	1	−1			
November 2016	0	0			
December 2016	0	0	⋮	⋮	⋮
January 2017	1	−1			
February 2017	−11	11			
August 2016	0	0	0.015	0.01	10
September 2016	0	0			
October 2016	1	−1			
November 2016	1	−1			
December 2016	0	0	⋮	⋮	⋮
January 2017	2	−2			
February 2017	−14	14			
August 2016	0	0	0.015	0.015	12
September 2016	0	0			
October 2016	1	−1			
November 2016	1	−1			
December 2016	1	−1	⋮	⋮	⋮
January 2017	3	−3			
February 2017	−18	18			
August 2016	1	−1	0.015	0.02	12
September 2016	0	0			
October 2016	1	−1			
November 2016	2	−2			
December 2016	1	−1	⋮	⋮	⋮
January 2017	3	−3			
February 2017	−20	20			
August 2016	6	−6	0.015	0.025	7
September 2016	0	0			
October 2016	1	−1			
November 2016	3	−3			
December 2016	1	−1	⋮	⋮	⋮
January 2017	3	−3			
February 2017	−21	21			

Month	ΔFP	ΔTP	M_{FP}	M_{TP}	$\sum_{\Delta\text{TP}}$
August 2016	6	−6	0.015	0.03	6
September 2016	0	0			
October 2016	1	−1			
November 2016	4	−4	⋮	⋮	⋮
December 2016	1	−1			
January 2017	5	−5			
February 2017	−23	23			
August 2016	0	0	0.02	0.0	3
September 2016	0	0			
October 2016	1	−1			
November 2016	−1	1	⋮	⋮	⋮
December 2016	0	0			
January 2017	1	−1			
February 2017	−4	4			
August 2016	0	0	0.02	0.005	10
September 2016	0	0			
October 2016	1	−1			
November 2016	0	0	⋮	⋮	⋮
December 2016	0	0			
January 2017	1	−1			
February 2017	−12	12			
August 2016	1	−1	0.02	0.01	9
September 2016	0	0			
October 2016	1	−1			
November 2016	1	−1	⋮	⋮	⋮
December 2016	0	0			
January 2017	2	−2			
February 2017	−14	14			
August 2016	0	0	0.02	0.015	12
September 2016	0	0			
October 2016	1	−1			
November 2016	1	−1	⋮	⋮	⋮
December 2016	1	−1			
January 2017	3	−3			
February 2017	−18	18			
August 2016	1	−1	0.02	0.02	12
September 2016	0	0			
October 2016	1	−1			
November 2016	2	−2	⋮	⋮	⋮
December 2016	1	−1			
January 2017	3	−3			
February 2017	−20	20			
August 2016	6	−6	0.02	0.025	7
September 2016	0	0			
October 2016	1	−1			
November 2016	3	−3	⋮	⋮	⋮
December 2016	1	−1			
January 2017	3	−3			
February 2017	−21	21			

Month	ΔFP	ΔTP	M_{FP}	M_{TP}	$\sum_{\Delta\text{TP}}$
August 2016	6	−6	0.02	0.03	6
September 2016	0	0			
October 2016	1	−1			
November 2016	4	−4	⋮	⋮	⋮
December 2016	1	−1			
January 2017	5	−5			
February 2017	−23	23			

B.3 Filter results

Table B.5: Results of ML-Detector and reduction of all filters for each month. The partial AUC is computed for the region of FPR $\leqslant 0.035$ as described in Chapter 3. Each filter reduces the average FPR a lot and still have an average TPR higher than 0.7. The RF-based filter has on average the lowest reduction of TPR while it reduces the FPR by a factor of 3. The FPR is reduced even more with a layered filtering approach of the RF-based and score-based filters.

Month	Filter	FP (ΔFP)	TP (ΔTP)	FPR	TPR	pAUC
January 2016	ML-Detector	33,657	51	0.01124	0.9808	0.0342
	Score-top-100	3,055 (−30,602)	45 (−6)	0.00102	0.8654	0.0304
	Modified-score-top-100	3,055 (−30,602)	45 (−6)	0.00102	0.8654	0.0304
	Ensemble-FPDs	33,493 (−164)	51 (0)	0.01119	0.9808	0.0293
⋮	Not-TPDs	2,483 (−31,174)	6 (−45)	0.00083	0.1154	0.0045
	RF-Desc	12,780 (−20,877)	51 (0)	0.00427	0.9808	0.0347
	RF-Desc_Score-top-100	2,907 (−30,750)	47 (−4)	0.00097	0.9038	0.0317
	RF-Desc_Mod-score-top-100	2,906 (−30,751)	47 (−4)	0.00097	0.9038	0.0317
February 2016	ML-Detector	83,221	47	0.02839	0.9592	0.0321
	Score-top-100	2,862 (−80,359)	38 (−9)	0.00098	0.7755	0.0271
	Modified-score-top-100	2,864 (−80,357)	36 (−11)	0.00098	0.7347	0.0257
	Ensemble-FPDs	58,976 (−24,245)	40 (−7)	0.02012	0.8163	0.0216
⋮	Not-TPDs	31,862 (−51,359)	25 (−22)	0.01087	0.5102	0.0152
	RF-Desc	27,934 (−55,287)	47 (0)	0.00953	0.9592	0.0330
	RF-Desc_Score-top-100	2,861 (−80,360)	39 (−8)	0.00098	0.7959	0.0277
	RF-Desc_Mod-score-top-100	2,863 (−80,358)	37 (−10)	0.00098	0.7551	0.0263
March 2016	ML-Detector	112,702	93	0.03378	0.9894	0.0340
	Score-top-100	3,014 (−109,688)	86 (−7)	0.00090	0.9149	0.0317
	Modified-score-top-100	3,047 (−109,655)	53 (−40)	0.00091	0.5638	0.0207
	Ensemble-FPDs	84,335 (−28,367)	75 (−18)	0.02528	0.7979	0.0196
⋮	Not-TPDs	44,814 (−67,888)	58 (−35)	0.01343	0.6170	0.0175
	RF-Desc	36,267 (−76,435)	93 (0)	0.01087	0.9894	0.0340
	RF-Desc_Score-top-100	3,013 (−109,689)	87 (−6)	0.00090	0.9255	0.0321
	RF-Desc_Mod-score-top-100	3,046 (−109,656)	54 (−39)	0.00091	0.5745	0.0210
April 2016	ML-Detector	129,163	126	0.03428	0.9921	0.0340
	Score-top-100	2,890 (−126,273)	110 (−16)	0.00077	0.8661	0.0302
	Modified-score-top-100	2,905 (−126,258)	95 (−31)	0.00077	0.7480	0.0272
	Ensemble-FPDs	93,475 (−35,688)	112 (−14)	0.02481	0.8819	0.0226
⋮	Not-TPDs	51,556 (−77,607)	76 (−50)	0.01368	0.5984	0.0170
	RF-Desc	43,701 (−85,462)	126 (0)	0.01160	0.9921	0.0344
	RF-Desc_Score-top-100	2,886 (−126,277)	114 (−12)	0.00077	0.8976	0.0313
	RF-Desc_Mod-score-top-100	2,903 (−126,260)	97 (−29)	0.00077	0.7638	0.0277

Month	Filter	FP (ΔFP)	TP (ΔTP)	FPR	TPR	pAUC
May 2016	ML-Detector	145,964	144	0.03575	0.9600	0.0327
	Score-top-100	2,980 (−142,984)	120 (−24)	0.00073	0.8000	0.0282
	Modified-score-top-100	3,011 (−142,953)	89 (−55)	0.00074	0.5933	0.0213
	Ensemble-FPDs	105,308 (−40,656)	122 (−22)	0.02579	0.8133	0.0188
⋮	Not-TPDs	54,680 (−91,284)	73 (−71)	0.01339	0.4867	0.0141
	RF-Desc	46,431 (−99,533)	142 (−2)	0.01137	0.9467	0.0327
	RF-Desc_Score-top-100	2,978 (−142,986)	122 (−22)	0.00073	0.8133	0.0287
	RF-Desc_Mod-score-top-100	3,007 (−142,957)	93 (−51)	0.00074	0.6200	0.0220
June 2016	ML-Detector	141,350	165	0.03557	0.9706	0.0320
	Score-top-100	2,883 (−138,467)	117 (−48)	0.00073	0.6882	0.0244
	Modified-score-top-100	2,919 (−138,431)	81 (−84)	0.00073	0.4765	0.0173
	Ensemble-FPDs	100,428 (−40,922)	143 (−22)	0.02527	0.8412	0.0202
⋮	Not-TPDs	53,718 (−87,632)	93 (−72)	0.01352	0.5471	0.0157
	RF-Desc	43,893 (−97,457)	154 (−11)	0.01105	0.9059	0.0314
	RF-Desc_Score-top-100	2,879 (−138,471)	121 (−44)	0.00072	0.7118	0.0252
	RF-Desc_Mod-score-top-100	2,916 (−138,434)	84 (−81)	0.00073	0.4941	0.0183
July 2016	ML-Detector	136,628	83	0.03437	1.0000	0.0339
	Score-top-100	3,034 (−133,594)	66 (−17)	0.00076	0.7952	0.0275
	Modified-score-top-100	3,043 (−133,585)	57 (−26)	0.00077	0.6867	0.0238
	Ensemble-FPDs	95,551 (−41,077)	77 (−6)	0.02404	0.9277	0.0244
⋮	Not-TPDs	50,172 (−86,456)	58 (−25)	0.01262	0.6988	0.0200
	RF-Desc	39,853 (−96,775)	81 (−2)	0.01003	0.9759	0.0337
	RF-Desc_Score-top-100	3,028 (−133,600)	72 (−11)	0.00076	0.8675	0.0300
	RF-Desc_Mod-score-top-100	3,042 (−133,586)	58 (−25)	0.00077	0.6988	0.0246
August 2016	ML-Detector	55,563	100	0.03392	0.9709	0.0320
	Score-top-100	3,014 (−52,549)	86 (−14)	0.00184	0.8350	0.0285
	Modified-score-top-100	3,015 (−52,548)	85 (−15)	0.00184	0.8252	0.0281
	Ensemble-FPDs	38,897 (−16,666)	98 (−2)	0.02375	0.9515	0.0255
⋮	Not-TPDs	20,307 (−35,256)	78 (−22)	0.01240	0.7573	0.0218
	RF-Desc	16,847 (−38,716)	99 (−1)	0.01028	0.9612	0.0333
	RF-Desc_Score-top-100	3,012 (−52,551)	88 (−12)	0.00184	0.8544	0.0291
	RF-Desc_Mod-score-top-100	3,015 (−52,548)	85 (−15)	0.00184	0.8252	0.0281
September 2016	ML-Detector	6,450	47	0.01940	0.9216	0.0321
	Score-top-100	2,937 (−3,513)	46 (−1)	0.00884	0.9020	0.0282
	Modified-score-top-100	2,937 (−3,513)	46 (−1)	0.00884	0.9020	0.0282
	Ensemble-FPDs	4,981 (−1,469)	42 (−5)	0.01498	0.8235	0.0252
⋮	Not-TPDs	2,459 (−3,991)	39 (−8)	0.00740	0.7647	0.0246
	RF-Desc	2,306 (−4,144)	46 (−1)	0.00694	0.9020	0.0324
	RF-Desc_Score-top-100	2,123 (−4,327)	46 (−1)	0.00639	0.9020	0.0298
	RF-Desc_Mod-score-top-100	2,119 (−4,331)	46 (−1)	0.00637	0.9020	0.0298
October 2016	ML-Detector	15,466	110	0.02545	0.9821	0.0334
	Score-top-100	2,800 (−12,666)	102 (−8)	0.00461	0.9107	0.0299
	Modified-score-top-100	2,801 (−12,665)	101 (−9)	0.00461	0.9018	0.0296
	Ensemble-FPDs	11,106 (−4,360)	104 (−6)	0.01828	0.9286	0.0261
⋮	Not-TPDs	5,242 (−10,224)	84 (−26)	0.00863	0.7500	0.0231
	RF-Desc	5,109 (−10,357)	109 (−1)	0.00841	0.9732	0.0338
	RF-Desc_Score-top-100	2,662 (−12,804)	104 (−6)	0.00438	0.9286	0.0305
	RF-Desc_Mod-score-top-100	2,660 (−12,806)	103 (−7)	0.00438	0.9196	0.0302

Month	Filter	FP (ΔFP)	TP (ΔTP)	FPR	TPR	pAUC
November 2016	ML-Detector	15,952	165	0.02370	0.9880	0.0340
	Score-top-100	2,841 (−13,111)	159 (−6)	0.00422	0.9521	0.0316
	Modified-score-top-100	2,843 (−13,109)	157 (−8)	0.00422	0.9401	0.0312
	Ensemble-FPDs	12,536 (−3,416)	159 (−6)	0.01862	0.9521	0.0265
⋮	Not-TPDs	7,524 (−8,428)	130 (−35)	0.01118	0.7784	0.0231
	RF-Desc	5,989 (−9,963)	165 (0)	0.00890	0.9880	0.0345
	RF-Desc_Score-top-100	2,802 (−13,150)	162 (−3)	0.00416	0.9701	0.0322
	RF-Desc_Mod-score-top-100	2,802 (−13,150)	162 (−3)	0.00416	0.9701	0.0322
December 2016	ML-Detector	17,213	173	0.02365	0.9774	0.0341
	Score-top-100	2,933 (−14,280)	167 (−6)	0.00403	0.9435	0.0318
	Modified-score-top-100	2,934 (−14,279)	166 (−7)	0.00403	0.9379	0.0316
	Ensemble-FPDs	13,739 (−3,474)	169 (−4)	0.01888	0.9548	0.0268
⋮	Not-TPDs	8,232 (−8,981)	161 (−12)	0.01131	0.9096	0.0272
	RF-Desc	6,579 (−10,634)	171 (−2)	0.00904	0.9661	0.0341
	RF-Desc_Score-top-100	2,839 (−14,374)	169 (−4)	0.00390	0.9548	0.0323
	RF-Desc_Mod-score-top-100	2,843 (−14,370)	165 (−8)	0.00391	0.9322	0.0316
January 2017	ML-Detector	15,241	311	0.02251	0.9936	0.0346
	Score-top-100	2,794 (−12,447)	306 (−5)	0.00413	0.9776	0.0323
	Modified-score-top-100	2,797 (−12,444)	303 (−8)	0.00413	0.9681	0.0320
	Ensemble-FPDs	11,973 (−3,268)	303 (−8)	0.01768	0.9681	0.0280
⋮	Not-TPDs	6,882 (−8,359)	285 (−26)	0.01016	0.9105	0.0273
	RF-Desc	5,514 (−9,727)	311 (0)	0.00814	0.9936	0.0347
	RF-Desc_Score-top-100	2,700 (−12,541)	309 (−2)	0.00399	0.9872	0.0327
	RF-Desc_Mod-score-top-100	2,702 (−12,539)	307 (−4)	0.00399	0.9808	0.0325
February 2017	ML-Detector	13,872	254	0.02232	0.8789	0.0277
	Score-top-100	2,641 (−11,231)	159 (−95)	0.00425	0.5502	0.0191
	Modified-score-top-100	2,620 (−11,252)	180 (−74)	0.00421	0.6228	0.0215
	Ensemble-FPDs	11,135 (−2,737)	247 (−7)	0.01791	0.8547	0.0243
⋮	Not-TPDs	6,371 (−7,501)	215 (−39)	0.01025	0.7439	0.0229
	RF-Desc	5,455 (−8,417)	202 (−52)	0.00878	0.6990	0.0243
	RF-Desc_Score-top-100	2,541 (−11,331)	169 (−85)	0.00409	0.5848	0.0214
	RF-Desc_Mod-score-top-100	2,535 (−11,337)	174 (−80)	0.00408	0.6021	0.0220
All months	ML-Detector	922,442	1,869	0.03041	0.9649	0.0325
	Score-top-100	40,678 (−881,764)	1,607 (−262)	0.00134	0.8296	0.0289
	Modified-score-top-100	40,791 (−881,651)	1,494 (−375)	0.00134	0.7713	0.0269
	Ensemble-FPDs	675,933 (−246,509)	1,742 (−127)	0.02228	0.8993	0.0242
⋮	Not-TPDs	346,302 (−576,140)	1,381 (−488)	0.01141	0.7130	0.0211
	RF-Desc	298,658 (−623,784)	1,797 (−72)	0.00984	0.9277	0.0322
	RF-Desc_Score-top-100	39,231 (−883,211)	1,649 (−220)	0.00129	0.8513	0.0298
	RF-Desc_Mod-score-top-100	39,359 (−883,083)	1,512 (−357)	0.00130	0.7806	0.0276