CHARLES UNIVERSITY

FACULTY OF SOCIAL SCIENCES

Institute of Economic Studies

Machine Learning Methods in Payment Card Fraud Detection

Master's thesis

Author: Bc. Jan Sinčák

Study program: Economics and Finance

Supervisor: doc. PhDr. Jozef Baruník, Ph.D.

Year of defense: 2023

Declaration of Authorship

Abstract

Protection of clients from fraudulent transactions is a complicated task. Banks tend to rely on rule-based systems which require manual creation of rules to identify fraud. These rules have to be set up by employees of the bank who need to look for any trends in fraudulent transactions themselves. This thesis deals with the problem of detection of fraudulent card transactions as it compares multiple machine learning models for fraud detection. These models can find complex relationships in the data and potentially outperform standard fraud detection systems, Logistic regression, neural network, random forest, and extreme gradient boosting (XGBoost) models are trained on a simulated dataset that closely follows properties of real card transactions. Performance of the models is measured by sensitivity, specificity, precision, AUC, and time to predict on the testing dataset. XGBoost shows the highest performance among the tested models. It is then compared to a standard fraud detection system used in a Czech bank. The bank system achieves higher specificity but XGBoost still shows promising performance. It is possible that certain machine learning models could outperform today's fraud detection systems if they are well-tuned.

JEL Classification	G21, K42
Keywords	machine learning, card fraud, fraud detection, imbalanced data
Title	Machine Learning Methods in Payment Card Fraud Detection

Abstrakt

Ochrana klientů před podvodnými transakcemi je náročný úkol. Banky se obvykle spoléhají na systémy založené na pravidlech, které vyžadují ruční tvorbu těchto pravidel pro identifikaci podvodu. Tato pravidla musí nastavit zaměstnanci banky, kteří musí sami vyhledávat trendy v podvodných transakcích. Tato práce se zabývá problémem odhalování podvodných karetních transakcí a porovnává několik modelů strojového učení pro detekci podvodů. Tyto modely mohou v datech najít složité vztahy a potenciálně překonat klasické systémy detekce podvodů, Logistická regrese, neuronová síť, random forest a extreme gradient boosting (XGBoost) jsou trénovány na simulovaném souboru dat, který věrně kopíruje vlastnosti skutečných karetních transakcí. Výkonnost modelů se měří podle citlivosti, specificity, preciznosti, AUC a časové náročnosti předpovědi na testovacím souboru dat. XGBoost vykazuje nejvyšší výkonnost mezi testovanými modely. Poté je porovnáván se standardním systémem detekce podvodů používaným v české bance. Bankovní systém dosahuje vyšší specificity, ale XGBoost přesto vykazuje slibné výsledky. Je možné, že některé modely strojového učení by mohly překonat současné systémy detekce podvodů, pokud budou dobře vyladěny.

Klasifikace JEL	G21, K42
Klíčová slova	strojové učení, karetní podvody, detekce podvodů, nevyvážená data
Název práce	Metody strojového učení v detekci podvodných karetních transakcí

Acknowledgments

I am very grateful to my doc. PhDr. Jozef Baruník, Ph.D.for his help, time and guidance. I would also like to offer my special thanks to my family for their undying support throughout my study.

Typeset in LaTeXusing the IES Thesis Template.

Bibliographic Record
Sinčák, Jan: *Machine Learning Methods in Payment Card Fraud Detection.* Master's thesis. Charles University, Faculty of Social Sciences, Institute of Economic Studies, Prague. 2023, pages 71. Advisor: doc. PhDr. Jozef Baruník, Ph.D.

Contents

List of Tables

List of Figures

Acronyms

AUC	Area Under ROC Curve
ATM	Automated Teller Machine
CNP	Card-not-Present
EBA	European Banking Authority
EEA	European Economic Area
FDS	Fraud Detection System
MCC	Merchant Category Code
PSD2	Payment Services Directive
POS	Point-of-Sale
ROC	Receiver Operating Characteristic
SMOTE	Synthetic Minority Over-sampling Technique
SCA	Strong Customer Authentication
XGBoost	Extreme Gradient Boosting

Master's Thesis Proposal

Author	Bc. Jan Sinčák
Supervisor	doc. PhDr. Jozef Baruník, Ph.D.
Proposed topic	Machine Learning Methods in Payment Card Fraud Detection

Motivation Nowadays, payment cards are one of the most commonly used means of making transactions. There are millions of card transactions made each month. But this popularity is also connected with a risk of fraud. According to The Nilson Report, a total of $28.65 billion was lost due to card fraud in 2019 across the world. Losses from card fraud were increasing in the past years and are expected to follow this trend in the future. The risk of fraud leads to the need for methods for card fraud prevention and detection. Banks use various methods to detect suspicious transactions to minimize losses borne by both their customers and themselves.

Traditionally used fraud detection systems are based on many rules which tend to be set empirically by bank employees. This may lead to possible inefficiencies which may be dealt with by using a certain machine learning approach (Abdallah et al. 2016). These methods can find more complex properties of fraudulent transactions and can also potentially lead to a better quality of the detection system.

This thesis aims to choose machine learning methods that can be used in the detection of fraudulent card transactions. Chosen machine learning methods will be compared to each other in performance on a large dataset of credit card transactions. The goal will be then to see whether these methods have sufficient performance and if they are suitable for real-life payment card fraud detection.

Hypotheses

Hypothesis #1: Machine learning methods are a viable alternative to conventional methods in a real-world application.

Hypothesis #2: Ensemble methods perform better in fraud detection.

Hypothesis #3: In fraud detection, Gradient boosting is superior to a Random Forest.

Methodology In fraud detection, financial institutions tend to use simple methods based on many empirical rules, which are based on the observation of patterns in transaction data. Machine learning methods may uncover the properties of payment card frauds more efficiently and easily.

We will make use of several machine learning approaches to identify fraudulent transactions among a sample of payment card transactions that come from an interval of approximately three years. The analysis will make use of multiple approaches. These would most likely be support vector machines, logistic regression, and a neural network. The analysis will contain a method based on gradient boosting since it has not been commonly used in this topic. We will also include another ensemble method such as a random forest.

Fraud data, in general, are highly imbalanced, since the proportion of frauds in a sample is very small. It is important to account for this imbalance as it may affect the quality of the models. It will most likely be dealt with by undersampling the data and thus manipulating the dataset in such a way, that frauds make up a larger proportion of data than in reality. Another approach would be the use of the Synthetic Minority Over-sampling Technique (SMOTE).

Individual models' quality will be ranked according to multiple criteria. These would most likely be sensitivity, precision, and Area under curve (AUC) or the F-score. The amount of time it takes to run the models on test data will also play a key role since there is a large number of transactions made each second in the real world and the model has to be able to keep up with this inflow.

Expected Contribution The goal and contribution of this thesis are to find a suitable machine learning method for payment card fraud detection. The advantages and disadvantages of individual approaches will be discussed, and the chosen method should provide the best performance in fraud detection and should be usable in a real-world application. One of the discussed approaches will be gradient boosting since it has not been used commonly in previous research. Its performance is not completely clear in comparison to other more common methods. While Taha & Malebary (2020) or Madkaikar et al. (2020) see it as the best method for card fraud detection, Gupta (2016) or Shakya (2018) claim that it is outperformed by a Random Forest. This thesis aims to answer the question of whether Gradient Boosting is the best method to use for this task.

Outline

1. Introduction
2. Literature Review
3. Overview of Suitable Machine Learning Methods
4. Data Description
5. Methodology
6. Results
7. Concluding remarks

Core bibliography

Abdallah, A., Maarof, M. A., & Zainal, A. (2016). Fraud detection system: A survey. Journal of Network and Computer Applications, 68, 90–113.

Bentéjac, C., Csörgő, A., & Martínez-Muñoz, G. (2020). A comparative analysis of gradient boosting algorithms. Artificial Intelligence Review, 54(3), 1937–1967

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16, 321–357.

Correa Bahnsen, A., Aouada, D., Stojanovic, A., & Ottersten, B. (2016). Feature engineering strategies for credit card fraud detection. Expert Systems with Applications, 51, 134–142.

Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C. & Bontempi, G. (2018). Credit Card Fraud Detection: A Realistic Modeling and a Novel Learning Strategy. IEEE Transactions on Neural Networks and Learning Systems, 29(8), 3784–3797.

Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2015). Credit card fraud detection and concept-drift adaptation with delayed supervised information. 2015 International Joint Conference on Neural Networks (IJCNN). Published.

Gupta, S. (2016). Deep Learning vs. traditional Machine Learning algorithms used in Credit Card Fraud Detection.

Madkaikar, K., Nagvekar, M., Parab, P., Raika, R., & Patil, S. (2021). Credit Card Fraud Detection System. International Journal of Recent Technology and Engineering (IJRTE), 10(2), 158–162.

Nami, S., & Shajari, M. (2018). Cost-sensitive payment card fraud detection based on dynamic random forest and k -nearest neighbors. Expert Systems with Applications, 110, 381–392.

Shakya, R. (2018). "Application of Machine Learning Techniques in Credit Card Fraud Detection, UNLV Theses, Dissertations, Professional Papers, and Capstones, 3454.

Taha, A. A., & Malebary, S. J. (2020). An Intelligent Approach to Credit Card Fraud Detection Using an Optimized Light Gradient Boosting Machine. IEEE Access, 8, 25579–25587.

Card Fraud Losses Reach $28.65 Billion. (2020, December 1). Nilson Report, 1187. https://nilsonreport.com/mention/1313/1link/

Chapter 1

Introduction

In banking, there are many risks that banks have to manage, because high risks may lead to increased costs. Like any other firm, banks are trying to minimize their costs to increase profits. Besides credit, market, or liquidity risk, there is also operational risk. Operational risk is the risk of losses due to failed internal processes, people, or external events. One of the external sources of operational risk is fraud.

Frauds are being committed with every product that banks offer and a large proportion of them belongs to payment card frauds. Even though frauds only make up a small share of all card transactions, there is still a very large number of them due to the massive amount of transactions overall. Banks are trying to minimize the number of frauds because they create costs as banks are legally obliged to compensate certain losses to clients. Moreover, a high fraud risk is connected to a high reputation risk, which leads to lower revenue. Therefore, banks are trying to identify and cancel fraudulent transactions.

Traditionally, banks use rule-based fraud detection systems. These systems are manually maintained and rely on rules created by employees of the bank. This process is time-consuming and may not be able to identify complex relationships in the data. But with the increasing popularity of artificial intelligence and statistical modeling, there is now a possibility to use various machine learning models to detect fraudulent payments.

This thesis builds and compares four machine learning models and analyses their ability to identify card frauds. The chosen models range from simple logistic regression to more complex ensemble algorithms. Multiple performance metrics are used to identify the best-performing model. This model is then compared to a Czech bank's rule-based fraud detection system. The main contribution of this thesis lies in this comparison. It tries to find out whether machine learning models can outperform a time-proven solution and whether these models are suitable for use in real-world fraud detection.

The thesis is structured as follows. In Chapter 2, we first describe the various fraud methods and recent trends in card fraud. This section also provides an overview of the ways how banks identify and prevent fraudulent transactions. Chapter 3 introduces the theoretical background of machine learning models used in this thesis and a general description of unsupervised and supervised learning. Chapter 4 deals with the dataset used in this analysis. It describes the source of the data, the process of data preparation, and the various obstacles we need to address when working with a dataset of fraudulent transactions. Further, distributions and descriptive statistics of all used variables are discussed. Besides this, Chapter 3 and Chapter 4 also go through the existing research on card fraud detection. In Chapter 5, we train all of our models and analyze their predictive abilities. We obtain various performance metrics and choose the best-behaving model. In Chapter 6, this model's performance is compared to the one of a fraud detection system used in a Czech bank. Finally, Chapter 7 summarizes our findings.

Chapter 2

Credit Card Fraud

2.1 Card Fraud Techniques

Credit and debit cards are some of the most common means of making transactions in the world. The ease with which they can be obtained and used is what makes them so popular. Global purchases of goods and services, cash advances, and withdrawals made with credit, debit, and prepaid cards reached more than 42 trillion USD in 2019 (The Nilson Report 2020), and are expected to increase to 56 trillion USD by 2025. Such a massive volume of transactions and the widespread use of payment cards encourage criminals to commit card fraud. Despite the low proportion of frauds among card transactions, total losses coming from fraudulent payments are still very high.

In 2019, gross fraud losses to issuers, merchants, and acquirers of transactions reached 28.65 billion USD worldwide (The Nilson Report 2020). Every year, these losses have been rising together with the total card transaction amount and are expected to keep rising in the future. In 14 selected EU countries, according to a study of the European Banking Authority (EBA), approximately 0.016% of card transactions were reported as fraudulent by issuers in the second half of 2020 (EBA 2022). In the case of transaction value, fraudulent payments make up 0.025% of the total value. Card fraud rates reported by acquirers are even higher. About 0.035% of transactions were identified as fraudulent, which translates to 0.046% of the total value. According to the EBA, the total card fraud value reported by issuers and acquirers in H2 2020 was more than EUR 440 million.

Credit card frauds can be split into application and behavioral frauds (Bolton & Hand 2001). Application fraud happens when individuals obtain credit cards based on applications with false personal or financial information. Such fraudsters may try to spend as much as possible shortly after obtaining their card and then refuse to repay the debt. This type of fraud is quite common, but it is dominated by behavioral frauds. Behavioral frauds make up a large group that can be divided into four main categories. These are Card-not-Present (CNP) fraud, counterfeit card fraud, lost/stolen card fraud, and card-never-arrived fraud.

The most common type of behavioral fraud is card-not-present fraud (The Nilson Report 2020). A card-not-present transaction is a transaction where the physical card is not used. These are usually online purchases. The card is present when it is inserted into the merchant's terminal or when its chip is contactless scanned at the Point-of-Sale (POS). Fraudsters most often obtain card details through a phishing attack. They can then misuse it in several ways. In the most simple one, they use the card in online shopping and buy goods for themselves. It is also possible to use the card to buy gift cards for online services such as Google Play Store, Amazon, or Apple Appstore. Fraudsters then offer these gift cards in grey markets and profit from sales. One of the sophisticated methods makes use of fake online shops. Fraudsters set up a shop and use the card to buy virtual goods with it. This way money from the card is transferred directly to the fraudsters' pockets.

CNP transactions are gaining popularity as customers are using their physical cards less and instead are shifting their shopping habits to the Internet. Card-not-present purchase volume corresponds to 15.4% of all purchase volume in 2019, but in fraud losses, the share of CNP transactions is 65% (The Nilson Report 2020). This makes CNP frauds the main priority when fighting fraudulent transactions.

Credit card information can be stolen via malware attacks that traditionally target computers, but nowadays malicious code can also be included in smartphone apps. Users, therefore, need to be cautious and carefully choose the applications which they are installing. Social engineering is the other and possibly more dangerous method of obtaining card information. Fraudsters keep making more and more complex methods by which they persuade card owners to hand over all key information about their cards. The exact approaches differ but fraudsters usually act as trustworthy and legit institutions and try to manipulate their targets. They can use fake e-mails, sms messages, or even phone calls, in which fraudsters talk to their victims in order to receive credit card information (The Nilson Report 2020). Such attacks are known as phishing, smishing, and vishing attacks. Their biggest danger lies in their credibility. At first glance, e-mails and messages seem legit and attackers act kindly and seem to be helping their targets during phone calls. It is too late when victims of these attacks find out that somebody made unauthorized transactions with their cards online and that their bank accounts are missing money. The popularity of these attacks among fraudsters comes from their profitability. All that is needed is a fake website that looks like its real counterpart. Then with the help of a stolen database of e-mail addresses or phone numbers, this website can be easily distributed to thousands of potential victims. Even though the click rate may be low and most people recognize these attempts and delete such messages, it only takes a couple of individuals who are not as careful. A successful attack typically earns the fraudster hundreds or thousands of euros. In the case of companies, the lost amounts can be much higher. This means that the effort-to-profit ratio is usually very high. And those who cannot build such fraudulent systems themselves have the opportunity to buy complete fraud solutions on a darknet market.

Another significant amount of fraud losses comes from lost and stolen cards. This type of fraud is self-explanatory. When a card owner loses their card or it is stolen, there is a certain time period before the owner finds this out and asks their bank to block the card. During this period, the lost card can be misused if found by a fraudster. This can then lead to a substantial loss. A similar type of fraud is card-never-arrived fraud. When a card owner orders a new card, this card is typically sent to them via mail. But during this process, the card can be stolen by a post office employee in transport or by somebody else when it is delivered to a mailbox.

The final major card fraud type is skimming and counterfeit fraud. Those occur when card details are illegally taken and used to create counterfeit cards. Skimming is a technique used to scan card information from its magnetic stripe with a device called a skimmer. Skimmers can be installed in payment terminals but most often they are used in an Automated Teller Machine (ATM). They tend to be installed in place of a legit card reader in an ATM and most of the time they are nearly identical to real readers. Therefore for a common consumer, it is very difficult to tell if a skimmer is used in an ATM. On the other hand, since skimmers need to be this complex, they are also expensive and hard to obtain. Banks are also monitoring their ATM with cameras and thus it is difficult to install a skimmer without getting noticed. These factors combined mean that skimming and counterfeit card frauds are disappearing.

Instead, fraudsters are shifting their interest to the internet and are starting to use various phishing methods. Banks often stay one step behind and are not able to prevent such attacks.

Besides attempts to scam cardholders, there are also card frauds whose objective is to get money from a bank. Fake chargebacks are one of the methods by which acquiring banks are being targeted. Such frauds can be made by individuals who pay for goods and services online with the intention to contradict the transactions and demand compensation from the bank. A special type of chargeback fraud is family fraud. Most often, the client disputes transactions that have been made by their children. A child can make transactions in games on mobile phones on which the parent has their card information stored. Once the parent finds out, they may try to apply for a chargeback (The Nilson Report 2020). In these cases, it can be quite challenging for the bank to prove that the transaction has been made by the cardholder and therefore there is a risk of losses, although usually not large. There are also organized groups that are capable of creating complicated transaction schemes which are difficult to detect and which can lead to large losses.

2.2 Card Fraud Detection

A large majority of losses from card fraud is borne by owners of the cards. Banks are only liable for a small proportion of losses. They may need to compensate a client when bank employees do not manage to block a lost card properly or in some cases when the law tells them to do so. These losses may not be a large enough incentive for banks to detect and prevent frauds. But they face a risk of a damaged reputation if they do not prevent frauds and clients start losing large amounts of money. This is why banks and card associations are actively trying to detect and prevent fraudulent transactions. These measures apply not only to card transactions but also to money transfers from one account to another and other types of transactions. For this task banks use various systems which monitor all transactions made by customers and look for potentially fraudulent transactions. These systems can either be rule-based or use some kind of machine learning method for fraud detection.

Traditional fraud detection systems are based on a predetermined set of rules (Ghosh & Reilly 1994). These are set manually by employees and come from rough observations of the data. Once a transaction turns out to be fraudulent, its properties are analyzed. With a large sample of fraudulent transactions, it is possible to extract properties that they have in common. Bank employees can then turn them into a set of rules against which all transactions are checked. Such detection systems tend to be quite simple and are not able to capture complex patterns in fraudulent transactions. Moreover, the set of rules is not being recalculated continually. Fraudsters are innovative and their methods are constantly evolving. A low frequency of rule recalculation means that the detection system may miss new trends in both fraudulent and genuine transactions and thus become unreliable (Pozzolo *et al.* 2015). This problem is called concept drift and it may lead to larger losses of customers and a worse reputation for the bank.

A modern substitute lies in detection models based on machine learning algorithms. With increasing and more affordable computing power and with fast development in machine learning, these methods are becoming more popular among banks. They perform better at the disclosure of complex relationships between a large number of variables. Systems based on machine learning algorithms work by detecting behavioral patterns in transaction data and allow for more complex analysis. Therefore these methods can be more successful and can prevent more losses. They can also learn on the go and immediately implement new fraud trends into their calculations. This makes machine learning systems more robust and helps them keep up with innovations in fraud techniques. Nowadays, a wide range of different machine learning methods is available, which gives banks a lot of freedom in their choice.

Overall, there is a very small proportion of frauds among all card transactions. The task of a Fraud Detection System (FDS) is therefore to identify a very small group of observations that differ from the majority of genuine transactions. This is therefore an application of an anomaly detection problem. Anomaly detection in general is „the problem of finding patterns in data that do not conform to expected behavior“ (Chandola *et al.* 2009). There are various applications of this problem, such as intrusion detection in computer systems and networks, analysis of health records of patients, sensor networks, and also bank frauds. In the field of financial fraud, some of the most popular machine learning techniques are neural networks, rule-based decision trees, or clustering (Chandola *et al.* 2009). It is also possible to use systems, which create a statistical representation of normal account usage and then look for deviations in the behavior of a customer (Edge & Sampaio 2009).

2.3 Fraud Prevention

Instead of trying to detect a fraudulent transaction, it may be more efficient to completely prevent it from happening. This is what Fraud prevention systems are trying to accomplish. These systems are in the first layer of protection and their purpose is to stop frauds from occurring at all (Abdallah *et al.* 2016). Among these mechanisms could be systems that encipher and decipher sensitive data, or various firewalls and other forms of data security.

An efficient tool to combat fraudulent transactions has been brought by the Payment Services Directive (PSD2). This EU Directive, amongst other things, requires the use of Strong Customer Authentication (SCA) for a majority of electronic payments. The requirement has had the greatest effect on standard transactions between two accounts but it has also targeted remote card transactions. The need to authenticate every card-not-present transaction by at least two elements has led to a great decline in fraud rates. When SCA is not required for remote transactions, the fraud proportion is five times higher in total volume as reported by both issuers and acquirers and three to four times higher in total value compared to payments authenticated with SCA (EBA 2022). This correlation is also noticeable with non-remote transactions and with cross-border transactions. Payments with counterparts outside of the European Economic Area (EEA) are not subject to the PSD2 regulation and therefore the fraud rates are much higher compared to domestic and within-EEA transactions. One of the most common implementations of strong customer authentication is a 3D-Secure protocol.

If the fraudulent transaction remains unprevented and undetected by the bank and the customer raises a complaint, there is still a chance to reverse the transaction through a chargeback. Chargebacks can be costly since they tend not to be regulated by law and to be successful, the target account still has to have sufficient funds. Fraudsters are aware of this and thus they often make use of straw men and create complex structures of multiple accounts through which the money is transferred. A vast majority of fraudulent card transaction volume comes from cross-border transactions, often from payments with counterparts outside of the European Economic Area (EBA 2022). This makes it harder for domestic banks to make a successful chargeback since the target banks may have poor fraud prevention and anti-money laundering tools.

Chapter 3

Machine Learning in Card Fraud Detection

We live in the era of big data. Human behavior and natural processes lead to the creation of extremely large and complex datasets. Thanks to the progress in information technology, we can store such data and because of human curiosity and innovativeness, we are trying to analyze, understand and use this data in order to create some value. With such massive and unorganized piles of information, it is almost unthinkable to attempt a manual analysis of this data. Programs written exactly for a given task would be very difficult to create, which leads to the need for a different approach to data analysis. A solution to this request lies in automated methods of data analysis known as machine learning algorithms.

We can define machine learning as "A set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty" (Murphy 2012). Machine learning methods give us generalized procedures which we can employ to obtain a solution to our problem given a large dataset. Another, more technical, definition of machine learning is provided by (Mitchell 1997): "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E". In our case, the task T lies in detection of fraudulent card transactions and E is the training data we provide to the algorithm. This task is controlled by P, which makes sure that the algorithm operates with high performance. In our case, we want the algorithm to flag as many fraudulent transactions as possible, but we also do not want it to produce too many false alarms.

Machine learning approaches can be divided into two main categories. These are supervised and unsupervised learning. Even though supervised and unsupervised learning are not formally defined terms (Goodfellow *et al.* 2016), there are certain features that distinguish these two approaches. Each approach has its pros and cons and both of them have been used in the past in the identification of fraudulent transactions (Ngai *et al.* 2011).

3.1 Unsupervised Learning

Unsupervised learning is a type of machine learning in which the algorithm is trained on data that is not labeled or pre-classified. The machine learning algorithm is trying to find patterns and relationships in the data on its own, without being provided with any specific guidance or feedback (Goodfellow *et al.* 2016).

The primary goal of unsupervised learning is to discover hidden structures, groups, or clusters in the data. The algorithm seeks to identify natural patterns and relationships in the data, and then use those patterns to make predictions or gain insights.

There are different use cases for unsupervised learning algorithms. They are widely used in image and speech recognition, recommendation systems, and anomaly detection. Among some of the most commonly used unsupervised learning algorithms are clustering algorithms, dimensionality reduction algorithms, and anomaly detection algorithms (Hastie *et al.* 2009).

Clustering algorithms group similar data points together into clusters, which can be used for customer segmentation, social network analysis, and many other applications. Dimensionality reduction algorithms reduce the number of features or variables in the data, which can be useful for visualization and compression of data, as well as for identifying important features for further analysis (Lee & Seung 1999). Anomaly detection algorithms identify unusual data points that are significantly different from the rest of the data and can be used in fraud detection, network intrusion detection, and other applications.

Another possible unsupervised method is Peer group analysis (Bolton & Hand 2001). This approach notices when an object starts to behave differently from objects which behaved similarly before. In the context of card fraud detection, this method may analyze behavior of individual clients and group them by their spending patterns. Once a client starts making transactions that diverge from the norm of the group, then the transactions are labeled as potentially fraudulent.

Unsupervised learning algorithms have several advantages over supervised learning algorithms. The data does not require any labeling, which means that obtaining data is easier. They can also be used to discover new patterns and relationships that may not have been apparent before, which can lead to new insights and discoveries.

However, unsupervised learning also has some limitations. Since there is no predefined output, the algorithms can sometimes produce meaningless or incorrect results. It can also be difficult to evaluate the performance of unsupervised learning algorithms since there is no predefined correct output to compare the results against. The required computations tend to be more demanding and time-consuming. With an increasing number of features, the complexity of the model rises substantially. Unsupervised algorithms tend to be used less in finance than supervised learning. This also applies to the topic of card fraud detection, even though it can still be utilized (Rai & Dwivedi 2020).

Unsupervised learning is a powerful tool for discovering hidden patterns and relationships in data. It is an essential component of many machine learning applications and has many practical applications in industry and research. While unsupervised learning algorithms have some limitations, they are a valuable tool for gaining insights and making discoveries in large and complex data sets in which supervised algorithms may struggle.

3.2 Supervised Learning

A more suitable approach to this topic is supervised learning. Unlike unsupervised methods, these algorithms require a label that assigns each observation the desired output. Supervised learning is used in two main tasks, called classification and regression (Murphy 2012).

In a classification problem, we want to map inputs x to an output y, where y is a discrete variable that represents several classes. Based on this number, we can distinguish a binary classification with only two possible classes, and a multiclass classification, where the number of classes is larger than two. Since we are deciding between two classes (fraudulent and legit transaction), we are dealing with a binary classification problem. We are trying to find a function f which gives us the probability distribution of possible classes, given x. In the case of only two classes, we only need to return a single number $p(y = 1|x, D)$, where D is the training set (Murphy 2012). The estimation of the true label y is then computed as $\hat{y} = \hat{f}(x) = p(y = 1|x, D)$. There is a wide selection of methods used for classification, such as decision trees, random forests, support vector machines, or neural networks.

In the case when y is a continuous variable, we are facing a regression problem. This is used when we need to predict a numerical value given input data. We are looking for a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$. Regression analysis is one of the most used tools in financial modeling. There are many regression models, and this thesis will utilize one of them, a logistic regression model. Even though this model is based on the regression framework, it is used in classification problems.

Advantages of supervised learning lie in the possibility of manually choosing the boundaries of classes. This allows us to be specific about the classes we want to have in our data. Supervised learning also tends to be less computationally demanding and therefore requires less processing time. On the other hand, it may underperform in very complex tasks when compared to unsupervised learning. It is also unable to identify labels in test data that are not present in the training data.

3.2.1 Logistic Regression

Despite being a regression model, logistic regression is used to estimate a discrete dependent variable. It does so by modeling a probability of a given class. This class tends to be binary, but the model can be extended to account for multiple classes. Logistic regression is based on linear regression, which is one of the most widely used regression models.

Linear regression is a method for modeling the relationship between a vector of explanatory variables and one explained variable. It is one of the simplest and most commonly used models. As the name suggests, this relationship is estimated using a certain linear function. This can be written as follows: $y(x) = w^T + \epsilon = \sum_{j=1}^{D} w_j x_j + \epsilon$ where $w^T x$ marks a scalar product between an input vector x and a weight vector w, which adjusts the influence of individual objects from x on y. ϵ then presents a residual error between the true value of an independent variable and our estimation (Murphy 2012). The residual error is assumed to follow a normal distribution. This is denoted by $\epsilon \sim N(\mu, \sigma^2)$. The probability distribution can then be written as $p(y|x, \theta) = N(y|\mu(x), \sigma^2(x))$ where θ is the parameter vector we are looking for. Linear regressions are often estimated using the method of ordinary least squares (OLS), which is a special case of maximum likelihood estimation (MLE).

Logistic regression can be derived from a generalized linear regression into a binary classification model. Therefore, the dependent variable will only take on two levels. This property comes from several factors. Since y will be a binary variable ($y \in \{0, 1\}$), we assume that it follows a Bernoulli distribution. Moreover, the inputs have to be transferred by a special function that will make sure that the output stays within the range from 0 to 1. This function is called a sigmoid function. Other names for this function are logistic or logit and these give a name to the whole regression algorithm. The sigmoid function is defined as $sigm(x) = \frac{1}{1+exp(-x)} = \frac{exp(x)}{exp(x)+1}$. This gives us the model for estimating probability of $y = 1$ given a vector x as $Prob(y = 1|x) = \frac{exp(x'w)}{1+exp(x'w)}$. This can be estimated using a maximum likelihood estimator.

Logistic regression is used for detection of card frauds as a simple model together with more complex models by Bahnsen *et al.* (2016). Their paper uses a normal logistic regression as well as its cost-sensitive variant. When using savings as a performance measure, the standard logistic regression falls behind, but the cost-sensitive version achieves some of the highest savings. This model manages to beat cost-sensitive decision trees, a Bayes minimum risk model, and a random forest.

3.2.2 Random Forest

Another option for a classification model is random forests, which are an extension of a decision trees method. Decision trees can be used for both regression and classification tasks and as the name suggests, they revolve around the construction of a tree with multiple decision points. Their goal is to predict the value of a dependent variable by learning simple decision rules for some of the independent variables. This makes them very simple to interpret and understand. They can also be nicely visualized. Unlike some other common learning methods, decision trees are "white box" models. This means that their behavior and decision-making can be easily explained. On the other hand, the model can grow into a very complex decision tree, which is prone to overfitting the data. The model then performs poorly on test data. Decision trees can also suffer from instability, where a small change in input data can lead to a completely different model (Murphy 2012). These drawbacks can be mitigated by creating a random forest of multiple decision trees.

Random forests build on the framework of decision trees and improve it by relying on the performance of a large number of individual decision trees (Ho 1995). They can be defined as a “combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest” (Breiman 2001). Random forests implement a technique called bootstrap aggregating, or bagging. This means that the model draws a random subsample with replacement from the training dataset. This dataset has the same size as the original one. Further, the bagging process is expanded and the algorithm chooses a random sample of independent variables and constructs a decision tree using the subsample of training data and chosen variables. Multiple such trees are built and then they are used to form a combined model.

In a regression task, the mean of their predictions of the value of output is used. When used for classification, the class selected by a majority of trees is chosen. Since the individual trees are trained on different data and use different variables, they are uncorrelated with each other, and therefore the final random forest does not suffer from instability like a decision tree. Besides that, random forests do not suffer from overfitting since the number of used trees greatly reduces this risk. They are a good choice of model for classification tasks because they offer good performance. But because of the random selection of observations and variables, random forests are no longer white box models. Hence, they are not as easily interpretable as decision trees.

In their paper, Nami & Shajari (2018) use a novel approach in which they combine a k-nearest neighbors model with a variant of the random forest model, called dynamic random forest. These random forests require fewer trees and time to predict. This paper shows that the combination of these models achieves a high sensitivity while controlling very well for specificity when used on a dataset of 54 thousand card transactions.

In a paper by Madkaikar *et al.* (2021), multiple machine learning models are used to identify fraudulent transactions on a dataset of 50 thousand observations. Besides random forest, authors also use gradient boosting, support vector machines, or logistic regression. Using accuracy as a performance measure, gradient boosting and random forest show the highest performance, closely followed by support vector machines.

3.2.3 Artificial Neural Networks

Artificial neural networks, or simply neural networks, are a group of algorithms inspired by animal and human brains. Neural networks are nowadays a widely used algorithm and their basic idea has been first introduced by McCulloch & Pitts (1943). There are multiple models of this type but the most successful one in pattern recognition is the feed-forward neural network, also called a multilayer perceptron (Bishop 2006). Neural networks are based on multiple connected units, called artificial neurons. These are supposed to work similarly to neurons in a real brain. A neuron can receive a signal, process it, and send it to another neuron for further calculations. The output of a neuron is a nonlinear function of its inputs. Several layers of neurons are connected by nodes. Each node and neuron have a certain weight, which manipulates the influence of one neuron on the next one.

In a neural network, input variables $x_1, \ldots, x_D$ are first transformed using M linear combinations into activations

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}$$

where (1) marks the first layer of the neural network and $j = 1, \ldots, M$. Parameters $w_{ji}^{(1)}$ represent individual weights and parameter $w_{j0}^{(1)}$ marks bias. The combinations a_j are then transformed using a linear or nonlinear activation function $h(.)$ into $z_j = h(a_j)$.

These elements are called hidden units. There can be several hidden layers in a neural network. Finally, using another linear combination of the hidden units, we receive the output unit activations

$$a_k = \sum_{i=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)}$$

In the case of classification, these are transformed into final outputs y_k using a logistic sigmoid function. In the topic of card fraud detection, there is only one output variable that states whether a transaction is fraudulent or not.

Neural networks vary in complexity based on the number of hidden layers they use. In the simplest case, there is only one hidden layer. Networks with a large number of hidden layers are one of the many types of deep learning algorithms. Neural networks have a broad specter of use from face identification, through text recognition to algorithms that recommend content on social networks. Their popularity lies in their ability to reliably learn complex relationships and predict on test data. There are also no restrictions on the structure of input data. On the other hand, neural networks work as a black box due to the computations which are made in multiple layers, and thus they are very complex algorithms. This is also connected with a high demand for computational power.

There are multiple variants of neural networks, that can be used in fraud detection (Zakaryazad & Duman 2016). Their paper builds several neural networks and looks at their accuracy in fraud detection on two datasets of credit card transactions obtained from two Turkish banks. These neural networks are also compared with a decision tree and naïve Bayes. Moreover, the authors focus on the total savings provided by the neural network as a performance metric. The results of this paper show that an ordinary neural network achieves the highest accuracy and true positive rate on both datasets. The extra complexity of other neural networks did not bring any significant performance improvement.

3.2.4 Gradient Boosting

Boosting is one of the most frequently used machine learning approaches. It provides high performance, and the process is understandable despite being a black-box model. Boosting relies on a so-called weak learning algorithm, which uses training data to create a weak classifier (Schapire & Freund 2012). These weak classifiers by themselves can be quite simple and provide low accuracy, as long as they perform better than pure guessing. The algorithm creates several such weak classifiers, where each one is based on a different subsample of the training dataset. Further steps of the procedure differ across individual boosting algorithms.

AdaBoost is one example of a boosting algorithm. It works in rounds when creating weak classifiers and uses an exponential loss function. In each round, it draws a subsample from the training set based on a distribution it creates over the examples. Each example is given a weight that gradually increases for the examples which are classified incorrectly. This makes the algorithm work harder on examples that have previously been classified incorrectly. After a given number of rounds, the algorithm combines the obtained classifiers into a single final one based on weights given to the classifiers (Schapire & Freund 2012). A large benefit of AdaBoost is that it takes it long to start overfitting (Murphy 2012).

Adaboost has been used in comparison to a real-world fraud detection system. Chan *et al.* (1999) use algorithm AdaCost, a variant of AdaBoost, and compare its performance to a fraud detection system of Chase Bank applied on real transactional data of Chase Bank and First Union. The AdaCost algorithm looks at the cost of misclassification of a fraudulent transaction and implements it into the optimization process. Moreover, the authors are trying to maximize savings produced by the predictions of the algorithm. Their analysis shows that the machine learning classifier achieved higher savings than the system used by the bank. Further, they also create so-called AdaCost metaclassifiers which combine multiple classifiers when deciding whether a transaction is fraudulent or not. This metaclassifier achieves even better results than the base classifier.

Gradient Boosting presents a rather new boosting algorithm, which offers high performance. Unlike other boosting methods, gradient boosting is unique because it provides a generic framework that can be used for different loss functions (Friedman 2001), (Mason *et al.* 2000). The name comes from the gradient descent process that this algorithm employs to minimize the loss function. This process minimizes the loss function in subsequent steps where the following estimation is based on a modified version of the previous dataset. After the first estimation on the original dataset, the algorithm finds out which errors between the true target value and prediction were highest and the model then focuses on those further. Once the errors are sufficiently low, the model takes all of the obtained predictions and combines them into the final result.

An extension of this approach are gradient boosted decision trees Friedman (2001). In this method, each weak learner takes on the form of a decision tree and is used in combination with gradient boosting. In each boosting step, the algorithm fits a decision tree and then combines the obtained trees in its final evaluation. Gradient boosted decision trees are a very popular gradient boosting method in both regression and classification tasks thanks to their accuracy and efficiency.

Another interesting implementation of gradient boosting is the so-called Extreme Gradient Boosting (XGBoost). Even though it is a very new algorithm (Chen & Guestrin 2016), it is quickly becoming very popular thanks to its many advantages over other gradient boosting models. XGBoost is a tree-based method just like the majority of other gradient boosting algorithms. Unlike other models, XGBoost makes use of a regularization term when it is training. This term limits the complexity of the algorithm which helps it to avoid overfitting (Chen & Guestrin 2016). It is added to a standard loss function and the algorithm tries to minimize both of them together. Furthermore, XGBoost provides the ability of parallel tree learning, which makes it more powerful and also speeds up the learning process. This algorithm also offers great flexibility in the type of data it can use. XGBoost is capable of handling many different types of data, including missing data and it supports a wide range of objective functions, evaluation metrics, and optimization objectives (Chen & Guestrin 2016). Finally, it provides a score of importance of each variable, which helps it identify the most important variables to make accurate predictions. Overall, it is a very powerful and flexible algorithm that is usable in both classification and regression tasks and offers great performance in many different applications (Bentéjac *et al.* 2021).

Gradient boosting is also a powerful algorithm for fraud detection. Randhawa *et al.* (2018) compare multiple machine learning models in their ability to identify frauds in two real transaction datasets from Europe and Malaysia. They create 12 different machine learning models and look at accuracy, sensitivity, specificity, and the Matthews correlation coefficient on both of the used datasets. Gradient boosting achieves some of the highest scores in each of these performance metrics. Further, the authors set up a majority voting system, where pairs of different models vote to produce the final verdict. In this scenario, a combination of a gradient boosting model with decision trees beats all other algorithms. Moreover, the dataset is then filled with up to 30% of noise data. Even on this data, the combination of gradient boosting and decision trees manages to lose very little performance.

A different implementation of gradient boosting, Light gradient boosting can also serve as a good algorithm for card fraud detection. In their paper, Taha & Malebary (2020) build an Optimized light gradient boosting model (OLGBM) and use it on two datasets of fraudulent transactions. They then compare it to models used by different authors on the same datasets. Some of those models are random forest, support vector machine, or logistic regression. The OLGBM algorithm manages to outperform all the other models in terms of the F1 score, precision, and accuracy on both datasets. Its sensitivity is somewhat lower, but still competitive with the rest of the models.

Chapter 4

Data

4.1 Data Source

Obtaining a sufficient dataset of fraudulent credit card transactions is a demanding task. Machine learning algorithms require a large number of observations. This is connected with the need for independent variables which can provide a good explanation of changes in the dependent variable. Even though banks and credit card associations own huge datasets on card payments and frauds, they tend not to provide them to the public. Such data contain a lot of private information which cannot be legally shared without considerable adjustments. Financial institutions cannot risk damaging the public opinion of themselves and therefore keep their data confidential.

Even if there is an opportunity to obtain real-life data from a bank or other financial institution, these data may often be incomplete. Companies are either unable to keep their data sufficiently clean and simple or they do not gather some important data at all. Data incompleteness is one of the biggest problems of real datasets. If the dataset is large enough and there is not much missing data, this problem can be mitigated quite easily. But in the case of too many missing data points, the dataset can be interesting because it consists of real-life data, yet unusable for analysis.

An alternative to getting a dataset directly from a bank is the use of data obtained and shared for the purposes of research. When it comes to credit card fraud, there are not many publicly available datasets, and the public ones are quite limited in the number of observations. This thesis, therefore, builds its analysis on a dataset that has been artificially generated for research purposes.

4.2 Dataset

This thesis uses a publicly available dataset that has been synthesized by researchers from the IBM T.J. Watson Research Center. Properties of this data correspond to those of human-made transactions (Altman 2019). Authors create models of thousands of individuals whose characteristics are representative of the United States. This synthetic sample of individuals matches closely the means and standard deviations of various characteristics of the real US population. Examples of these are age, income, geographical distribution, or credit card spending. Correlations between these variables are also taken into account and simulated.

Next, credit card transactions are synthesized. In the same way as consumers, merchants are also modeled so that their properties match their real counterparts. They are spread around the world since US citizens may use foreign shops. Sales amounts differ across merchant categories and transactions also follow trends over the course of a day. Overall this card usage data makes for a good representation of real consumer behavior.

Finally, fraudulent transactions are simulated. Fraudsters are generated while keeping some of their special features in mind. In the same way, as in reality, they only commit frauds for a limited time period. Moreover, a person can become a fraudster and they can also decide to stop their illegal activities. Fraudulent transactions are randomized across time for each consumer. This feature of synthetic data can be very beneficial since it simulates the worst-case scenario when fraudsters find a way to make their transactions completely random, not following any patterns. If a fraud detection algorithm can identify random frauds well, then it will likely do even better in a real-world situation where certain transaction patterns are present.

The dataset consists of transactions made over a long period and transactions reflect changes in technology and consumer behavior. The proportion of online payments is gradually increasing and so is the share of online frauds. Consumers also get older and thus some retire and stop spending as much as they used to. Others are becoming adults and start using their cards more and more. Overall, the dataset provides a very good simulation of general population characteristics, transaction behavior, technological changes, and fraudsters' activity across a long time period. Therefore it is a valid alternative to real data. Much less data manipulation is needed and the randomness of frauds provides a bigger challenge for the fraud detecting algorithms.

4.3 Data Preparation

The full dataset comprises three files with a total of 48 variables. The files are divided into information on individual consumers, their cards, and their transactions. These files are linked together and thus all variables can potentially be used in the analysis. The dataset is massive, as it consists of more than 20 million transactions made between 1991 and 2020. There are 2000 unique consumers and more than 6000 cards used. Due to the large number of observations, a sufficiently sized subsample will be drawn from the data. Moreover, only data on transactions made in recent years are used since card technology, transaction trends, and methods of fraud are evolving in time. Data on older transactions would thus carry much less information.

We choose a sample of original data where we only choose transactions made between the years 2018 and 2020. The amount of data from 2020 is limited as the latest transactions come from February of 2020 and there are no fraudulent transactions recorded in this year. There were approximately 3.6 million transactions made during this period but only 4450 of them were fraudulent.

Since the variable which distinguishes fraudulent and genuine transactions is highly imbalanced, we need to manipulate the data to make it more balanced. The proportion of frauds only makes up approximately 0.1% of all transactions. Such a small proportion of frauds would make it very difficult for the algorithms to correctly distinguish fraudulent transactions and therefore lead to worse performance (Gupta 2016). Therefore first, the dataset is undersampled. This procedure randomly removes legitimate transactions and only leaves us with 120 thousand of them. After that, we oversample using SMOTE in order to increase the proportion of fraudulent transactions in our sample. This combination of undersampling and SMOTE is an ideal solution to a large imbalanced dataset (Haixiang *et al.* 2017).

Synthetic Minority Over-sampling Technique (SMOTE) is a method used in machine learning to address the problem of imbalanced datasets. It works by creating new synthetic data points for the minority class by interpolating between existing minority class data points (Chawla *et al.* 2002). Specifically, for each minority class data point, SMOTE selects one or more of its nearest neighbors (other minority class data points) and creates new synthetic data points along the line segments between the original data point and its selected neighbors. The algorithm allows users to control the degree of oversampling by specifying the number of synthetic samples to create. SMOTE can be used with a variety of machine learning algorithms, and it is particularly useful for training classifiers that need to achieve good performance on both the minority and majority classes. SMOTE can have many applications in fields, where highly imbalanced data are present. It is very useful in the classification of DNA proteins, computer vision, text classification, and also fraud detection (Kaur *et al.* 2020), (Mqadi *et al.* 2021).

We oversample frauds and partially legitimate transactions as well and end up with 164 650 transactions. 31 150 of those are fraudulent, which translates to a proportion of 18.9% of frauds in our dataset. This will allow the machine learning algorithms to perform much better.

There is a large number of variables in our dataset and only a part of them is used by our algorithms. We choose 10 variables that we will use in our models. 5 of those are categorical variables, 2 are numeric and the remaining three are binary variables. Since some of the models we are using cannot easily work with categorical variables, we need to modify these variables into numerics. There are three possible approaches to how we can do this. We can use learned embedding of categorical variables, integer encoding of variables, or one hot encoding.

Learned embedding transforms categorical variables into one or more vectors of continuous numerical variables. It can transform complex relationships between categories and turn them into low-dimensional variables. They can then be used in machine learning models as standard explanatory variables. Learned embedding is very often used to represent words and their relative meanings in text analyses. But it is a very flexible method of transformation of categorical variables and therefore it is widely used in other fields as well.

An alternative to learned embedding is called integer or ordinal encoding. This method assigns an integer to each category of the original variable. The variable can then be used to train a machine learning model. This is not as complex as learned embedding since the numerical values are not calculated. But it requires the original data to be ordinal. We can only use this method if we can order the single categories of a given variable. This is not the case with our data and therefore we cannot use this approach.

Finally, there is one hot encoding. This method checks the number of categories in a variable and then creates a single new column of data for each category. Each column is binary and contains a 0 or 1 depending on the category in the original variable. One hot encoding is very easy to implement as it only splits categorical data into multiple binary columns. Machine learning models are also able to process binary variables easily. But if there are variables with many categories, they will be transformed into a large number of columns. It can be difficult to understand such a wide dataset. Moreover, machine learning models may become slow with a large number of columns. We use this method on categorical variables which have only a few different categories.

4.4 Data Description

The variable we are most interested in is *fraud*. This is a binary variable that denotes whether a transaction was fraudulent or genuine and it is our dependent variable. In our original dataset, frauds only make up 0.12% of all transactions, as shown in Table 4.1.

Table 4.1: Proportion of Frauds in Original Dataset

Fraud	Proportion
No	99.88%
Yes	0.12%

Therefore we undersample and oversample using SMOTE in order to increase the proportion of fraudulent transactions in our dataset. This way we increase the proportion of frauds to 18.92% as shown in Table 4.2.

Table 4.2: Proportion of Frauds in Resampled Dataset

Fraud	Proportion
No	81.08%
Yes	18.92%

Probably the most important numeric variable in our dataset is *amount*. This variable describes the amount in US dollars of each transaction. In our analysis, we use a logarithmic transformation of this variable since the distribution of the original variable would be very skewed. Descriptive statistics of the variable are presented in Table 4.3.

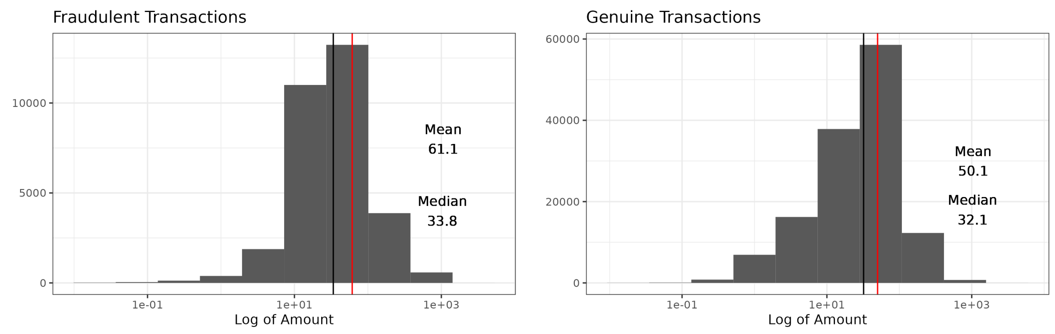
We can look at the difference in transaction amounts between fraudulent and genuine transactions. This comparison is presented in Figure 4.1. We

Table 4.3: Properties of log(*amount*)

Descriptive Statistics					
Statistic	N	Mean	St. Dev.	Min	Max
amount	164 650	52.19	75.56	0.01	1 710.02

can see that amounts of both fraudulent and genuine transactions follow a normal distribution. We can also see that fraudulent transactions have a higher average amount than legitimate transactions. This can be caused by the fact that fraudsters can try to make as high transactions as possible once they are in control of a card. On the other hand, the median amount is similar in both groups. This means that the higher average amount is likely caused by higher transaction amounts in the upper tail of the distribution.

Figure 4.1: Distribution of log(*amount*)



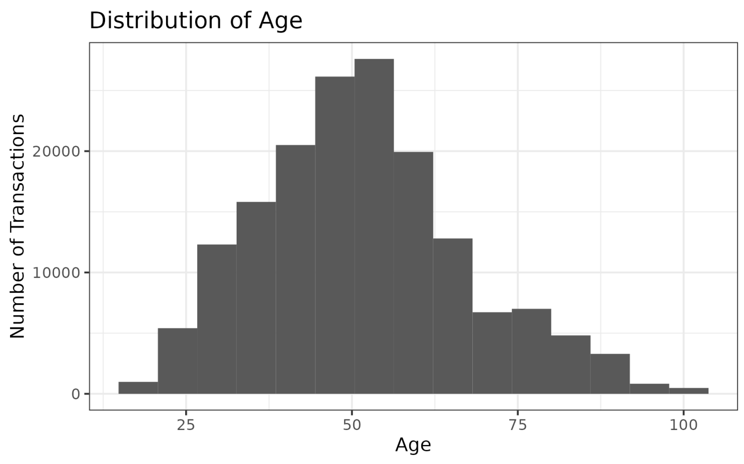
Gender shows the share of males and females in our data. It is summarized in Table 4.4. 55.54% of payments were made by women and 44.46% by men.

Table 4.4: Proportion of Genders

Gender	Proportion
Female	55.54%
Male	44.46%

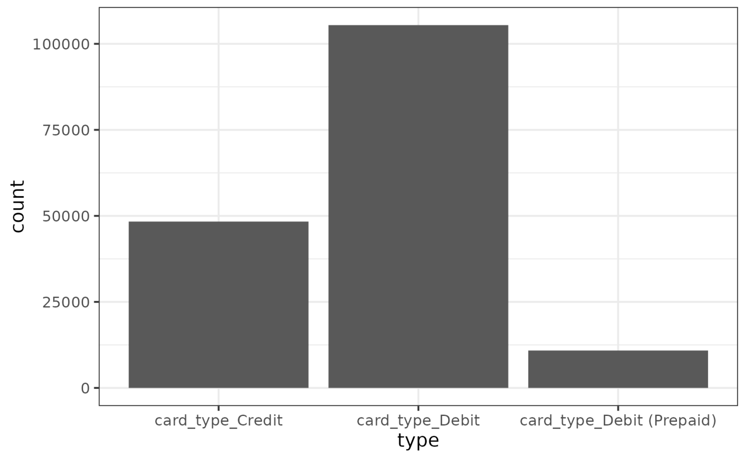
Variable *age* contains age of the cardholder at the time of the transaction. Figure 4.2 shows the distribution of cardholders' age in our dataset. This variable can be interesting because some age groups may be more likely to become victims of card fraud. For example, older people may fall for phishing attacks more often because they are not able to tell a scam e-mail from a genuine one as easily as younger generations.

Figure 4.2: Distribution of Age



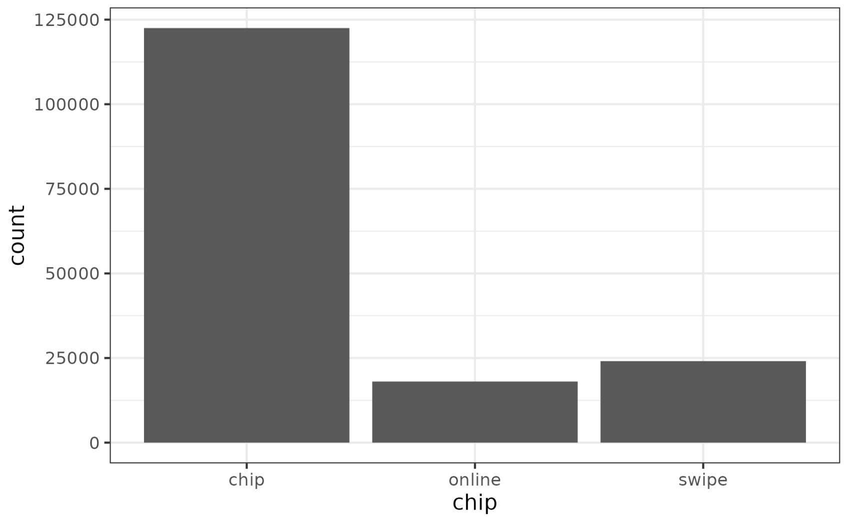
Card Type describes the distribution of different types of payment cards used. There are three main card types: credit cards, debit cards, and prepaid debit cards. The majority of transactions in our dataset were made by debit cards (64.04%). They are followed by credit cards with 29.36% of all transactions. The remaining 6.6% were made by prepaid cards. It is summarized in Figure 4.3

Figure 4.3: Distribution of Card Type



Use Chip shows the means of authorization of the card. Transactions were either made using the chip on the card at a POS, using the magnetic stripe and swiping the card, or using the card to pay online. This variable is important because it can be used to tell apart different fraud types which were explained in Chapter 2. Fraudulent online transactions can come from phishing attacks and POS transactions most often come from lost and stolen cards or less frequently from skimming attacks. The distribution of authorization methods is presented in Figure 4.4. Most transactions were authorized using a chip with 74.4%. The rest is split between swipe (14.63%) and online transactions which made up 10.97% of all payments.

Figure 4.4: Distribution of Chip Use



Variable *same_state* is a binary variable with value 1 when a transaction is made in a state that is different from the cardholder's state of permanent residence. This variable can be especially useful for the detection of online frauds such as CNP frauds. Distribution of *same_state* is presented in Table 4.5. Approximately one third of transactions in our sample were made in a state that is different from cardholders' permanent residence. Not every transaction from a different state should automatically be flagged as fraudulent but we believe that this will be an important variable in fraud detection. Especially given the popularity of phishing attacks in recent years.

Table 4.5: Proportion of *same_state*

Same state	Proportion
Yes	63.83%
No	36.17%

MCC is an interesting and possibly also an important variable in our analysis. Merchant Category Code (MCC) is a four-digit number that is used by card issuers to classify a business. They are most often assigned according to the main business activity of a merchant (e.g. VISA assigns 5944 to “Jewelry Stores, Watches, Clocks, and Silverware Stores”, or 4111 to “Suburban and Local Commuter Passenger Transportation, Including Ferries”). Some are also assigned to a single merchant, such as 3217 to Czech Airlines. These codes are mostly shared by all payment card issuers, although some may vary. There are various purposes for this categorization of businesses. MCCs can be used to determine the interchange fee a merchant pays to their bank when their customer pays with a card. Businesses and financial institutions use MCCs to track spending patterns and consumer behavior or to calculate credit card rewards to customers for spending on certain goods and services. We can use them to label businesses that may be connected with higher card fraud rates. One such high-risk MCC is 6051 which is assigned to “Non-Financial Institutions – Foreign Currency, Non-Fiat Currency (for example: Cryptocurrency), Money Orders (Not Money Transfer), Account Funding (not Stored Value Load), Travelers Cheques, and Debt Repayment”. Using this MCC, banks can limit or completely block purchases of cryptocurrencies since these transactions have a high risk of being fraudulent.

Since *MCC* is a categorical variable with many different categories, we need to modify it in order to use it in our models. It is possible to use one hot encoding the same way as we use it for the rest of our categorical variables. But the large number of different MCCs means that we would end up with tens or hundreds of binary variables. This would make our dataset more complicated and our models might suffer from training on an excessively large number of explanatory variables. Therefore we use learned embedding to transfer each category into a numerical variable. This way we can interpret all MCCs while using only a single variable that retains the original information. This will increase the training speed of our models and likely not harm their predictive performance. One downside is that we cannot easily tell which MCC is which since their values are scrambled into numerics.

Variable *has_chip* denotes whether the card has an embedded chip or if it only uses a magnetic stripe. Cards with no chip are potentially less secure because credit card credentials stored on a magnetic stripe can be accessed quite easily by fraudsters. Skimming devices are accessible and therefore cards that only use magnetic stripes can be more prone to skimming fraud. Table 4.6

shows the proportion of cards with and without chips. A vast majority of cards nowadays use a chip and cards with only a magnetic stripe are becoming rare. Cards without a chip are most likely among the oldest ones in our dataset.

Table 4.6: Proportion of Cards with Chip

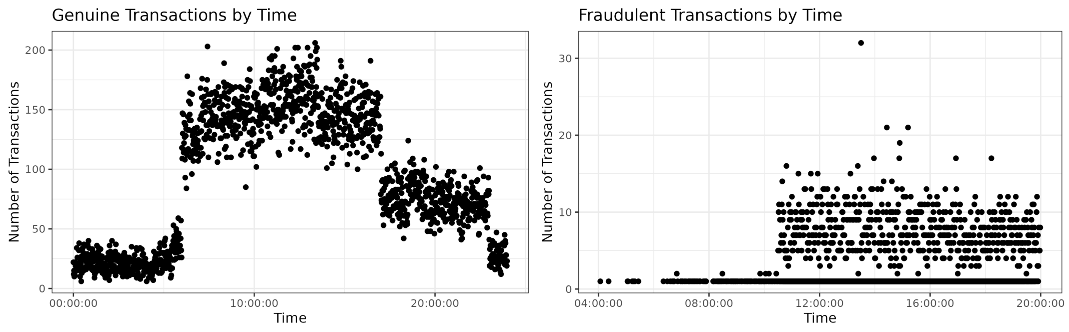
Has Chip	Proportion
Yes	91.06%
No	8.94%

Finally, we look at the distribution of transactions in time and the differences between genuine and fraudulent transactions. We present the comparison in Figure 4.5. Since clients in our dataset are located in one time zone, we can see that there are large differences in the number of genuine transactions during the day. There is a noticeable cluster of a high number of transactions approximately between 8 AM and 6 PM. This is the time of day when most economic activity happens and during this period people make most of their daily transactions. There is another cluster in the evening until 11 PM. This group is smaller both in number of transactions and in length. The last cluster is set during nighttime and contains the smallest number of transactions.

The comparison of distributions of fraudulent and genuine transactions is interesting. We can see that fraudulent transactions are grouped in one large cluster between 10 AM and 10 PM. The distribution of frauds during this time period seems uniform with a few outliers. On the other hand, between 10 PM and 10 AM, there were very few fraudulent transactions in our sample. Looking at both distributions, we can see that they do not look alike. Therefore, there is a certain pattern among fraudulent transactions that our models might recognize and use in their training. Since the majority of fraudulent transactions are made between 10 AM and 10 PM, we set up a variable *daytime* that labels transactions from this period. There is a higher probability that a transaction is fraudulent if it was made in this time period.

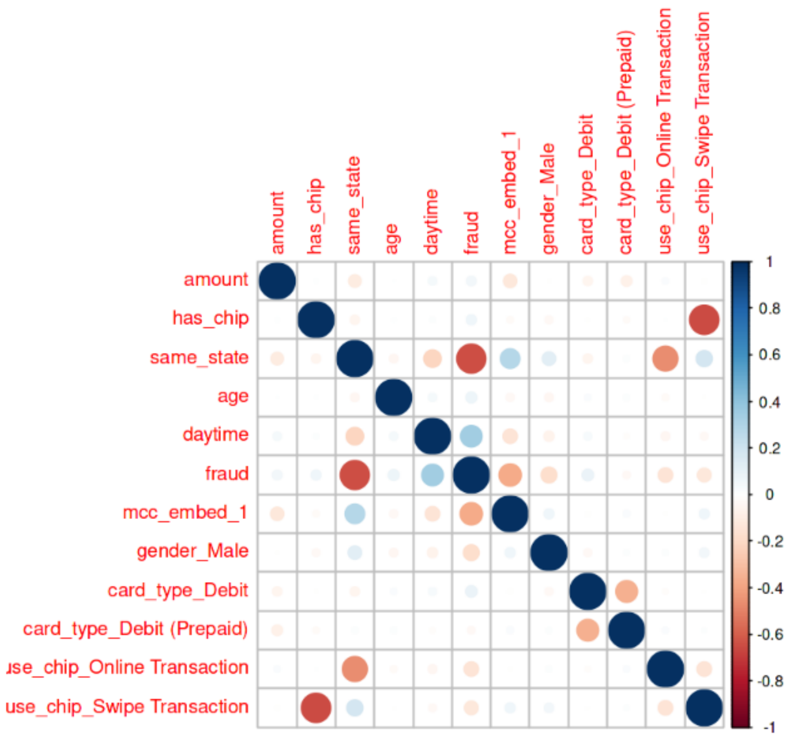
There is also an alternative approach to the way how time is used as a variable in the models. Bahnsen *et al.* (2016) propose to model the time of the transaction as a period variable using the von Mises distribution. This is especially useful for the computation of the mean time at which a transaction is made. This can then be used to create time confidence intervals for genuine transactions. A deviation from the confidence interval is then labeled as suspicious and potentially fraudulent.

Figure 4.5: Distribution in Time



To check correlation coefficients across variables, we make a correlation plot. Information on correlation can give us important insights into our dataset. We present the correlation matrix of our dataset in Figure 4.6.

Figure 4.6: Correlation Matrix of All Variables



We can see that fraud is negatively correlated with *same_state*. This means that transactions made in a different state are systematically more likely to be fraudulent. This is a feature that we are expecting to hold. There is also a negative correlation between transactions authorized using a magnetic stripe and transactions made with cards that have a chip. This makes sense since we can guess that people who have a card with a chip will not use the outdated magnetic stripe. Finally, we can see a negative correlation between online transactions and transactions made in the same state. This is also a valid relationship because there is a huge number of international online shops, and they are becoming more and more popular every year. Other correlation coefficients are not too large in magnitude.

Finally, we split our dataset into a training and testing sample. We randomly sample 70% of our dataset into a training set and we make the remaining 30% our testing set. On top of that, we sample another training set of 50 000 observations from our original imbalanced dataset. We do so because we are interested in the performance of our models on data with a real-life structure. If we only tested our models on the oversampled and balanced testing data, we might end up with models that only work well on balanced data but their performance drops when applied to imbalanced data. Therefore, we end up with a balanced training sample, a balanced testing sample, and an imbalanced testing sample. We summarize our data in Table 4.7.

Table 4.7: Properties of Used Subsamples

Name	Number of Observations	Proportion of Frauds
Training Sample	115 255	18.92%
Imbalanced Testing Sample	49 395	18.92%
Balanced Testing Sample	50 000	0.14%

Chapter 5

Estimation and Evaluation

In this section, we describe the process of building each of our models and identifying the importance of variables. We also describe the tuning of the parameters of each model so that they perform as best as possible. We train all of our models on the training sample which we have described in Chapter 4. Each model is provided with all of the available variables. After that, we extract variable importances and tune the model parameters.

There are two approaches to tuning the models. In this thesis, we want our models to maximize the number of transactions that are correctly classified. There is also a cost-sensitive approach (Sahin *et al.* 2013) which aims to minimize costs coming from the wrong classification. In this analysis, we are using the count-sensitive models because the bank system which we will use for comparison uses this approach as well. We want our models to be as comparable to the benchmark system as possible.

Models are tuned so that their performance on the imbalanced testing sample is maximized. In the case of fraud detection, the choice of a performance metric that we want to optimize is not as straightforward as it might seem. On the one hand, we want our algorithm to find as many fraudulent transactions as possible. On the other hand, the model must not produce too many false positive outcomes. In a real application of fraud detection, each transaction that is flagged as fraudulent is transferred to a new process. Such a transaction is either immediately blocked by the bank or has to be manually checked by an employee. In this case, the employee most often calls the card owner and asks them whether they were trying to make the suspicious transaction. This process is time-consuming and costly. The bank bears the cost of the employee's salary and the card owner has to spend time talking to the bank and waiting for their transaction to be approved. Therefore, we cannot simply aim for the highest number of uncovered fraudulent transactions without looking at the number of false positives.

In our analysis, we try to balance two performance metrics: sensitivity and specificity. Sensitivity refers to the ability of a model to correctly identify fraudulent transactions. It measures the proportion of frauds that were labeled as fraudulent. A model that has high sensitivity will therefore have a low false negative rate. Specificity, on the other hand, refers to the ability of an algorithm to correctly identify genuine transactions. Specificity measures the proportion of valid transactions that the model labeled as valid. A model with high specificity will give us only a few false positive cases. We are aiming to reach as high sensitivity as possible while keeping specificity at high enough levels.

We can also use precision as a complementary measure of performance. Precision describes the proportion of true positives among all transactions that were labeled as fraudulent. Higher precision is connected with a low false positive rate.

Further, we look at Area Under ROC Curve (AUC). In our case, high AUC only serves as a necessary but not sufficient condition to say that our model performs well. This is caused by the large imbalance between the number of fraudulent and genuine transactions in our dataset. When the majority class makes up a large proportion of the dataset, AUC can be misleading because it only measures the model's ability to rank examples, but it does not take into account the prevalence of the classes. In our case, we could receive a high AUC if our model simply labeled all transactions as legitimate even though this would make our model completely useless. On the other hand, we cannot completely ignore AUC because there is still a possibility that a model would wrongly label both fraudulent and legitimate transactions. Therefore, we should check that the models we train are able to reach high AUC but after that, we need to check other performance measures as well.

Precision and AUC are recommended as performance measures for machine learning models in fraud detection by Pozzolo *et al.* (2018) in their paper on fraud detection strategies. They also propose a cost-based measure. Since we want to compare our models to a fraud detection system used in a bank and we do not have a cost-based measure for this system, we are not going to use it in this analysis. Instead, we choose sensitivity and specificity.

5.1 Logistic Regression

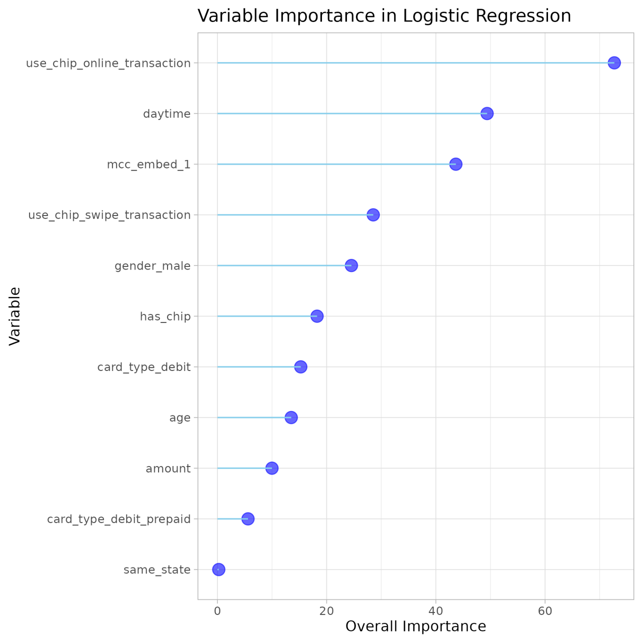
We use logistic regression as our first model for the detection of fraudulent transactions. Compared to other algorithms that we use in this analysis, logit is the simplest one. First, we set up a logistic regression of all available independent variables and check the performance of the model. Results of the model are presented in Table 5.1. We can see that all independent variables are highly significant. The only exception is *same_state* which has a very high p-value.

Table 5.1: Summary of Logistic Regression

	Dependent variable:
	fraud
amount	−0.002*** (0.0002)
has_chip	−1.925*** (0.105)
same_state	−22.513 (90.875)
age	0.019*** (0.001)
daytime	4.664*** (0.094)
mcc_embed_1	−5.154*** (0.118)
gender_male	−1.017*** (0.041)
card_type_debit	0.682*** (0.045)
card_type_debit_prepaid	0.508*** (0.091)
use_chip_online_transaction	−5.712*** (0.079)
use_chip_swipe_transaction	−3.017*** (0.106)
Constant	−1.821*** (0.161)
Observations	115,255
Log Likelihood	−10,837.000
Akaike Inf. Crit.	21,698.010
Note:	*p<0.1; **p<0.05; ***p<0.01

This behavior is somewhat puzzling since we were expecting *same_state* to be an important predictor of fraudulent transactions. An analysis of the importance of each dependent variable further confirms this claim. Figure 5.1 shows that *same_state* is by far the least important variable from our selection. Since this is a logistic regression, we cannot use the standard R^2 metric to see the proportion of variance of our dependent variable that is explained by the independent variables. Instead, it is necessary to use a pseudo-R^2. In this analysis, we use McFadden's R^2. Its value is 0.80, which indicates a very good fit (Domencich & McFadden 1975).

Figure 5.1: Variable Importance of Logistic Regression



We use the imbalanced testing sample to obtain predictions from the logistic regression. The model shows a very good performance. Figure 5.2 presents a confusion matrix of the actual and predicted values, and also shows sensitivity and specificity. These results are very good given that it is a simple logistic regression. Out of the 71 fraudulent transactions present in the testing sample, the model managed to find 63, reaching a sensitivity of 88.73%. Further, it produced 2491 false positive predictions which means that it operated with a specificity of 95.01%. There are also levels of accuracy, F1-score, and kappa as secondary performance metrics in Figure 5.2.

In a real-life application of a fraud detection system, it is also important to look at the speed of such a system. Since there can be thousands of card transactions made each minute in a bank, the model has to be able to label transactions as fraudulent or genuine in real-time. Therefore, we also look at the average time it takes our model to predict on the testing sample. We do not look at the training time because the model would not need to be retrained too often and because it is possible to train the model separately from live data. This logistic regression predicted all of the 50 000 transactions from the testing sample in 0.0055 seconds on average from 100 runs.

Good performance is confirmed by the Receiver Operating Characteristic (ROC) curve, which is presented in Figure 5.3. The magnitude of AUC reaches

0.967, which is a good result. Overall, this model seems to be able to predict fraudulent transactions well. It was able to find a large proportion of frauds in the testing sample while keeping the number of false positives reasonably high.

Figure 5.2: Confusion Matrix of Logistic Regression

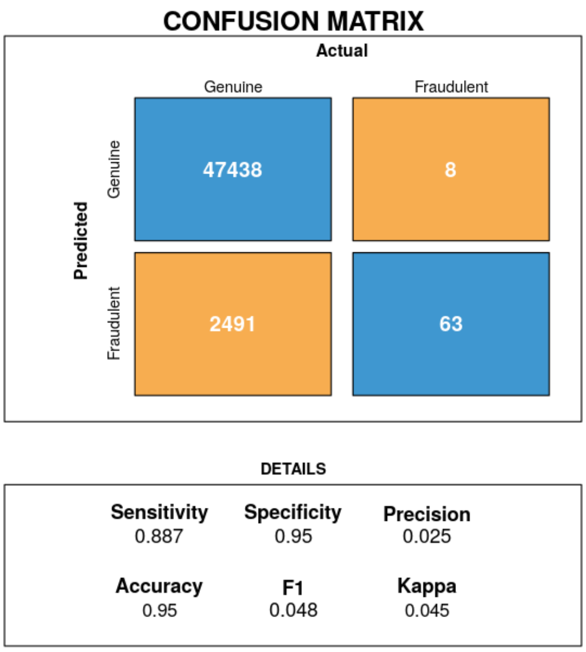
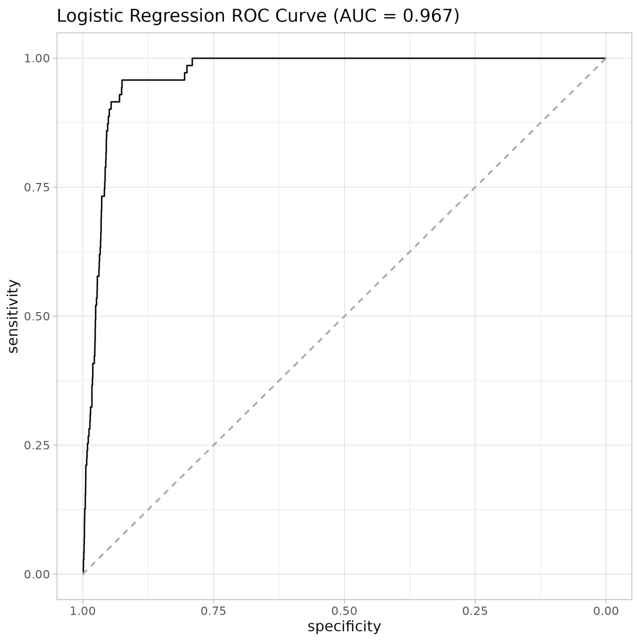


Figure 5.3: ROC Curve of Logistic Regression



To address the issue of insignificant *same_state*, we make some changes to our regression and run it again. First, we start removing explanatory variables and we check the drop in McFadden's R^2 with each removed variable. This way we identify variables that contain little information that is useful for explaining fraud and we remove these variables. Table 5.2 shows the second logistic regression after the removal of unnecessary variables. McFadden's R^2 drops from 0.80 to 0.78.

Table 5.2: Summary of Logistic Regression 2

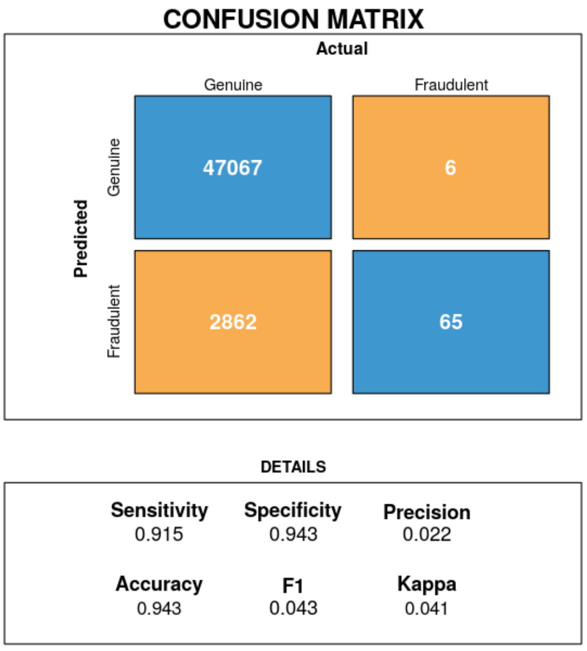
	Dependent variable:
	fraud
same_state	−22.544 (93.364)
daytime	4.675*** (0.092)
use_chip_online_transaction	−5.208*** (0.070)
age	0.022*** (0.001)
amount	−0.002*** (0.0002)
mcc_embed_1	−5.052*** (0.112)
Constant	−3.914*** (0.118)
Observations	115,255
Log Likelihood	−11,849.120
Akaike Inf. Crit.	23,712.240
Note:	*p<0.1; **p<0.05; ***p<0.01

Predictive performance of the second logistic regression is summarized in Figure 5.4. Overall, this regression reports a larger number of fraudulent transactions. This means that it found more frauds at the cost of producing a higher number of false positives. Sensitivity increases from 88.73% to 91.55% while Specificity drops from 95.01% to 94.27%. Within these two regressions, there is a clear trade-off between sensitivity and specificity in their performance.

In terms of AUC, the second model performs slightly worse. Figure 5.5 shows that the model achieves an AUC of 0.964 while the first regression reached 0.967. The time to label all transactions from the testing sample drops by 32.7% to 0.0037 seconds. This is a major improvement over the original logistic regression. We can attribute this to the simplification of the model because it relies on a smaller number of variables.

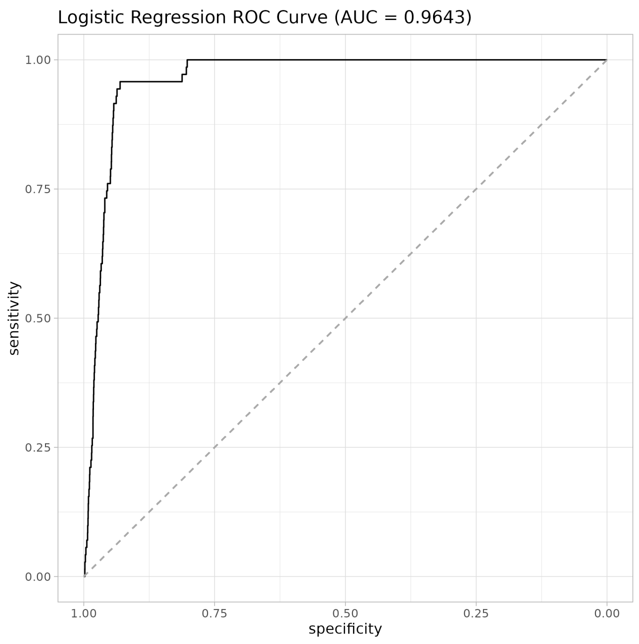
In this regression, *same_state* is still highly insignificant. Estimates of explanatory variables have not changed too much and all variables keep their significance. Thus, we run a bootstrap with 1000 resamples on the first model.

Figure 5.4: Confusion Matrix of Logistic Regression 2



Sensitivity	Specificity	Precision
0.915	0.943	0.022
Accuracy	F1	Kappa
0.943	0.043	0.041

Figure 5.5: ROC Curve of Logistic Regression 2



By bootstrapping, we obtain much more precise standard errors for each predictor. This is possible because bootstrapping draws a large number of resampled datasets from the original sample. The logistic regression is then run on each of these datasets and standard errors are calculated for every regression. After that, the algorithm examines the distribution of these standard errors and provides us with a more precise estimate.

Table 5.3 shows that after bootstrapping, the standard error of *same_state* drops massively, which means that it is a statistically significant variable. Standard errors of all other variables are very similar to estimates from the logistic regression. Therefore, the original regression behaves as expected.

Table 5.3: Bootstrap Standard Errors of First Logistic Regression

	original	bootBias	bootSE	bootMed
(Intercept)	-1.8205	0.0083	0.1473	-1.8065
amount	-0.0016	0.0000	0.0001	-0.0016
has_chip	-1.9247	-0.0050	0.0856	-1.9327
same_state	-22.5127	-0.0021	0.0339	-22.5153
age	0.0186228	-0.0000	0.0016	0.0185
daytime	4.6643	0.0007	0.0973	4.6672
mcc_embed_1	-5.1541	-0.0026	0.1219	-5.1574
gender_male	-1.0167	-0.0018	0.0448	-1.0177
card_type_debit	0.6815	-0.0003	0.0466	0.6801
card_type_debit_prepaid	0.5076	0.0059	0.1087	0.5128
use_chip_online_transaction	-5.7120	0.0016	0.0868	-5.7083
use_chip_swipe_transaction	-3.0167	-0.0038	0.0855	-3.0233

Table 5.4 compares our two logistic regression models. The first regression produces a smaller number of false positives and has higher precision and AUC, whereas the second model correctly identifies a higher number of frauds and also takes significantly less time to predict on the testing sample.

Table 5.4: Comparison of Logistic Regressions

Model	Sensitivity	Specificity	Precision	AUC	Training Time (s)
Logit All Variables	0.8873	0.9501	0.0247	0.9670	0.0055
Logit Selected Variables	0.9155	0.9427	0.0222	0.9643	0.0037

5.2 Neural Network

We continue our analysis with a neural network. Unlike logistic regression, a neural network is a "black-box" algorithm. This means that we cannot see how the learning process works and we do not know why exactly the neural network decided to label a transaction as genuine or fraudulent. While logistic regression uses a linear combination of explanatory variables, neural networks use one or more hidden compute layers, which in sequence transform explanatory variables into the explained variable.

The neural network is built using the R package *nnet*. This package only allows for neural networks with one hidden layer. It is possible to tune the network by setting the number of neurons in the hidden layer (*size*) and a parameter *decay*. This parameter is a form of regularization, that is used to limit overfitting (Venables & Ripley 2002).

In order to find optimal values of the two parameters, we set up vectors with four different values of *size* and eight values of *decay*. We then perform a grid search, which uses all 32 combinations of these parameters to find a model with as high performance as possible. This method uses brute force in order to find the optimal combination of parameters. We also perform a 2-fold cross-validation which is repeated two times with a different set of folds. This will further improve the accuracy and robustness of the found parameters.

Grid search finds optimal parameters for the neural network. Its performance on the test set is summarized in Figure 5.6. The algorithm managed to correctly identify 62 fraudulent transactions out of 71, which translates to a sensitivity of 87.3%. It also made 2353 false positive predictions. Specificity is therefore 95.3%. In comparison to both logistic regressions, the neural identified a slightly smaller number of frauds while producing fewer false positives. The model took 0.04 seconds on average to predict on 50 thousand transactions, which is approximately 8-10 times as long in comparison to the logistic regressions. This is a very large difference, even though the neural network should still be able to predict comfortably in a real-world application.

The model performs well given the AUC level of 0.943, as shown in Figure 5.7. This is a high value of AUC, but lower than the AUC of both logistic regressions. This is caused by a lower number of identified frauds despite the higher specificity.

Finally, we check the importance of each variable in the training of the neural network. This is presented in Figure 5.8. Compared to logistic regres-

Figure 5.6: Confusion Matrix of Neural Network

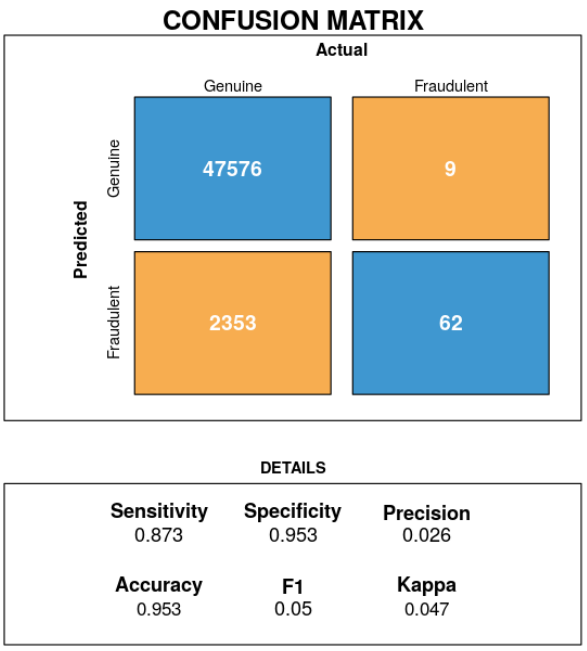
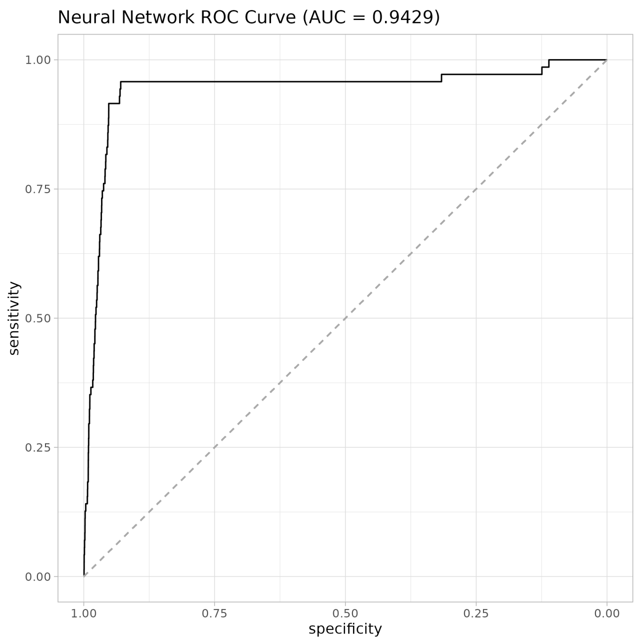
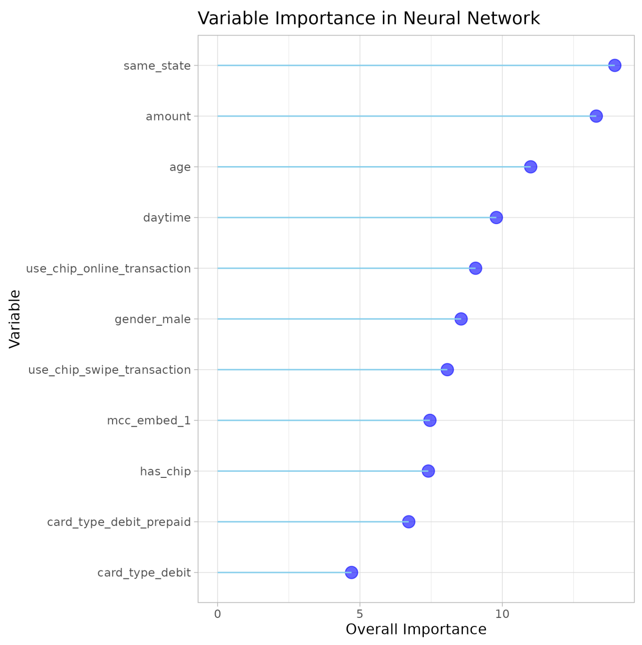


Figure 5.7: ROC Curve of Neural Network



sions, the dispersion in variable importance is much smaller in the case of the neural network. The most important variable is approximately three times as important as the least important variable. Therefore, the removal of any variable would likely harm the predictive performance of the model. Variable importance for the neural network is based on combinations of absolute values of weights for each variable (Gevrey *et al.* 2003).

Figure 5.8: Variable Importance of Neural Network



5.3 Random Forest

The next model that we use for fraud identification is a random forest. There are multiple different implementations of random forests available in R and we are using one made by company h2o.ai in their package *H2O*. This package provides an open-source platform for building of a broad range of machine learning models, including deep learning, gradient boosting, or generalized linear models.

Random forests trained using the h2o package can be tuned using a large number of parameters. We chose to optimize the model by tuning parameters *mtries*, *max_depth*, *min_rows* and *sample_rate*. *Mtries* defines the number of variables randomly sampled as candidates at each split. *Max_depth* is the maximum depth of a single tree. *Min_rows* sets the fewest allowed observations in a leaf and *sample_rate* defines the proportion of the number of rows that are sampled for each tree from the training dataset. Same as in the case of a neural network, the decision process of a random forest cannot be simply observed because it is an ensemble of a large number of smaller tree models. In this case, the random forest consists of 1000 decision trees.

We run a grid search across 225 combinations of selected parameters to find a model with as good predictive performance as possible. Further, the random forest also uses 5-fold cross-validation to ensure that the best parameter values are chosen.

The predictive performance of the final model is presented in Figure 5.9. Interestingly, the random forest finds a much smaller number of fraudulent transactions. It managed to correctly identify 48 frauds and reached a sensitivity of 67.6%. This is a much lower number of true positives compared to the logistic regressions and the neural network. On the other hand, the random forest also produces much fewer false positives than the prior models and operates with a specificity of 97.1%. This is another example of the trade-off between true positives and false positives that many real fraud detection systems have to face. While the random forest certainly outperforms other models in the number of false alarms it produces, its ability to find a high number of actual frauds is limited.

Figure 5.9: Confusion Matrix of Random Forest 1

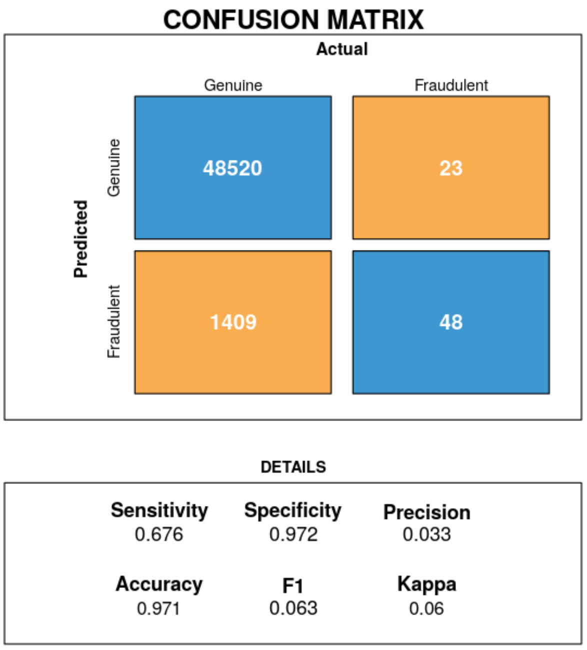
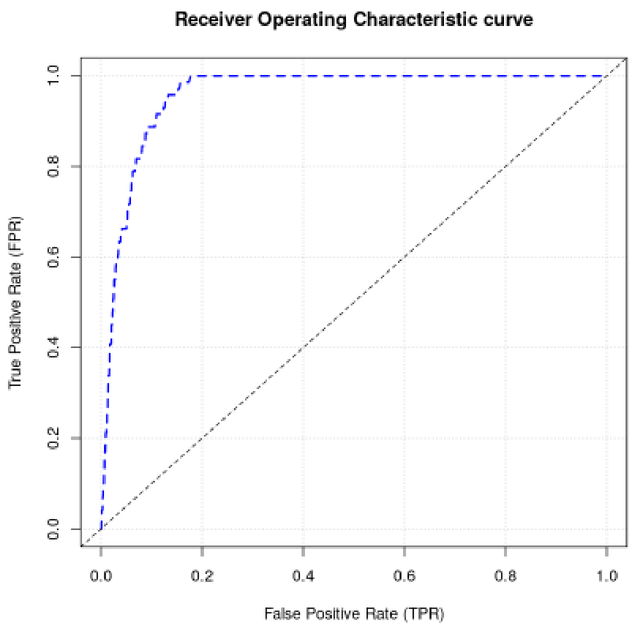


Figure 5.10 shows ROC curve of the random forest model. Area under the curve equals 0.9728 which is higher compared to both logistic regression and neural network models. This is however carried only by the lower false positive rate and not the true positive rate. The average time required for a prediction on the testing sample rose to 10.4 seconds. This is a massive increase from the sub-0.1 second prediction time of logistic regressions and the neural network. This may hint that the random forest model grew very complex even though its performance is lacking.

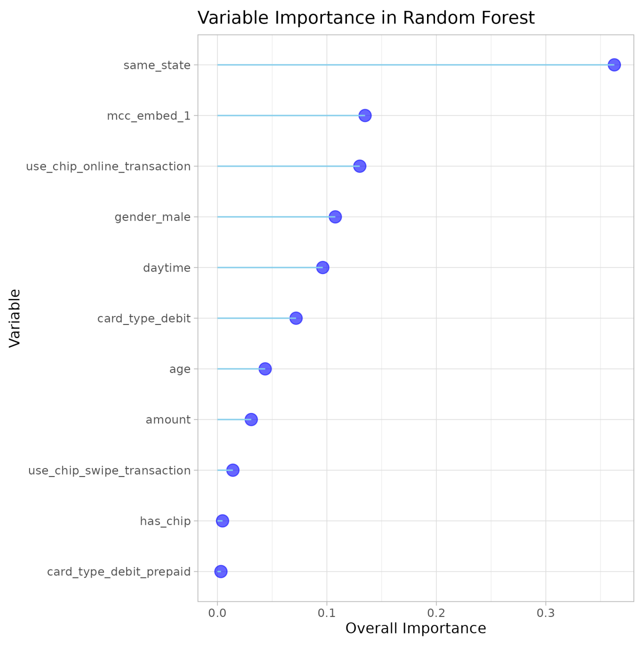
Figure 5.10: ROC Curve of Random Forest 1



Further, we inspect a plot of variable importance presented in Figure 5.11. Unlike the neural network, our random forest treats individual variables differently. There is a large spread in variable importance. The information whether a card has a chip, or if it is a prepaid card has very little overall value for the model. These variables may therefore only make the model too complex despite having little influence on its outcome. We will therefore run the model once more while removing the three least important variables. The remaining variables have comparable importance, except for *same_state*.

We run the random forest again on a smaller number of explanatory variables. Results are shown in Figure 5.12. We can see that the predictive performance of the algorithm has slightly dropped. Both sensitivity and specificity are lower in comparison to the original random forest. On the other hand, the

Figure 5.11: Variable Importance of Random Forest 1

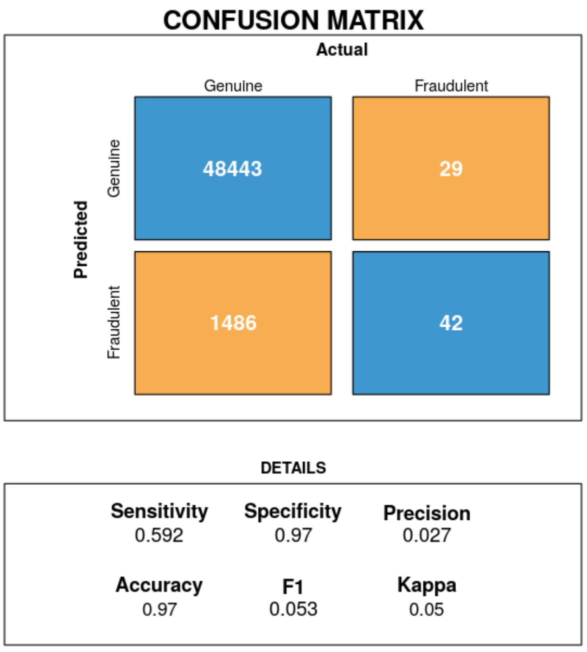


mean time to predict on the testing sample decreased to 8.7 seconds. This is 16% lower but compared to the other models it is still very high. Neither the second random forest is therefore a good candidate for a card fraud detection system.

5.4 Gradient Boosting

The final algorithm of this analysis is gradient boosting. We use Extreme Gradient Boosting and its implementation through R package *xgboost*. This is a specialized package for XGBoost and it offers a high level of customization for this task. It offers both a linear and a tree-based booster, both with a large selection of tuning parameters. For our task, we use a tree-based model. The model is tuned using parameters *max_depth*, *eta*, *nrounds*, *max_delta_step* and *lambda*. *Max_depth* sets the largest depth a tree can grow. *Eta* is used to control the learning rate. It scales the contribution of each tree and it is used to prevent overfitting by making the boosting process more conservative with the lower value of eta. *Nrounds* specifies the maximum number of boosting iterations. *Max_delta_step* is used to improve performance on imbalanced datasets. *Lambda* is a weight regularization term that helps prevent overfitting

Figure 5.12: Confusion Matrix of Random Forest 2

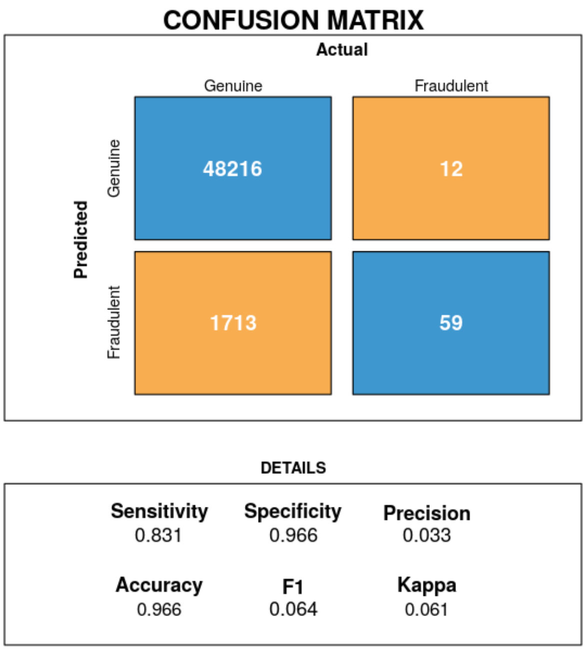


and makes the model more conservative. The algorithm is set up so that it predicts the probability of the transaction being fraudulent.

Figure 5.13 shows the predictive performance of the model given optimal parameter tuning. XGBoost managed to correctly identify 59 frauds out of 71 from the imbalanced testing sample, which translates into a sensitivity of 83.1%. This is not the highest number of true positives compared to other models used in this analysis. But together with a specificity of 96.6%, this means that the model managed to limit the number of false positives it produced. 1713 false positive fraudulent transactions are the second lowest number of all the tested models. Only the random forest kept this number lower, but it struggled to correctly predict true positives. XGBoost also shares the highest precision with the random forest.

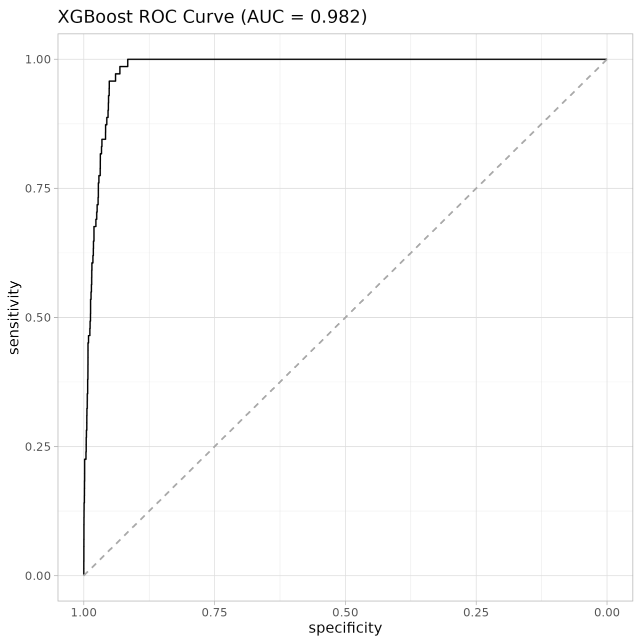
The area under the ROC curve of this algorithm reached a score of 0.9820, as shown in Figure 5.14. This is the highest AUC of all the tested models and it further hints that this model outperforms others. Both high sensitivity and specificity contribute to the AUC score. Further, the average time to predict on the test sample of 50 000 transactions was 0.0007 seconds. This is by far the lowest prediction time, as it is approximately five times lower than the next model. XGBoost is therefore very good at finding fraudulent transactions and

Figure 5.13: Confusion Matrix of XGBoost


CONFUSION MATRIX
Actual
Genuine
Fraudulent
Predicted
Genuine
Fraudulent
48216
12
1713
59
DETAILS
Sensitivity
0.831
Specificity
0.966
Precision
0.033
Accuracy
0.966
F1
0.064
Kappa
0.061

also manages to do so very fast.

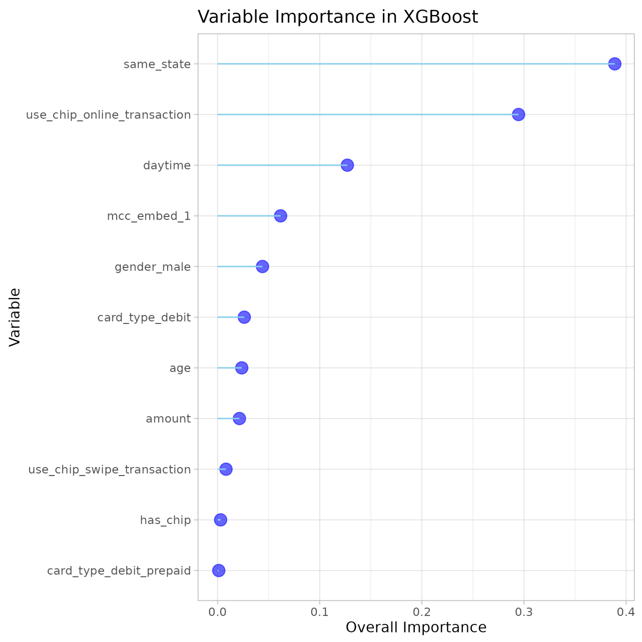
Figure 5.14: ROC Curve of XGBoost



In Figure 5.15, we look at variable importance in the model. For extreme gradient boosting, variable importance is calculated as an improvement in accuracy brought by the variable to the branches it is on. There is a clear difference between the importance of different variables,*same_state* and *use_chip_online_transaction* being the most important ones. Other variables contribute significantly less to the model. Given the very high performance of the model, it is not necessary to retrain the model with a smaller selection of explanatory variables. Moreover, the prediction speed is already very high as well, and therefore the complexity of the model is not an issue either.

While there are some differences, the order of variables by their importance is comparable in gradient boosting, logistic regression, and random forest models. The algorithms mostly differ in the way they work with the variables and in the way they use the information brought by them.

Figure 5.15: Variable Importance of XGBoost



Chapter 6

Comparison to a Real Fraud Detection System

The models that we have tested seem to have performed well in the detection of fraudulent transactions. The only exception is the random forest which identified a low number of frauds even though it kept the number of false positives low as well. But there was no clear benchmark in the analysis, to which we could compare the performance of our models. There was no connection to actual fraud detection systems used in banking.

Therefore, we use data from a card fraud detection system that is used in a middle-sized Czech bank. This is a traditional rule-based fraud detection system, as described in Chapter 2. This makes it a good example for comparison between the traditional systems and modern solutions which make use of some machine learning techniques. But the comparison is not perfect. While the bank system uses actual transactional data, models from this analysis are built on a synthetic dataset which was presented in Chapter 4. This means that we cannot directly compare the performance between our models and the bank detection system. But the comparison should still have some relevance and give us more insight into the performance of our models.

Data on the performance of the bank detection system comes from the year 2022. This year, the system analyzed approximately 204 million card transactions. Its performance is summarized in Table 6.1. The first thing we can immediately see is that this system achieved a very high specificity, meaning that it produced a very low number of false positives. This is an important feature of the model because a higher rate of false positives translates into higher expenses on the salaries of employees who verify fraudulent transactions.

Table 6.1: Performance of Benchmark FDS

Sensitivity	0.7845
Specificity	0.9996
Precision	0.0643

Further, we create a performance summary of all our models. This summary is presented in Table 6.2. In terms of sensitivity only, our best-performing model is the logistic regression trained on a selection of the most important variables. This model operated with a sensitivity of 91.6% but it lacked in terms of specificity, where it ended in the last place.

Not taking into account the bank detection system, the highest sensitivity was reached by the random forest with a score of 96.6%. Though again this model placed last in sensitivity. Moreover, the random forest took by far the longest to predict on the testing sample. This shows that no "wonder" model dominates others in all of the observed performance metrics. There will always be a trade-off between the proportion of true positives and false positives, and it is up to the model creator to decide which model suits their needs best.

It is, therefore, necessary to look at other performance metrics besides just sensitivity and specificity to identify the best-performing model. We have also looked at the AUC for each model in this analysis. But this is just a subsidiary performance metric since all of the models achieve high AUC. This is caused by the fact that we use an imbalanced dataset, and the majority class boosts AUC significantly. But a high AUC serves as an indicator that all of the tested models were able to adapt to the data and find information that could be used to classify frauds.

There is still precision as the remaining performance metric of interest. Random forest and gradient boosting models share the highest precision of 0.033. But the random forest suffers from a very low true positivity rate and only makes up for the high precision with a low number of false positives. On the other hand, XGBoost does not show such weaknesses. While it does not have the highest sensitivity or specificity, they are both high enough, so that the overall performance beats all the other tested models.

Moreover, its prediction speed is by far the highest and therefore it can process a huge number of transactions in real time. Though it is safe to say that all of the models should be able to predict in real-time if they were implemented in the bank whose data we use. Given that there were 204 million transactions in a year, then there were approximately 6.5 transactions made every second on average. Even if we take into account some seasonality and occurrence of time periods with a much larger average number of transactions, then no models should still have problems keeping up with the inflow of transactions. The by far slowest random forest model managed to predict 4800 transactions a second on average. Therefore, the speed of gradient boosting is only a bonus to its predictive performance, but it does not have to be taken into account in model selection.

Table 6.2: Comparison of All Models

Model	Sensitivity	Specificity	Precision	AUC	Time (s)
Logit All Variables	0.887	0.950	0.025	0.967	0.0055
Logit Selected Variables	0.916	0.943	0.022	0.964	0.0037
Neural Network	0.873	0.953	0.026	0.943	0.04
Random Forest	0.676	0.972	0.033	0.973	10.4
Gradient Boostig	0.831	0.966	0.033	0.982	0.0007
Bank Detection System	0.784	0.999	0.064	X	X

Gradient boosting is the best algorithm that we have tested for card fraud detection. But it is necessary to compare it to the bank fraud detection system. It could be possible that the XGBoost model is just the best of not very good models. Table 6.2 shows that gradient boosting managed to find a larger proportion of fraudulent transactions and therefore had a larger sensitivity. On the other hand, the bank detection system operates with nearly perfect specificity and produces very few false positives in its predictions. There is approximately a 3.3 percentage points difference in specificity between our gradient boosting model and the bank system. While it may not seem like a significant difference, it can have a large impact on the number total number of false positive alerts.

If the XGBoost model were implemented on the 204 million yearly transactions in the bank, it would produce around seven million false positives. Meanwhile, the bank system operated with only 70 thousand false positive predictions. This is a hundredfold difference, which would lead to a massive increase in the number of employees who manually check suspicious transactions or to the discontent of clients whose transactions were blocked. Either way, this would likely lead to large expenses and losses for the bank. Therefore, it seems that the current detection system is a superior solution to a machine learning model.

But the verdict cannot be so clear because the bank model operated on a different dataset. It is possible that the gradient boosting model would perform better if it was trained using data from the bank. The fraud risk specialist could also decide to prioritize specificity over sensitivity so that the model would produce fewer false positives while achieving a high enough level of fraud detection. With sufficient tuning, a fraud detection system based on extreme gradient boosting would likely perform as well as a standard rule-based system.

Chapter 7

Conclusion

The goal of this thesis was to train multiple machine learning models and use them for the identification of fraudulent transactions made with payment cards and to decide whether or not these models are suitable for this task. The tested algorithms were logistic regression, neural network, random forest, and gradient boosting in the form of XGBoost. Further, the thesis aims to find out whether ensemble machine learning methods are superior to standard models in fraud detection. Finally, we try to decide if machine learning models perform better than traditional rule-based card fraud detection systems and if they could be used in a real-world application.

The thesis first presented current trends in card fraud. The most common types of fraud have been discussed. Then the various methods of card fraud detection have been described. After that, the theoretical background for all of the models was presented. Then the data and their properties were described. The dataset used in this analysis has been simulated in a way that closely resembles real transactional data. Since there is a very small proportion of frauds among credit card transactions, the data had to be oversampled to improve model training.

Some of the benefits of machine learning models are their flexibility and simple implementation. On the other hand, they also have disadvantages. Most of them are black-box models and therefore their operations cannot be easily interpreted.

None of the tested models surpassed the benchmark fraud detection system. But all of them achieved an acceptable predictive performance. There was quite a significant variation in performance metrics across the models. Overall, random forest and gradient boosting operated with higher precision than the neural network and logistic regression. Random Forest fell behind in sensitivity, though. This partially confirmed our initial hypothesis, that ensemble models are a better choice for card fraud detection.

We also measure the prediction speed of all our models. While there were large differences in the time it took each algorithm to predict on a sample of 50 000 transactions, all of the models would be able to keep up with the inflow of transactions in real-time.

The best algorithm from our analysis is extreme gradient boosting as it outperforms the remaining models. But it lacks specificity when compared to a rule-based system that is used for real-world fraud detection in a middle-sized Czech bank. While XGBoost can find a larger proportion of fraudulent transactions, it also produces a much larger number of false positives than the bank system. Since our models and the bank model have been trained on a different dataset, their performance cannot be directly compared. We conclude that the gradient boosting algorithm could be tuned to outperform the bank detection system if trained on the actual bank data. This confirms our hypothesis that machine learning models are a viable alternative to existing rule-based models that are mostly set up manually.

To sum up, machine learning models are an interesting substitute for traditional fraud detection systems. While these systems have proven themselves over time, they can be difficult to set up and they need constant manual tweaks. Machine learning models and especially gradient boosting offer a quick and reliable way to identify fraudulent transactions. With proper tuning, they could outperform standard fraud detection systems while requiring much less maintenance.

The contribution of this thesis lies in the comparison of machine learning models to the currently used methods in fraud detection. Moreover, it compares multiple algorithms, from simple logistic regression to more complex models. An extension of this thesis could analyze the performance of these models on real data used by traditional fraud detection systems. This would shed more light on the comparison of their performance. If the benchmark bank model used a cost-sensitive performance measure, it would also be interesting to use this measure for our models and compare them to the bank system once again.

Bibliography

ABDALLAH, A., M. A. MAAROF, & A. ZAINAL (2016): “Fraud detection system.” *Journal of Network and Computer Applications* **68**: pp. 90–113.

ALTMAN, E. (2019): “Synthesizing credit card transactions.”

BAHNSEN, A. C., D. AOUADA, A. STOJANOVIC, & B. OTTERSTEN (2016): “Feature engineering strategies for credit card fraud detection.” *Expert Systems with Applications* **51**: pp. 134–142.

BENTÉJAC, C., A. CSÖRGŐ, & G. MARTÍNEZ-MUÑOZ (2021): “A comparative analysis of gradient boosting algorithms.” *Artificial Intelligence Review* **54(3)**: pp. 1937–1967.

BISHOP, C. M. (2006): *Pattern recognition and machine learning.* New York: Springer.

BOLTON, R. & D. HAND (2001): “Unsupervised profiling methods for fraud detection.” *Conference on Credit Scoring and Credit Control* **7**.

BREIMAN, L. (2001): “Random forests.” *Machine Learning* **45(1)**: pp. 5–32.

CHAN, P., W. FAN, A. PRODROMIDIS, & S. STOLFO (1999): “Distributed data mining in credit card fraud detection.” *IEEE Intelligent Systems* **14(6)**: pp. 67–74.

CHANDOLA, V., A. BANERJEE, & V. KUMAR (2009): “Anomaly detection.” *ACM Computing Surveys* **41(3)**: pp. 1–58.

CHAWLA, N. V., K. W. BOWYER, L. O. HALL, & W. P. KEGELMEYER (2002): “Smote.” *Journal of Artificial Intelligence Research* **16**: pp. 321–357.

CHEN, T. & C. GUESTRIN (2016): “Xgboost: A scalable tree boosting system.” *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* pp. 785–794.

DOMENCICH, T. A. & D. L. MCFADDEN (1975): *Urban Travel Demand.* Amsterdam: North-Holland, 1st edition.

EBA (2022): "Discussion Paper on the EBA's preliminary observations on selected payment fraud data under PSD2, as reported by the industry." `https://www.eba.europa.eu/eba-publishes-discussion-paper-its-preliminary-observations-selected-payment-fraud-data-under`.

EDGE, M. E. & P. R. F. SAMPAIO (2009): "A survey of signature based methods for financial fraud detection." *Computers & Security* **28(6)**: pp. 381–394.

FRIEDMAN, J. H. (2001): "Greedy function approximation." *The Annals of Statistics* **29(5)**.

GEVREY, M., I. DIMOPOULOS, & S. LEK (2003): "Review and comparison of methods to study the contribution of variables in artificial neural network models." *Ecological Modelling* **160(3)**: pp. 249–264.

GHOSH & REILLY (1994): "Credit card fraud detection with a neural-network." *Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences HICSS-94* pp. 621–630.

GOODFELLOW, I., Y. BENGIO, & A. COURVILLE (2016): *Deep learning.* Cambridge: MIT Press.

GUPTA, S. (2016): *Deep Learning vs. traditional Machine Learning algorithms used in Credit Card Fraud Detection.* Master's thesis, National College of Ireland, Dublin.

HAIXIANG, G., L. YIJING, J. SHANG, G. MINGYUN, H. YUANYUE, & G. BING (2017): "Learning from class-imbalanced data." *Expert Systems with Applications* **73**: pp. 220–239.

HASTIE, T., R. TIBSHIRANI, & J. H. FRIEDMAN (2009): *The elements of statistical learning.* New York: Springer, 2nd edition.

HO, T. K. (1995): "Random decision forests." *Proceedings of 3rd International Conference on Document Analysis and Recognition* pp. 278–282.

KAUR, H., H. S. PANNU, & A. K. MALHI (2020): "A systematic review on imbalanced data challenges in machine learning." *ACM Computing Surveys* **52(4)**: pp. 1–36.

Lee, D. D. & H. S. Seung (1999): "Learning the parts of objects by non-negative matrix factorization." *Nature* **401(6755)**: pp. 788–791.

Madkaikar, K., M. Nagvekar, P. Parab, R. Raikar, & S. Patil (2021): "Credit card fraud detection system." *International Journal of Recent Technology and Engineering* **10(2)**: pp. 2277–3878.

Mason, L., J. Baxter, P. Barlett, & M. Frean (2000): "Boosting algorithms as gradient descent." *NIPS* **12**: pp. 512 – 518.

McCulloch, W. S. & W. Pitts (1943): "A logical calculus of the ideas immanent in nervous activity." *The Bulletin of Mathematical Biophysics* **5(4)**: pp. 115–133.

Mitchell, T. M. (1997): *Machine learning.* Boston: McGraw-Hill, 1st edition.

Mqadi, N., N. Naicker, & T. Adeliyi (2021): "A smote based oversampling data-point approach to solving the credit card data imbalance problem in financial fraud detection." *International Journal of Computing and Digital Systems* **10(1)**: pp. 277–286.

Murphy, K. P. (2012): *Machine Learning: a probabilistic perspective.* Cambridge, MA: MIT Press, 1st edition.

Nami, S. & M. Shajari (2018): "Cost-sensitive payment card fraud detection based on dynamic random forest and k -nearest neighbors." *Expert Systems with Applications* **110**: pp. 381–392.

Ngai, E. W., Y. Hu, Y. Wong, Y. Chen, & X. Sun (2011): "The application of data mining techniques in financial fraud detection." *Decision Support Systems* **50(3)**: pp. 559–569.

Pozzolo, A. D., G. Boracchi, O. Caelen, C. Alippi, & G. Bontempi (2015): "Credit card fraud detection and concept-drift adaptation with delayed supervised information." *2015 International Joint Conference on Neural Networks (IJCNN)* pp. 1–8.

Pozzolo, A. D., G. Boracchi, O. Caelen, C. Alippi, & G. Bontempi (2018): "Credit card fraud detection." *IEEE Transactions on Neural Networks and Learning Systems* **29(8)**: pp. 3784–3797.

RAI, A. K. & R. K. DWIVEDI (2020): "Fraud detection in credit card data using unsupervised machine learning based scheme." *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)* pp. 421–426.

RANDHAWA, K., C. K. LOO, M. SEERA, C. P. LIM, & A. K. NANDI (2018): "Credit card fraud detection using adaboost and majority voting." *IEEE Access* **6**: pp. 14277–14284.

SAHIN, Y., S. BULKAN, & E. DUMAN (2013): "A cost-sensitive decision tree approach for fraud detection." *Expert Systems with Applications* **40(15)**: pp. 5916–5923.

SCHAPIRE, R. E. & Y. FREUND (2012): *Boosting.* Cambridge, MA, USA: The MIT Press, 1st edition.

TAHA, A. A. & S. J. MALEBARY (2020): "An intelligent approach to credit card fraud detection using an optimized light gradient boosting machine." *IEEE Access* **8**: pp. 25579–25587.

THE NILSON REPORT (2020): "Card fraud losses reach $28.65 billion." `https://nilsonreport.com/mention/1313/1link/`.

VENABLES, W. & B. RIPLEY (2002): *Modern applied statistics with S.* New York: Springer, 4th edition.

ZAKARYAZAD, A. & E. DUMAN (2016): "A profit-driven artificial neural network (ann) with applications to fraud detection and direct marketing." *Neurocomputing* **175**: pp. 121–131.