Universiteit van Amsterdam

Masters Thesis

Cryptocurrency Fraud Detection: Supervised Machine Learning for Illicit Ethereum Transactions

Author:
Thomas Dornigg

Supervisor:
Torsten JOCHEM

A thesis submitted in partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE IN (QUANTITATIVE) FINANCE

Academic Semester 2021/22

July 2022

Statement of Originality

This document is written by Student *Thomas Dornigg* who declares to take full responsibility for the contents of this document. I declare that the text and the work presented in this document are original and that no sources other than those mentioned in the text and its references have been used in creating it. The Faculty of Economics and Business is responsible solely for the supervision of completion of the work, not for the contents.

Universiteit van Amsterdam
Faculty of Economics and Business

Citation of this thesis:
Dornigg, Thomas. Cryptocurrency Fraud Detection: Supervised Machine Learning for Illicit Ethereum Transactions. Master's thesis. Universiteit van Amsterdam, Faculty of Economics and Business, 2022.

Contents

Abstract viii

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Distributed Ledger Technology (DLT) . . . 2
- 1.3 Blockchain Technology . . . 4
- 1.4 Purpose . . . 5
- 1.5 Set-up and scope of this thesis . . . 5

2 Literature Review 6
- 2.1 Definition of fraud . . . 6
- 2.2 Different types of fraud . . . 7
- 2.3 Development of fraud methods . . . 8
  - 2.3.1 Rule-based systems in fraud detection . . . 8
  - 2.3.2 Machine learning based fraud detection . . . 9

3 Fraud Modeling Techniques 10
- 3.1 Introduction to Machine Learning . . . 10
- 3.2 Characteristics of Binary Classification Problems . . . 11
- 3.3 Benchmark Classifier: Logistic Regression . . . 12
- 3.4 Advanced Modeling Techniques . . . 14
  - 3.4.1 Tree models . . . 15
    - 3.4.1.1 Decision Tree . . . 15
    - 3.4.1.2 Random Forest . . . 17
  - 3.4.2 Boosting . . . 18
    - 3.4.2.1 XGBoost . . . 19
    - 3.4.2.2 AdaBoost . . . 21
  - 3.4.3 Support Vector Machines . . . 22

4 Data Organization 24
- 4.1 Dataset characteristics . . . 24
- 4.2 Data preprocessing . . . 25
  - 4.2.1 Outlier & missing value handling . . . 25
  - 4.2.2 Correlation analysis . . . 25
  - 4.2.3 Pipeline . . . 27
- 4.3 Feature selection . . . 29
  - 4.3.1 Variable selection with RFE . . . 30
  - 4.3.2 Variable selection with PCA . . . 31

4.4 Handling of imbalanced data with SMOTE 35

5 Evaluation Metrics and Feature Interpretability **36**
5.1 Model Performance Assessment 36
5.1.1 Confusion Matrix 37
5.1.2 Binary Classification Measures 38
5.1.3 Receiver Operating Characteristic (ROC) 38
5.1.4 Area Under the Curve (AUC) 38
5.2 Test, training and validation sets 39
5.3 K-fold cross-validation 40
5.4 Feature Importance with SHAP 41

6 Model Results **43**
6.1 Model performance on training/test-set 43
6.2 Model performance on holdout-set 45
6.3 Model interpretation with SHAP 46
6.4 Comparison of results of the suggested approach 48

7 Potential Impact of Fraud Models and Regulation in the U.S. **49**
7.1 Cryptocurrency market manipulation 49
7.2 Cryptocurrency regulation: Enforcement and policy responses in the U.S. with regards to fraud 51

8 Discussion **54**
8.1 Research question 54
8.2 Limitations and future work 55

Bibliography **72**

Appendix **72**
A Figures 72
B Tables 75

List of Figures

1.1 Distributed Ledger Technology 3

1.2 Illustration of a blockchain 4

3.1 Types of Machine Learning Techniques 11

3.2 Sigmoid Function 13

3.3 Decision Tree (DT) structure & boundary 16

3.4 Impurity measures 17

3.5 Bagging framework 18

3.6 Boosting framework 19

3.7 Concept of Support Vector Machines 22

4.1 Fraud detection pipeline 27

4.2 Pre-processing pipeline 28

4.3 RFECV for RandomForestClassifier 30

4.4 Variance explained in % by each principal component 33

4.5 Cumulative explained variance ratio 34

4.6 SMOTE algorithm 35

5.1 Boosting framework 37

5.2 Test, training and validation set 39

5.3 K-fold cross validation framework 40

6.1 ROC curves for PCA and RFECV selected feature models 44
6.2 ROC curves for PCA and RFECV selected feature models 45
6.3 SHAP summary (beeswarm) plot 46
6.4 SHAP (global) feature importance plot 47

A.1 Distribution plot for *avg_value_received* 72
A.2 Distribution plot for *max_value_sent* 72
A.3 Correlation matrix (heatmap) 73
A.4 Feature Selection Techniques 73
A.5 ROC-curve 74
A.6 Bitcoin price history 74

List of Tables

6.1 Model-performance summary (PCA feature selected) 44

6.2 Model-performance summary (RFECV feature selected) 44

6.3 Model-performance summary (PCA feature selected) 45

6.4 Model-performance summary (RFECV feature selected) 45

B1 Variable description of the dataset (1/3) 75

B2 Variable description of the dataset (2/3) 76

B3 Variable description of the dataset (3/3) 77

B4 Descriptive statistic of the dataset (1/2) 78

B5 Descriptive statistic of the dataset (2/2) 78

Abstract

By definition, fraud is described as the intentional deception of a person or an organization, to gain personally or financially. In particular, the more complex the environment, the more difficult it becomes to detect fraudulent activities. Traditionally, this is performed by employing rule-based, behavioral, or manual techniques. With the increase in popularity of cryptocurrencies, Bitcoin, Ethereum, and other tokens are becoming more and more prevalent as an alternative source of payment, which however has also led to a spike in market manipulation. Thus, more advanced statistical models need to be utilized to spot bogus transactions. This paper investigates the possibilities of using supervised machine learning models to detect frauds in the Ethereum blockchain, while at the same time, also looking into the model's feature explainability. Results indicate that Random Forest and XGBoost are particularly capable of detecting frauds; outperforming traditional methods on a broad scale, measured by several key metrics. In addition, this thesis also shows potential implications of the developed model on security-related aspects of cryptocurrencies, in particular, that authorities and other related parties need to take to tackle future hurdles.

Keywords: fraud detection, Ethereum, blockchain, supervised machine learning, SHAP, binary classification, crypto-legislation, risk reduction

Abbreviations

AdaBoost	Adaptive Boosting
AI	Artificial Intelligence
ANN	Artificial Neural Network
AUC	Area under the ROC Curve
BTC	Bitcoin
CART	Classification And Regression Tree
DLT	Distributed Ledger Technology
DT	Decision Tree
ETH	Ethereum
FN	False Negative
FP	False Positive
ICO	Initial Coin Offering
LR	Logistic Regression
ML	Machine Learning
PCA	Principal Component Analysis
RF	Random Forest
RFECV	Recursive Feature Elimination with Cross-Validation
ROC	Receiver Operating Characteristic
SEC	U.S. Securities and Exchange Commission
SHAP	SHapley Additive exPlanations
SMOTE	Synthetic Minority Over-sampling TEchnique
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
XGB	Extreme Gradient Boosting

Chapter 1

Introduction

In this chapter, the motivation of this thesis is provided. Additionally, topics such technological background information, research question, set-up, and scope are briefly addressed.

1.1 Motivation

Since 2008, the popularity of cryptocurrencies has risen, with more than 10,000 coins listed on CoinMarketCap - one of the most comprehensive public cryptocurrency data provider - and a combined market-value of about US $2 trillion by the end of February 2022. Specifically with the introduction of Bitcoin in 2009 by a computer scientist (or, possibly, a group of programmers) under the pseudonym *Satoshi Nakamoto*, the first decentralized digital currency for peer-to-peer transactions (P2P) became widely popular around the globe. While in the beginning many labelled it as play money established by geeks and tech enthusiasts, Bitcoin has been providing the basic blueprint or framework for thousands of other coins and has inspired the creation of an innumerable number of blockchain-based solutions to real-world problems over the last decade, ranging from the financial sector, to healthcare, manufacturing and many other industries. Thus, any proactive or forward-thinking entrepreneur must acknowledge the potential benefits coming from the blockchain market. Particularly, the largest advantageousness of the underlying blockchain technology is due to its very own *trustless* network structure (Filippi, 2020). As cryptocurrency transactions are purely digital, they only exist on a common database, once funds are transferred, also called *distributed public ledger*. This decentralized structure not only guarantees enhanced security and transparency, but also instant traceability and cost-savings on a broad scale for the general public. The transparency characteristic of blockchain stems from

the fact that its public ledger is open to view, adding an unprecedented layer of accountability to financial transactions (Koksal, 2019).

The recent innovations in the last couple of years, specifically the seamless integration of smart gadgets and other devices, have led to a vigorous increase in transactions, which also comes with a risk, in particular, as cryptocurrency-based transfers are more prone to fraud than standard bank transfers. As Troy and Pratt (2021) from CNBC reports, a record amount of approx. US $14 billion around the world in cryptocurrencies was taken by scammers in 2021, constituting an increase in crypto-related crime by about 79% compared to 2020. These losses demonstrate the need for protection against potential scammers and fraudsters for all parties involved by implementing a fully automated detection-based system, that can prevent similar crimes in the future, which can be applied by both crypto providers and financial authorities.

Thus, the motivation of this thesis is to closely analyze and demonstrate a machine learning-based solution of a fraud detection system, that is capable of tracking down potential frauds within the Ethereum blockchain. The reason why Ethereum was chosen over the popular coin Bitcoin is twofold: First, while the latter is merely used as a mean of financial transaction, Ethereum focuses also on decentralized applications (dApps), that allows users to create smart contracts or protocols, thus expanding the horizon of potential frauds. Secondly, previously conducted work in this field put a great emphasis on the analysis of Bitcoin, given its domination in terms of total asset value, while Ethereum and other coins did not receive as much attention.

1.2 Distributed Ledger Technology (DLT)

As Natarajan et al. (2017) explain, Distributed Ledger Technology (DLT) refers to a digital system for recording and sharing data across various data stores (ledgers), which each have the same data records and are collectively maintained and controlled by a distributed network of computer servers, which are called nodes. Unlike traditional databases (centralized ledger), where the ledger is stored by a central authority, DLTs have no central data storage nor administration responsibility, thus immutable once the information is stored. The most widely recognized form of Distributed Ledger Technology is the underlying framework for most cryptocurrencies and other applications, the blockchain setup (Kuhn et al., 2019). A schematic overview of a centralized versus distributed ledger technology is shown in Figure 1.1.

In a distributed ledger system, each node independently builds and records the updates in the ledger, thereby creating consensus on its veracity (Voulgari, 2019).

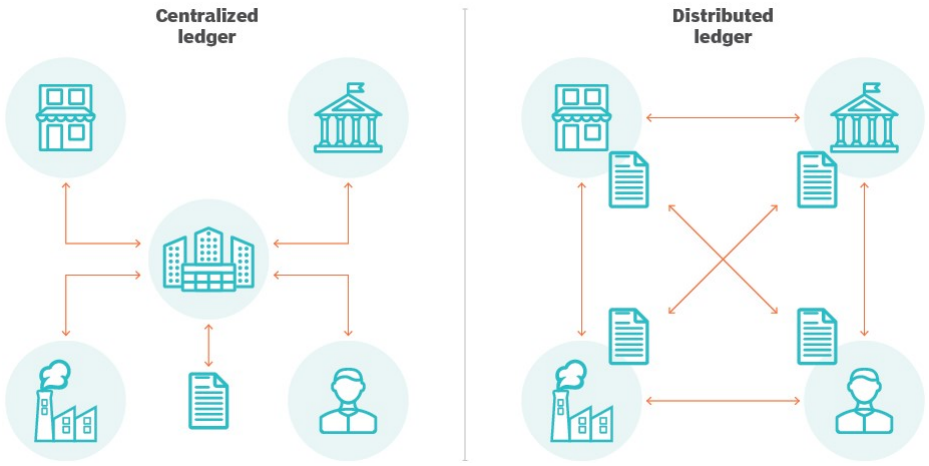


FIGURE 1.1: Distributed Ledger Technology [Troy and Pratt (2021)]

However, consensus is only reached, if most of the network accepts the update (voting nodes). Once this process is completed, consensus algorithms are used to determine if book entries were validly made (Kulnigg, 2020). In terms of how consensus is reached in the DLT/blockchain world, different proven mechanisms exist, most commonly are the following:

- **Proof-of-Work (PoW)**

  Both Ethereum and Bitcoin are currently using Proof-of-Work (PoW) consensus protocol. As highlighted in the official Ethereum documentation, PoW is done by miners, who are competing to create new blocks full of processed transactions, which is achieved by investing a certain amount of computational effort to solve a complex arithmetic problem, thus using energy and money. Hence, individuals who participate and win the mining or block generation competition, are rewarded with coins. Therefore, mining can be referred to as the act of adding valid blocks to the blockchain (Boucher et al., 2017).

- **Proof-of-Stake (PoS)**

  Proof-of-Stake is based on the idea that whoever owns the most tokens within the network, i.e. the largest proportion of total token outstanding, has also an interest in keeping the blockchain running and maintained (Kulnigg, 2020). As a result, instead of investing a vast amount of resources into solving complex puzzles, one simply needs to stake up its ETH in the network, which constitutes a deterministic way of choosing the next creator of a block.

1.3 Blockchain Technology

Given the confusion among readers between the terms Distributed Ledger Technology and blockchain, the following section introduces the main differences between the two concepts. An elaborate explanation is given by Holland (2021), who describes it as follows: *while DLT is a decentralized database managed by multiple participants, across various nodes, blockchain is merely a type of DLT where transactions are recorded with an immutable cryptographic signature called a hash.* By grouping or chaining the transactions into blocks, each new block includes the hash of the previous one, hence the reason why distributed ledgers are often called blockchains (Corda, 2021). Once the blocks are linked together in a chain, the data saved on an individual block cannot be altered, without having to change every consecutive block coming after, thus making the blockchain so trustworthy.

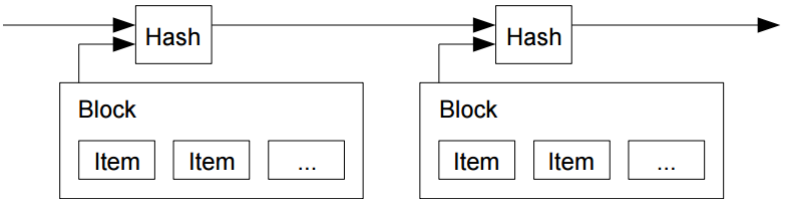


FIGURE 1.2: Illustration of a blockchain [Musienko (2021)]

Interestingly, blockchain technology is not a completely new concept. However, with the publication of the white paper in 2008 by Satoshi Nakamoto, it gained widespread public attention. The novel approach which was introduced with this release was the combination of multiple technologies, such as peer-to-peer network, cryptography, digital signatures, nodes, hashing, consensus protocols, and mining, to create the world's first-ever digital currency (Voulgari, 2019). At its most basic level, blockchain incorporates multiple properties, including without limitation: privacy, time-stamping, decentralization, immutability, and transparency. While privacy and transparency might contradict each other, they go hand-in-hand. This is due to the components of cryptography, as every user on a blockchain network has a set of two keys, a private and a public key, which are both *hashed*. The private key is used to make digital signatures for transactions, and the public key is known to the entire network (Houben and Snyers, 2018). As a consequence, the physical identity of a user is secure, while the digital identity is monitored and recorded, so that transparency is secure as well (Voulgari, 2019). Both keys of a user are stored in a digital wallet, which can be saved either online, referred to as *hot storage*, and/or offline, also called *cold storage* (ECB, 2015).

1.4 Purpose

The main objective of this thesis is to investigate if (supervised) machine learning classifier(s) are able to predict potential frauds in the Ethereum blockchain, which can serve as an input for law enforcement agencies to detect potential scams in the future. As a result, a contribution to the common fraud detection literature might include a new understanding of how frauds can be classified with new technologies, in a way that has not been thematized before by traditional quantitative methods. As a consequence, the thesis addresses the following research question:

RQ: By utilizing state-of-the-art machine learning modeling techniques, how well do they perform in predicting fraudulent transactions in Ethereum, respectively, which attributes have the largest impact on the final outcome?

1.5 Set-up and scope of this thesis

The aim of this thesis is to provide law enforcement agencies and other crypto-related parties with a comprehensive example of a potential fraud detection tool that is capable of detecting illicit Ethereum transactions. This is achieved by training binary (supervised) machine learning classifiers and evaluating their predictive performance, as well as performing state-of-the-art feature explainability to shed light on the decision-making process.

The rest of this thesis is organized as follows: section 2 provides the reader with a brief literature review on previously conducted work in the field of (cryptocurrency) fraud detection. Next, in section 3, the traditional or benchmark fraud model and more advanced machine learning classifiers are presented. Section 4 deals with data-related aspects, ranging from descriptive statistics, Exploratory Data Analysis (EDA), and the data processing pipeline, which handles not only outlier-detection and missing value handling, but also feature scaling and variable encoding. Additionally, feature reduction techniques such as Principal Component Analysis and Recursive Feature Elimination are discussed. While section 5 highlights the main evaluation techniques which are used to determine the best algorithms, section 6 briefly presents the outcomes of the trained models. The second to last chapter, section 7, summarizes potential implications of the discussed fraud detection model and also reviews already implemented laws & regulations with regards to cryptocurrency and blockchain-related aspects. Finally, in the last chapter - section 8 - the reader is presented with a conclusion and potential limitations of the work.

Chapter 2

Literature Review

This section provides necessary background information regarding a general definition of fraud models as well as previously conducted work in the same area.

2.1 Definition of fraud

Financial fraud constitutes an issue with far-reaching consequences, that affects not only many business areas but also the common daily life of ordinary people. Depending on the field of study, ranging from social sciences, economics, and psychology to law, various definitions of financial fraud have been proposed. For instance, in the social science literature, Ramamoorti and Olsen (2007) define financial fraud as *"a human endeavor, involving deception, purposeful intent, the intensity of desire, risk of apprehension, violation of trust, rationalization".*

With regards to a common legal definition, fraud is a rather complex and elusive concept, given that it varies from country to country, thereby no universal and coherent definition exists. In England and Wales, fraud has only been properly defined by law in 2006, with the introduction of the Fraud Act (Birch et al., 2008). In the discipline of accounting, a more elaborate interpretation is given by Albrecht et al. (2015) who say that *"fraud is a deception that includes the following elements: a representation about a material point which is false, and intentionally or recklessly so; which is believed and acted upon by the victim, to the victim's damage"* (Akers and Bellovary, 2006). Moreover, in the context of economics, specifically financial market activities, fraud can best be described as the unlawful falsification or manipulation of financial information, given that it acts as the key pillar for market transactions (Reurink, 2016).

2.2 Different types of fraud

Because financial fraud can occur in many different scenarios where misleading statements are made, as well as a variety of data mining methods that are actively being researched to find appropriate measures to tackle those issues, common financial fraud categories include but are not limited to:

- **Ponzi schemes:** According to the U.S. Securities and Exchange Commission, a Ponzi scheme, which got its name from the notorious 1920s swindler Charles Ponzi, is an investment fraud that pays existing investors with funds collected from new investors, by promising them to pay relatively high rates of returns for fixed-term investments.
- **Pump and Dump schemes:** A pump and dump scheme is a manipulative type of securities fraud, where fraudsters attempt to spread false, misleading, or greatly exaggerated information to artificially inflate ("pump") the price of a certain security. The impostor profits from the inflated price by quickly selling ("dump") the security at a high price.
- **Phishing:** In the most classical form, phishing is a type of social engineering attack that attempts to steal a person's private information, e.g. bank data, credit card numbers, or login credentials, by targeting their email, telephone or text messages.
- **Identity fraud**: As highlighted by the OAIC, identity fraud is the usage of a person's (financial) information without their consent, with the goal of committing fraud, such as making unauthorized transactions, purchases or to conduct illegal activities.
- **Transaction fraud:** Paygilant, a fintech specialized in payment fraud prevention, defines transaction fraud as the usage of stolen payment card or other kind of sensitive information that can be used to generate an unauthorized transaction.

In the context of this specific paper, where transactions were flagged by the Ethereum community for illicit behavior for several cases, which Farrugia et al. (2020) investigated by using a similar subset of the Ethereum account dataset (see section 4.1 for more details), some of the frauds include (i) trying to imitate other contract addresses providing tokens, (ii) scam lotteries, (iii) fake initial coin offerings (ICO), (iv) imitating other users, (v) Ponzi schemes, (vi) phishing and (vii) mirroring websites.

2.3 Development of fraud methods

According to Sanchez et al. (2021), various approaches exist for detecting frauds, most of them are focused nowadays on statistical and parametric analyses based on data mining techniques, but also traditional behavioral screening; none of them however can capture the problem of timely fraud detection. Due to innovation and advancements in the telecommunication technology in recent years, foremost in mobile computing and modern payment methods, a sharp increase in financial fraud was observable (Yeh and Lien, 2009). However, as scammers and fraudsters are continually improving their game, there is a demand for reviewing the latest research conducted in the field of fraud management. As a result, the literature on traditionally used fraud detection techniques and more advanced statistical models are investigated in this part of the paper.

2.3.1 Rule-based systems in fraud detection

Many fraud detection applications rely on their core engine on rule-based methodologies for generating alerts about suspicious behaviors, so-called Business Rule Management Systems (Rosset et al., 1999). As the name implies, those detection approaches rely on hard-coded rules that are set to flag transactions if they meet certain criteria (Karczewski, 2022). Such detection systems commonly include the following rules:

- **Location:** The system triggers a warning if the transaction is outside the usual location of the sender, or if multiple transactions are made in a short period of time from different, implausible locations.
- **Frequency:** An alert is issued if a user with almost no-existing transaction history suddenly appears in numerous, unusual wires with no explainable connection.
- **Sender/Receiver:** In case of unusual high amount of payments, coming from multiple newly created accounts, or several unusual transactions coming from the same IP address.

For instance, Böhmer and Rinderle-Ma (2019) analyzed anomalies in-process runtime logs by utilizing association rule mining. Unlike machine learning, where model explainability is notoriously difficult, rule-based learning benefits from its simplicity (Collery, 2021). Historically, this approach was used in many applications, for instance telecommunication, investigated by Rosset et al. (1999), credit-risk (Stefanowski and Wilk, 2001), and transportation (Xiao et al., 2019). However, as Lipton (2018) also mentions, up to a certain dimensional stage, the approach

may become less interpretable than simple ML-based models due to its limited capacity for human cognition. More recently, Kraiem (2021) applied rule-based methods for multiple anomaly detection; Sushil et al. (2018) proposed if-then-else rules to explain predictions of supervised learning models, and Nye et al. (2020) presented a neuro-symbolic model which learns entire rule systems from a small set of examples.

2.3.2 Machine learning based fraud detection

Unlike rule-based learning, which relies on human intervention to check for updates and quality checks, machine-learning-based fraud detection models operate in a fully automatic manner, cutting down manual interaction significantly since it only occurs at certain stages of the process, making the overall search for frauds more efficient (Fraud.net, 2021). In essence, as Rosset et al. (1999) highlight, automatic fraud analysis aims at using past examples of fraudulent and legitimate usage to find new patterns and rules to help distinguish between the two.

Taking into account the shortcomings of rule-based methods, a sheer amount of machine learning-based methods have been developed in the last couple of years. The general workflow, as well as an in-detail description of the functionality of the most commonly utilized models, is shown in section 3 and 4. For instance, logistic regression (LR) has shown promising results by Artís et al. (2002) to detect fraudulent insurance claims based on the Spanish market. One further linear classifier, Support Vector Machine (SVM), was successfully tested by Pediredla et al. (2011) on Chinese firms to detect fraudulent financial statements. Additionally, as Zhu et al. (2021) highlight, tree-based models, such as Decision Tree (DT) and its more advanced derivatives like Random Forest (RF), XGBoost (XGB), or LightGBM (LGBM) show superior performance results compared to traditional classifiers, as analyzed by Caruana and Niculescu-Mizil (2006), Caruana et al. (2008), or Khoshgoftaar et al. (2007).

With regards to cryptocurrency fraud detection, Monamo et al. (2016) proposed using trimmed k-means and kd-trees for the detection of bogus Bitcoin transactions; similarly, Pham and Lee (2016) analyzed suspicious or anomalous behavior in the Bitcoin network with unsupervised learning methods, and Jung et al. (2019) study the possibilities of data mining techniques to detect Ponzi schemes in the Ethereum blockchain network. Also, more importantly, Ibrahim et al. (2021) investigated illicit accounts on the Ethereum blockchain by using three different machine learning algorithms, which are DT, RF, and K-nearest neighbors (KNN), applied on a similar dataset as utilized in this paper.

Chapter 3

Fraud Modeling Techniques

In the modeling technique section, relevant background information behind advanced fraud detection models and their potential drawbacks are explained.

3.1 Introduction to Machine Learning

In general, machine learning (ML) is often categorized as a subfield of artificial intelligence (AI) or a field of computer science, that explores the construction and the study of statistical models that can automatically learn from and make predictions on data (Kohavi and Provost, 1998). According to Wehle (2017), the underlying idea has its origins dating back to Alan Turing's seminal 1950 paper *Computing Machinery and Intelligence*, which covered a section on "Learning Machine" that was able to fool a human into believing that it is real.

At its most basic level, machine learning modeling can be grouped into three types: supervised learning, unsupervised learning, and semi-supervised learning. Furthermore, Reinforcement Learning (RL) is another type of machine learning model, that enables an agent through a trial and error approach to learning in an interactive environment. A comprehensive overview of the different types of machine learning is shown in Figure 3.1. While ***supervised learning*** is most useful when the underlying dataset is labeled, ***unsupervised learning*** is utilized in the context of clustering, i.e. grouping a set of data points in a way such that the objects belong to a group that shares similar characteristics. The technique of ***semi-supervised learning*** is a medium between supervised and unsupervised learning, given that it uses both labeled and unlabeled data for training.

3.2 Characteristics of Binary Classification Problems

Due to the nature of the data and the initial problem definition (see section 4 for more details regarding the dataset), special attention will be stressed on supervised machine learning modeling. To be more precise, supervised learning can be further grouped into **regression** and **classification** problems, where both techniques share the same goal of predicting values of the dependent class from a set of attributes. The only difference between the two tasks is the nature of the dependent variable, which is numerical for regression and categorical for classification problems.

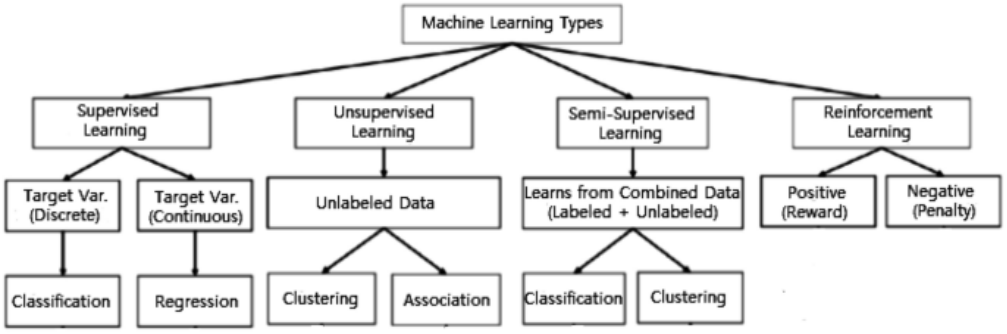


FIGURE 3.1: Types of Machine Learning Techniques [Sarker (2021)]

As the goal of this project is to detect fraudulent from non-fraudulent transactions, this clearly falls into the category of classification. As Learned-Miller (2014) highlights, the training data in any supervised machine learning set-up consists of a **training sample** with m ordered pairs $S = (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m) \in (\mathcal{X} \times \{\pm 1\})^m$, where each $x_i \in \mathcal{X}$ is a data point in $\mathbf{X}$ (in the context of fraud modeling, the vector space might describe multiple features of Ethereum transactions), and $y_i \in \{\pm 1\}$ represents a binary label vector corresponding of two classes, where x_i belongs to either of the two labels (such as "frauds" or "non-frauds"). Most often, the pair (x_i, y_i) is referred to as **(labeled) training example**. According to Agarwal (2019), the goal of supervised learning is to train a classifier from S, call it $h_S : \mathcal{X} \to \{\pm 1\}$, which can take a new input $x \in \mathcal{X}$, that can correctly predict its class label via $\widehat{y} = h_S(x)$. To put it differently, the aim in a binary classification task is to search for a decision function whose level sets may be used to separate the $\mathbf{X}$ of differing class labels $Y_i \in \{0, 1\}$ (Jordan, 2004).

For the sake of completeness, the way how a simple linear binary classifier works, is as follows: based on the estimated decision function (see Equation 3.1) of the inputs, the model determines

whether or not the value is larger than some threshold r (Grosse, 2019a).

$$w_1 x_1 + \cdots + w_D x_D + b = \mathbf{w}^T \mathbf{x} + b \tag{3.1}$$

where **w** is a **weight vector** and b is a scalar-valued bias. As a result, the prediction $\widehat{y}$ can be calculated as follows:

$$\begin{aligned} z &= \mathbf{w}^T \mathbf{x} + b \\ \widehat{y} &= \begin{cases} 1 & \text{if } z \geq r \\ 0 & \text{if } z < r \end{cases} \end{aligned} \tag{3.2}$$

The chosen classification models in this thesis are the following: First, as one of the most widely used models in the industry, Logistic Regression was picked as a threshold comparison method. Secondly, for more advanced statistical models, Decision Tree, Random Forest, XGBoost, AdaBoost, and Support Vector Machine were selected. The theory for each of these classification methods will be addressed in later subsections of this paper.

3.3 Benchmark Classifier: Logistic Regression

As mentioned in section 2, numerous studies have shown promising results in the usage of logistic regression throughout many years, thus the same method was chosen as a benchmark classifier. The aim of binary logistic regression is to train a model that is capable of performing a binary decision about an observation. Given the sample space definition from the previous section, one can re-write **X** in a vector form as $x = [x_{i1}, x_{i2}, \ldots, x_{ip}]^T$ where p is the total amount of features in the dataset.

As Lindholm et al. (2021) further highlights, learning a binary model means to train a classifier of $p(y = 1|x)$ and $p(y = 0|x)$. Due to the laws of probability, it follows that $p(y = 0|x) + p(y = 1|x) = 1$, as a result, it is enough to learn a model for only one of the class probabilities, for example $p(y = 1|x)$, from which $p(y = 0|x)$ will follow. By learning from the training set, a vector of **weights** and a **bias term**, logistic regression can solve this task, where each weight w_i is a real number that is associated with one of the input features x_i (Jurafsky and Martin, 2008).

In logistic regression, the bias term, which is another real number that's added to the weighted inputs, is also called intercept (Pedregosa et al., 2011). In order to make a decision on a new

test input (after the weights are learned from the train-set), the classifier multiplies each x_i by its weight w_i, sums up the weighted features, and adds the bias term b, to make a decision, which is expressed in the number z (Jurafsky and Martin, 2008):

$$z = \left(\sum_{i=1}^{n} w_i x_i \right) + b \tag{3.3}$$

A more convenient representation of Equation 3.3 can be achieved by utilizing dot product notation, which is shown in Equation 3.4:

$$z = \mathbf{w} \cdot \mathbf{x} + b \tag{3.4}$$

However, as seen from Equation 3.4, nothing forces z to lie between 0 and 1. In particular, the output of the given equation can take values ranging from $-\infty$ to ∞. To overcome this issue, z is passed through the **sigmoid function** $\phi(z)$, also called logistic function, that is able to squeeze the output of a linear equation between the interval [0,1], as shown mathematically in Equation 3.5, and graphically in Figure 3.2, exactly what is required to obtain a probability:

$$\phi(z) = \frac{1}{1 + e^{-z}} = \frac{1}{1 + \exp(-z)} \tag{3.5}$$

An important feature that can be observed from the S-shaped sigmoid function is that it increases monotonically, with asymptotes at 0 and 1 (Grosse, 2019b).

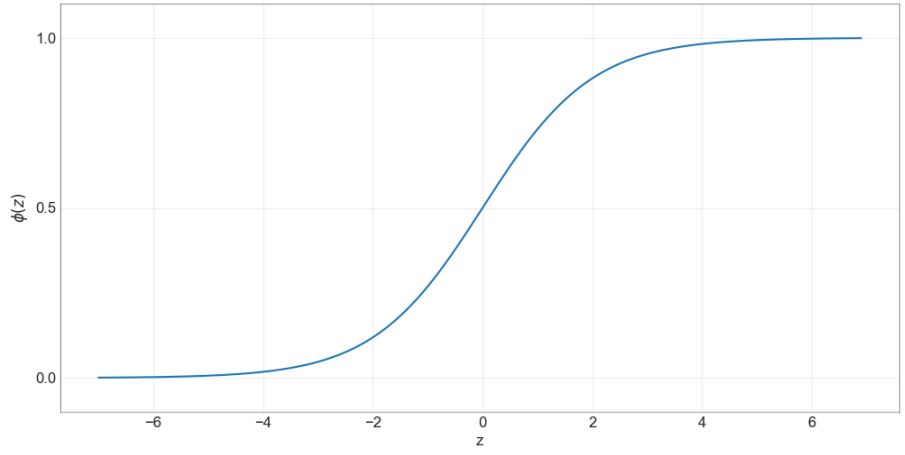


FIGURE 3.2: Sigmoid Function $\phi(z)$

In a final step, to ensure that the two cases, $p(y = 1|x)$ and $p(y = 0|x)$ sum to 1, the following step needs to be applied (Jurafsky and Martin, 2008):

$$\begin{aligned} P(y = 1) &= \phi(\mathbf{w} \cdot \mathbf{x} + b) \\ &= \frac{1}{1 + \exp(-(\mathbf{w} \cdot \mathbf{x} + b))} \\ \\ P(y = 0) &= 1 - \phi(\mathbf{w} \cdot \mathbf{x} + b) \\ &= 1 - \frac{1}{1 + \exp(-(\mathbf{w} \cdot \mathbf{x} + b))} \\ &= \frac{\exp(-(\mathbf{w} \cdot \mathbf{x} + b))}{1 + \exp(-(\mathbf{w} \cdot \mathbf{x} + b))} \end{aligned} \tag{3.6}$$

which has the property

$$1 - \phi(x) = \phi(-x) \tag{3.7}$$

Taking into account the **decision boundary function** as defined in Equation 3.2, for the logistic regression classifier to make a choice which class a new instance x should take, a threshold z of 0.5 was chosen.

3.4 Advanced Modeling Techniques

Even though logistic regression is one of the simplest (linear) machine learning algorithms, providing reliable and robust estimates, it is notoriously lagging when it comes to high-dimensional datasets, which may lead to model performance overfitting. Furthermore, given that most real-world applications are not linear by nature, non-linear problems cannot be solved with a logistic classifier since it only incorporates a linear decision surface.

To tackle these problems, advanced statistical models could provide a potential, more scientific solution, in helping to achieve higher predictive accuracy of the outcomes. Thus, in the following, a brief (mathematical) introduction is given to the utilized supervised classification models that were used in comparison against logistic regression. In particular, this sub-chapter deals with tree-based models (Decision Tree and Random Forest), Boosting models (AdaBoost and XGBoost), and Support Vector Machine.

3.4.1 Tree models

Tree-based models are part of non-parametric algorithms that can deal both with classification and regressions tasks, which were first introduced from a statistical point of view by Breiman et al. (1984). While tree models present high flexibility when it comes to capturing complex non-linear relationships, they however suffer from memorizing the error (or noise) incorporated in the dataset (Sheehy, 2018).

3.4.1.1 Decision Tree

A decision tree is a type of supervised machine learning algorithm that can be used in both regression and classification problems, but mostly it is preferred in classification tasks. It is a tree-structured classifier, hence the name, where internal nodes represent the features of the dataset, branches represent the decision rules and each leaf node represents the outcome (Arain, 2021). An example of a theoretical multi-way split feature space and its corresponding tree is depicted in Figure 3.3. In general, there are many methodologies for constructing decision trees, including CART, ID3, C4.5, CHAID, and QUEST, but the most well-known technique is the classification and regression tree (CART) algorithm, first proposed by Breiman et al. (1984). The success of CART can largely be attributed to how splits are performed.

To define a good split, the CART ecosystem is utilizing a function called impurity. The best partition is defined as one that has more purity after the splitting (i.e., all records in a given partition belong to the same class). There are various impurity measurements used, the most common are: Classification Error, Gini Index and Entropy. To illustrate the splitting process, Figure 3.3 will be investigated in greater detail.

A basic decision tree partitions the training data into homogeneous subgroups (i.e., groups with similar response values) and then fits a simple constant in each subgroup. The subgroups (nodes) are formed recursively using binary partitions formed by asking simple yes-or-no questions about each feature. This is done a number of times until a suitable stopping criteria is satisfied (Boehmke and Greenwell, 2020). During the building process of the Decision Tree model, not all potential input variables will be used and in some cases a specific input variable may be used multiple times at different levels of the tree (Song and Lu, 2015). The endpoint, i.e. the predicted response of each branch A, B, C, D and E are called *leaf* or *terminal nodes* and the internal splits, $x_1 > \phi_1$, $x_2 \leq \phi_2$, $x_2 > \phi_3$ and $x_1 \leq \phi_4$ are known as the *internal nodes* (Lindholm et al., 2021).

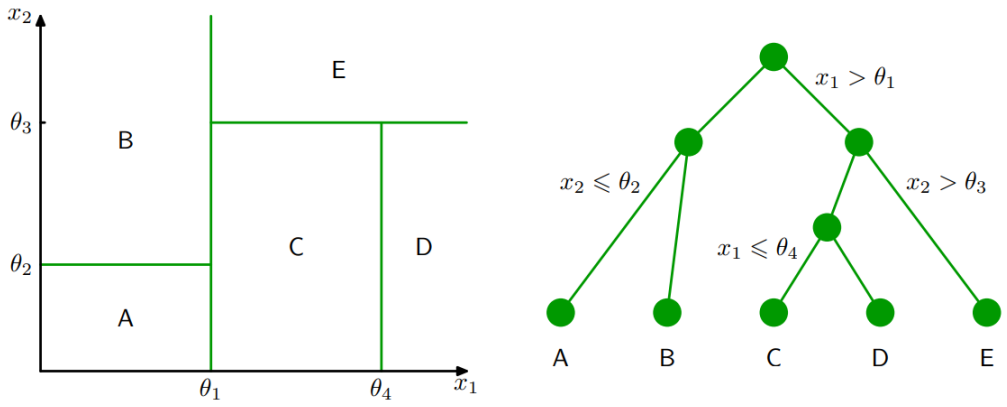


FIGURE 3.3: The decision boundary results in rectangular regions that enclose the observations. The class with the highest proportion in each region is the predicted value (left). Generalized Decision Tree structure (right) [Schaar (2017)]

Growing a binary classification Decision Tree is done by using recursive binary splitting. In a binary classification setting, a commonly used criterion for making a split or partition is the so-called *classification error rate*, which can be defined as the fraction of the training observations in that region that do not belong to the most common class (James et al., 2013):

$$E = 1 - \max_k(\hat{p}_{mk}) \tag{3.8}$$

where $\hat{p}_{mk}$ represents the proportion of training observations in the mth region that are from the kth class. However, given that the classification error is sensitive to noise and thus is prone to overfitting, in practice two other measures for impurity are preferred.

Entropy, which is incorporated in the ID3 algorithm, measures the homogeneity of a sample. The entropy is an absolute measure which provides a number between 0 and 1 - indicating if a branch is completely homogeneous (0) or equally divided between the individual labels (1) (Ricaud, 2017).

$$E(x) = \sum_{i=1}^{n} p_i log_2(p_i) \tag{3.9}$$

Arguably, as the most popular impurity measurement in the CART framework, ***Gini Index*** (also called Gini impurity) measures the degree of probability of a particular variable being incorrectly classified when it is chosen randomly (Li, 2021).

$$GINI(x) = 1 - \sum_{i=1}^{n} p_i^2 \tag{3.10}$$

Depending on the main purpose, any of the three mentioned impurity measures might be used for pruning a tree. While the classification error rate is preferable if the prediction accuracy of the final pruned tree is the goal, Gini index or Entropy are typically used to evaluate the quality of a particular split given their sensitivity to node purity (James et al., 2013). Figure 3.4 depicts the three mentioned node impurity measures for a two-class classification problem.

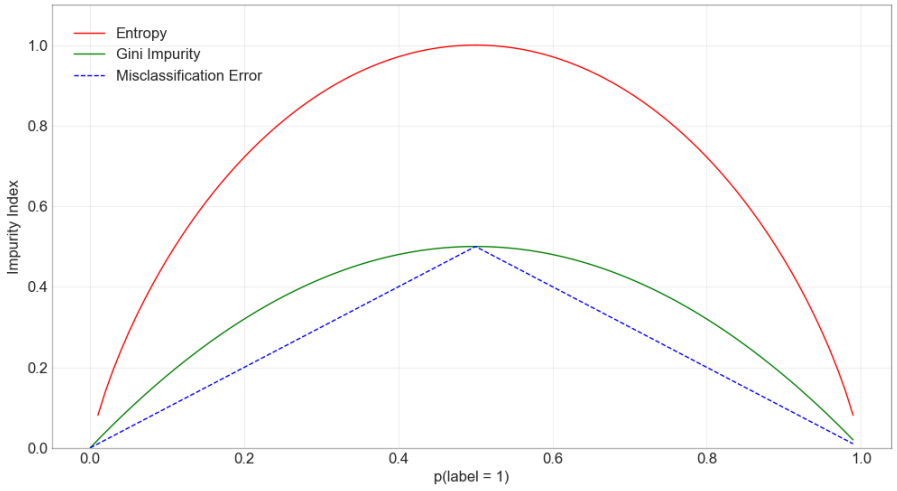


FIGURE 3.4: Impurity measures

3.4.1.2 Random Forest

To explain the basic principle of the Random Forest algorithm, the main concepts of bagging and bootstrapping must be discussed beforehand. The bootstrap method, or abbreviated bootstrapping, first proposed by Efron (1979), is a resampling technique used to estimate statistics on a population by sampling a dataset with replacement (Brownlee, 2021). It replicates the sample variation and allows the calculation of summary statistics such as the mean or standard errors (Teixeira-Pinto, 2020).

As addressed in the previous chapter, a single Decision Tree suffers from high variance and low bias since it does not generalize well on new data. Bootstrap aggregation, or bagging, is a general-purpose procedure for reducing the variance of a statistical learning method, hence it is particularly useful and frequently used in the context of Decision Trees (James et al., 2013). As Breiman (1996) states, bagging is regarded as a fairly straightforward procedure where multiple versions of a prediction model are fitted and then combined (or ensembled) together into an aggregated prediction. From a mathematical point of view, b bootstrap copies of the original training data are created and the base learner is applied to each bootstrap sample (Cao and

Francis, 2021). For classification tasks, the base learner predictions are combined using plurality vote or by averaging the estimated class probabilities together (James et al., 2013).

This can visually be seen in Figure 3.5 and conceptually in Equation 3.11

$$\widehat{f_{bag}} = \widehat{f}_1(X) + \widehat{f}_2(X) + \cdots + \widehat{f}_b(X) \tag{3.11}$$

where X constitutes the the record for which a prediction will be generated, $\widehat{f_{bag}}$ is the bagged prediction and $\widehat{f}_1(X), \widehat{f}_2(X), \ldots, \widehat{f}_b(X)$ are the predictions from the individual base learners.

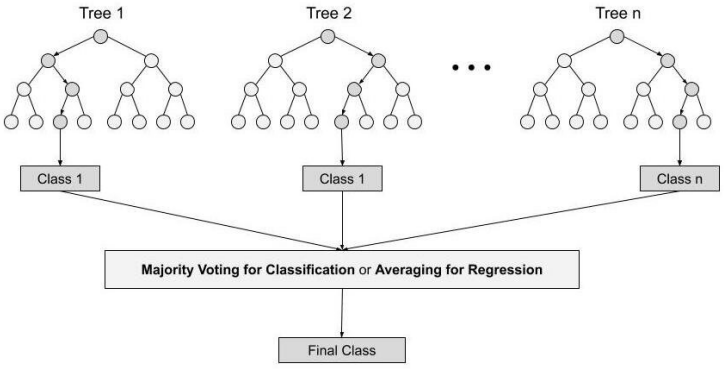


FIGURE 3.5: Bagging framework [Cowan (2020)]

While pruning is a good method of improving the predictive performance of a Decision Tree model, a single tree will generally not produce strong prediction outcomes. Thus, to improve its predictive power, the Random Forest algorithm incorporates not only the methodology of Decision Trees but also the same setup as bagging when building trees on bootstrapped data sets but overcomes the problem of highly correlated trees (Granström and Abrahamsson, 2019). This is achieved, as Breiman (2001) explained, by selecting at random, at each node, a small group of the original input variables p from a subset of samples m_{try}, to split on.

3.4.2 Boosting

In this section, the concept of boosting with a special focus on its most common implementations, XGBooost (short for e**X**treme **G**radient **Boost**ing) and AdaBoost (a sobriquet for **Ada**ptive **Boost**ing) will be briefly addressed. As a general rule, boosting models are part of the ensemble technique family that attempts to build a strong classifier by combining a set of weak learners (wisdom of crowds) to minimize the training error. As Lindholm et al. (2021) further emphasizes, boosting is built on the idea that even a simple (or weak) high-bias classification model often can

capture some of the relationships between the inputs and the output. In other words, a single model (weak learner) may not perform well alone due to bias/variance tradeoff, an aggregation of base learners however can yield a better model performance given the reduction in bias or variance. Breiman (1996) motivated bagging by using decision trees, even though it can be used with any type of model, because this algorithm can be prone to overfitting when it has not been pruned and it can also lend itself to underfitting when it is very small, like a decision stump, which is a decision tree with one level (IBM-Cloud-Education, 2021). A general boosting framework is depicted in Figure 3.6.

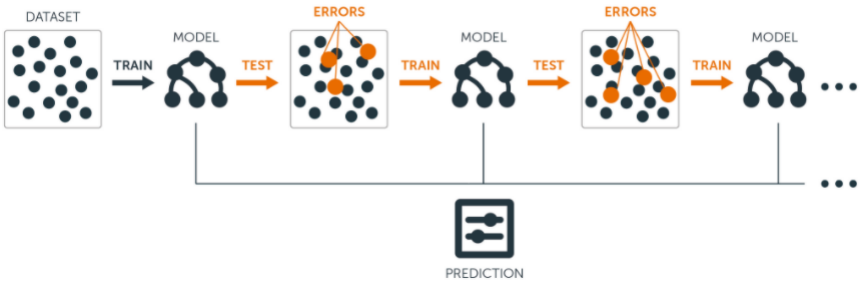


FIGURE 3.6: Boosting framework [Le (2018)]

In essence, boosting handles the bias-variance trade-off in the following way: Unlike fitting a single large Decision Tree to the data, it sequentially boosts its performance by continuing to build new trees, where each new tree in the sequence tries to fix up where the previous one made the largest mistakes (Molnar, 2021).

3.4.2.1 XGBoost

The XGBoost (XGB) model is regarded as one of the most powerful algorithms for training on tabular data, handling both classification and regression problems. Initially, it started as a research project in 2014 but quickly gained widespread popularity among the community, and in 2016, it was officially presented by Tianqi Chen and Carlos Guestrin at the SIGKDD Conference. More importantly, XGBoost is not only an algorithm but also an open-source library, designed as an optimized implementation of the Gradient Boosting framework which focuses on speed, flexibility, and model performance (Martins, 2021). As the creators of the library Chen and Guestrin (2016) illustrate, Extreme Gradient Boosting is approx. 10-times faster and achieves higher accuracy rates in terms of model performance than standard tree-based models. Additionally, it is highly popular among researchers and machine learning practitioners, given

the winnings in multiple *Kaggle* and data-mining competitions (for an overview of former contests, see here). In addition, modifications to the original method exist for dealing with highly imbalanced datasets (Wang et al., 2020).

As Chen and Guestrin (2016) highlight, XGBoost has three important aspects, which are regularized objective function for better generalization, gradient tree boosting for additive training, and shrinkage and column subsampling for preventing overfitting (Can et al., 2021). The following equations (3.12) - (3.16) are summarizing the algorithmic process. Similarly to other Gradient Boosting implementations, XGBoost builds an additive expansion of the objective function by minimizing a loss function, which is used to control the complexity of the trees

$$\mathcal{L}(\Theta) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega\left(f_k\right) \tag{3.12}$$

where the first term in Equation 3.12 is a loss function, intending to measure the difference between the target y_i and the prediction $\hat{y}_i$, and the second one (see Equation 3.13) is a penalty term for controlling model complexity to avoid overfitting (Noh et al., 2021).

$$\Omega\left(f\right) = \gamma T + \frac{1}{2}\|\omega\|^2 \tag{3.13}$$

To minimize the loss function at step t, an iterative approach is applied, given by Equation 3.14.

$$\mathrm{L}^{(\mathrm{t})} = \sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{l}\left(\mathrm{y_i}, \hat{\mathrm{y}}_{\mathrm{i}}^{(\mathrm{t}-1)} + \mathrm{f_t}\left(\mathrm{x_i}\right)\right) + \Omega\left(\mathrm{f_t}\right) \tag{3.14}$$

To speed up the optimization process, second-order Taylor expansion is applied to the objective. After removing the constant terms, a simplified objective function at step t is given in Equation 3.15 (Can et al., 2021).

$$\overline{\mathrm{L}}^{(\mathrm{t})} = \sum_{\mathrm{i}=1}^{\mathrm{n}} \left[\mathrm{g_i f_t}\left(\mathrm{x_i}\right) + \frac{1}{2}\,\mathrm{h_i f_t^2}\left(\mathrm{x_i}\right)\right] + \Omega\left(\mathrm{f_t}\right) \tag{3.15}$$

where;

$$\begin{aligned} \mathrm{g_i} &= \partial_{\hat{\mathrm{y}}^{(\mathrm{t}-1)}} \mathrm{l}\left(\mathrm{y_i}, \hat{\mathrm{y}}^{(\mathrm{t}-1)}\right) \\ \mathrm{h_i} &= \partial^2_{\hat{\mathrm{y}}^{(\mathrm{t}-1)}} \mathrm{l}\left(\mathrm{y_i}, \hat{\mathrm{y}}^{(\mathrm{t}-1)}\right) \end{aligned} \tag{3.16}$$

To conclude, the optimization of the objective function, as seen in Equation 3.15, is converted to a problem of determining the minimum of a quadratic function. In addition, because of the introduction of a regularization term, XGBoost has a better ability to handle overfitting (Liang et al., 2020).

3.4.2.2 AdaBoost

In the same way as XGBoost, AdaBoost is considered as an ensemble algorithm that combines a set of base (weak) learners, which are slightly better than random guessing, into a weighted sum to create a stronger and more accurate classifier. Initially, the original (Discrete) AdaBoost model was published by Freund and Schapire (1996) but many variants of the algorithm soon emerged (an overview see here). Similar to other boosting models, the most common algorithm used with AdaBoost are Decision Trees with one level. Given that these DTs are short and only contain one classification decision, they are also called decision stumps (Brownlee, 2021).

The AdaBoost classifier, also called AdaBoost.M1, works as follows: It starts by fitting a classifier on the dataset and then fits the same version of that classifier on the same dataset where the weights of the misclassified instances are modified so that the next classifier is improved to work on the more challenging instances (Yassin et al., 2020).

Conceptually, as Peng et al. (2018) highlights, this can be expressed in the following way: Given a set of training data S $= \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$, $x_i \in R^m$ and $y \in Y = \{-1, +1\}$, where each x_i belongs to the domain and each y belongs to the label space. In terms of a binary classification problem, the label space is equal to $\{-1, +1\}$. The goal is to search a classification principle among the training data, and then if a new x is given, y from $\{-1, +1\}$ can be assigned to it.

1. Initialize the distribution of weights of observations:

$$\mathrm{D}_1 = (w_{11}, \cdots, w_{1i}, w_{1m}), \;\; w_{1i} = \tfrac{1}{m}, \;\; i = 1, 2, \ldots, m \tag{3.17}$$

2. For $j = 1$ to T

   - Use the weights w_i' to train the weak learner $f^j(x)$
   - Compute the error rate of weak learner:

$$e_j = P\left(f^j\left(x_i\right) \neq y_i\right) = \sum_{i=1}^{m} w_i^j I\left(f^j\left(x_i\right) \neq y_i\right) \tag{3.18}$$

- If $e_j > 0.5$, then stop
- Calculate the coefficient of the weak learner:

$$\alpha_j = \frac{1}{2} \ln \left((1 - e_j) / e_j \right) \tag{3.19}$$

- Update the distribution of weights:

$$w_i^{j+1} = w_i^j \exp \left(-\alpha_j y_i f^j \left(x_i \right) \right), i = 1, 2, \ldots, m \tag{3.20}$$

3. Output the final classifier:

$$H(x) = sign \left(\sum_{j=1}^{T} \alpha_j f^j(x) \right) \tag{3.21}$$

3.4.3 Support Vector Machines

The Support Vector Machine (SVM) model is a supervised learning technique that belongs to the family of generalized linear classifiers, which is able to solve both classification and regression tasks, first introduced in 1992 by Boser, Guyon, and Vapnik in COLT-92 (Grosse, 2019b). As Khemakhem and Boujelbene (2017) highlights, the aim of SVM is to find the boundary hyperplane parameters by maximizing the distance between the hyperplane and the support vectors from the training sample space. A general framework of SVM is highlighted in Figure 3.7.

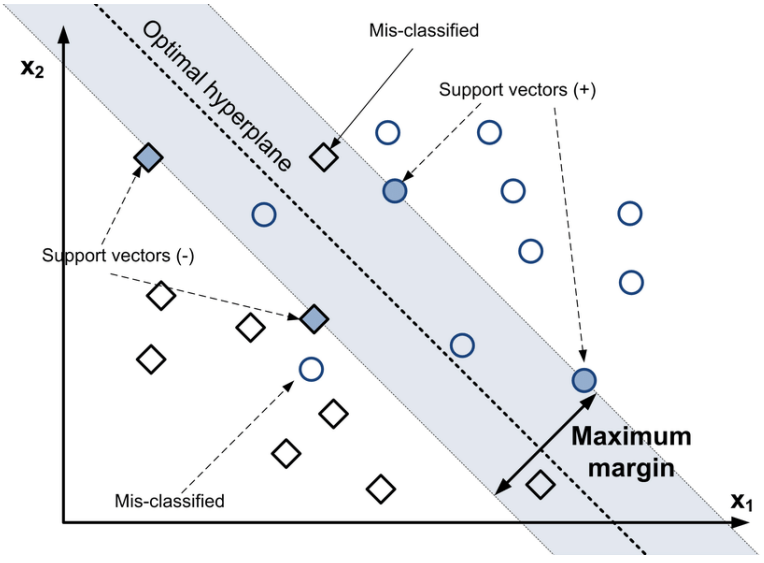


FIGURE 3.7: Concept of Support Vector Machines [Nguyen Duc et al. (2017)]

For a binary classification problem, SVM seeks to separate the two classes by mapping the input vectors to a higher dimensional space and then constructing an optimal hyperplane with maximum margin to achieve optimal separation (margin) between the classes (Kaya et al., 2008). Given a training set with labelled instance pairs $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$, $x_i \in R^n$ and $y \in Y = \{-1,\ +1\}$, where each x_i belongs to the domain and each y belongs to the label space, the optimal decision boundary or hyperplane in SVM can be defined as

$$w \cdot x + b = 0 \tag{3.22}$$

As Goh and Lee (2019) further express, the convex optimization problem is then determined as

$$\begin{aligned} \min \quad & \phi(w, b) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\epsilon_i, \\ \text{s.t.} \quad & y_i\,(w \cdot x_i + b) \geq 1 \end{aligned} \tag{3.23}$$

where a Lagrange function is utilized to find the global maximum. The regularization factor C in Equation 3.23 is responsible for balancing the importance between the maximization of the margin width and the minimization of the training error (Zhang et al., 2015). In addition, the slack variable ϵ_i takes into account the misclassification. Liu and Lin (2005) further describe, a nonlinear classification problem can be converted into a linear classification problem (see Equation 3.24) by applying a kernel trick which can be then used to distinguish customers with low and high default risk.

$$f(x) = \text{sgn}\left(\sum_{i=1}^{n}{\alpha_i}^* y_i K\left(x_i, x_j\right) + b^*\right) \tag{3.24}$$

For SVM, popular kernel choices are linear, Radial Basis Function (RBF), and polynomial (Goh and Lee, 2019).

Chapter 4

Data Organization

In this section, characteristics of the used dataset, common pre-processing steps, and feature selection techniques are discussed.

4.1 Dataset characteristics

In this master thesis, a publicly available open-source Ethereum fraud transaction dataset from Kaggle, provided by Vagiv Aliyev in 2021, was used to model potential frauds in the cryptocurrency blockchain. This data contains 9,841 former Ethereum transactions, consisting of known frauds and valid transferals, described by 49 features. A full list of the attributes in combination with a short description of their meaning are depicted in Table B1, Table B2, and Table B3. Furthermore, descriptive statistic are displayed in Table B4 and Table B5. For the quantitative attributes in the data, numerical statistics such as row-count (count), mean, standard deviation (std), minimum (min), maximum (max), 25th/50th, and 75th percentile were computed.

In addition, the binary outcome variable $FLAG$ shows if a transaction was previously detected as fraud, where 0 indicates *non-fraud* and 1 indicates *fraud*. By investigating the target variable in greater depth, one can observe that the dataset is highly imbalanced (or skewed), with a total of 77.86% non-frauds and 22.14% frauds. In machine learning, class imbalance constitutes an important, yet challenging problem, given that most classifiers were designed around the assumption of class equality, i.e. far more examples are labeled as one class, while far fewer instances are labeled as the other class, the more important class (Guo et al., 2008). To tackle this issue, a methodology called Synthetic Minority Oversampling (SMOTE) is shown in section 4.4.

4.2 Data preprocessing

In any machine learning project, the most time-intensive part constitutes the data preparation and configuration task. According to Press (2016), data scientists spend around 80% of their work time cleaning up and pre-processing data, as opposed to doing actual machine learning engineering or generating insights. Since the individual processing tasks are outside the scope of this master thesis, a more profound analysis of the constructed preparation pipeline is given in section 4.2.3.

4.2.1 Outlier & missing value handling

As seen from the descriptive statistics Table B4 and B5, missing values as well as outliers can be found. In total, for 25 numeric variables out of the 49 features overall, missing values are observable. However, given their insignificance with respect to the total amount, for instance, the attribute $ERC20\_most\_rec\_token\_type$ has the highest portion of missing data, with a ratio of just 8.60%; the null-values were handled via median-imputation as part of the processing-pipeline (see section 4.2.3 for more details).

Regarding outliers or anomalies, this can most easily be detected by conducting an exploratory data analysis (EDA), more specifically, by computing distribution plots per attribute. Exemplary, this was performed for two features, depicted in Figure A.1 and Figure A.2. Based on these analyses, outliers were individually removed, resulting in a reduction of the dataset from originally 9,841 to 8,059 rows.

4.2.2 Correlation analysis

In statistics, correlation is defined as a measure of a linear relationship between two variables. Finding two features that are statistically correlated can have a crucial impact not only in the feature selection process but also on model performance and accuracy. Even though they may not always worsen a model, they will however not improve its predictive power either. In the econometric literature, the phenomenon of having multiple variables that are dependent on each other is called *multicollinearity*. Particularly for linear models like logistic regression, collinearity in the variables may yield unstable and varying results. Hence, often a simpler model with fewer features is preferred over a needlessly complex and slow one (the concept of ***Occam's Razor*** makes this more precise).

More generally, as Yu and Liu (2004) confirm, the redundancy between attributes is most widely measured in terms of their correlation. The (Pearson) correlation coefficient ρ_{xy} for the two variables x_i and y_i, as the most commonly used procedure for numerical variables, is determined by the following formula:

$$\rho_{xy} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}} \tag{4.1}$$

where x_i and y_i are two vectors of length n, and $\bar{x}$ and $\bar{y}$ correspond to the means of x and y, respectively. Generally, the value of the correlation coefficient ρ ranges between -1 and +1. While values of ρ greater than 0 indicate a positive relationship between two variables, where an increase in the value of one feature increases the value of another, a correlation below 0 indicates the same relationship, just in a negative direction.

To highlight the dependencies between the attributes, a correlation matrix (heatmap) was computed, as seen in Figure A.3. Note that the correlation coefficients along the diagonal of the matrix are all equal to 1 because each variable is perfectly correlated with itself. Furthermore, the matrix is perfectly symmetrical along the diagonal since the correlation between two variables, x_i and y_i, is commutative: $Corr(X, Y) = Corr(Y, X)$. In addition, due a large number of variables, the heatmap only displays correlation if it is larger than 0.5.

Figure A.3 displays the heatmap of the Ethereum fraud dataset, where a high positive correlation is shown in dark blue, high negative correlation in bright yellow, while light-green represents no correlation between two variables. As visible from the correlation matrix, multiple features exist that possess a correlation close to 1 (perfect positive correlation), for example *ERC20 avg val sent* and *ERC20 max val sent*, *ERC20 max val rec* and *ERC20 total Ether received*, or *total ether sent contracts* and *max val sent to contract*. Looking at the other side of the scale, (strong) negative correlation can only be found in one feature pair, *total ether balance* and *max val sent*. The importance of handling correlation in a dataset is particularly of interest when it comes to machine learning modeling, given that highly correlated features might deteriorate the performance of the overall model. As a result, to overcome this issue, feature selection techniques, specifically Principal Component Analysis (PCA) and Recursive Feature Elimination (RFE) are introduced in section 4.3, which can handle the dependencies among the variables. Additionally, as a first data preparation step, correlated features greater or equal to the threshold of 0.9 were dropped from the dataset.

4.2.3 Pipeline

As highlighted in the beginning of this chapter, data cleaning constitutes the most time-consuming part of a machine learning project. Hence, to simplify the overall process, **pipelines** are most often applied to optimize and automate the end-to-end workflow. In essence, pipelines constitute an independently executable workflow of complete machine learning tasks that streamline multiple pre-processing steps, for instance, variable encoding, feature selection, normalization, model fitting, and classification (Pedregosa et al., 2011).

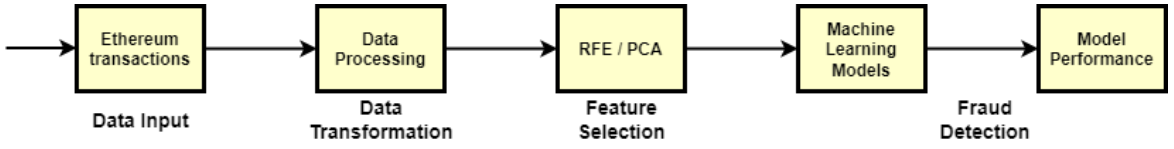


FIGURE 4.1: Fraud detection pipeline

In this thesis, the pipeline module *scikit-learn.preprocessing* and its individual components from the Python package scikit-learn are used. When dealing with pipelines, three important terms must be explained beforehand:

- **Estimator**: To put it simply, the object that learns from the data, i.e. fits the data, is called an estimator. Within this object, methods like ***fit*** and ***predict*** are incorporated.
- **Transformer**: As the name already implies, transformers convert the data towards a pre-defined format, later used in the machine learning model. For instance, often used transformers in practice are *OneHotEncoder*, *SimpleImputer* and *StandardScaler*. Like other estimators, they are represented by classes with a ***fit*** method, which is responsible for learning a model from the training data, and a ***transform*** method that is in charge of applying the learned transformation on unseen test data.
- **ColumnTransformer**: Often, it is necessary only to apply certain transformers to a distinct group of attributes, e.g. $OneHotEncoder$ only to categorical columns. To do so, $ColumnTransformer$ is an effective tool to achieve that goal.

The used pre-processing pipeline in this work project is pictured in Figure 4.2, which includes both a numerical and a categorical transformer. For the first, *StandardScaler*, responsible for scaling the features, and a *SimpleImputer*, responsible for filling any missing values with the median of the respective column, were applied. The *StandardScaler* within the *scikit-learn* package is defined in the following way:

$$z_{ki} = \frac{x_{ki} - \bar{x}_k}{s_k} \tag{4.2}$$

where $\bar{x}_k$ is the mean of the training sample and s_k is the standard deviation of the sample for a specific attribute k (Raschka, 2014). Furthermore, the mean of the sample x_{ki}, where $i = 1 \dots n$, can be written as

$$\bar{x}_k = \frac{1}{n} \left(\sum_{i=1}^{n} x_{ki} \right), \tag{4.3}$$

and the standard deviation of a sample is computed as follows

$$s_k = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (x_{ki} - \bar{x}_k)^2}. \tag{4.4}$$

Similar to the numeric, the categorical transformer also incorporates a *SimpleImputer*, however, with a constant filling method. Additionally, categorical values are converted to integers by leveraging *OneHotEncoder* for the dummy encoding.

```
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='constant', fill_value='missing')),
    ('onehot', OneHotEncoder(handle_unknown='ignore', sparse=False))])

numeric_features = X.select_dtypes(include=np.number).columns
categorical_features = X.select_dtypes(exclude=['number']).columns

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)])

preprocessor.fit(X_train)

# Get final feature names
cat_columns = preprocessor.named_transformers_['cat']['onehot'].get_feature_names
    (categorical_features)
columns_pipeline = np.append(cat_columns, numeric_features)
```

FIGURE 4.2: Pre-processing pipeline

4.3 Feature selection

Feature selection is the procedure of reducing the number of variables within a dataset when developing a statistical model, based on certain methodologies. Particularly for fraud transaction data, multiple variables can be collected not only during the transaction process but also before and after. By excluding entirely irrelevant, insignificant, and unimportant features, evaluation results such as accuracy as well as the overall performance of the classification model may improve. However, as Laborda and Ryoo (2021) note, the selection process is considered to be complicated, arbitrary, and unsystematic given that there is no uniform theory and the process of feature selection works differently on different data.

Various techniques in the domain of feature selection have been developed, ranging from classical univariate analysis which determines the relative attribute importance by calculating either the correlation of the predictor on the class variable independently or between the predictors themselves, to supervised classification models that perform the selection process (Ang et al., 2015). For instance, some supervised algorithms contain built-in feature selection techniques, that only select predictors which help to maximize the accuracy of the model (Kuhn and Johnson, 2013). An overview of different feature selection techniques is depicted in Figure A.4.

Broadly speaking though, feature selection methods can be divided into three main categories: *filter*, *wrapper* and *embedded* selection techniques (Jović et al., 2015).

- **Filter:** As the name suggests, filter methods select features based on their characteristics or performance measure, regardless of the employed modeling algorithm - hence, they are computationally less expensive (Naqvi, 2012).
- **Wrapper:** In contrast to filter methods, wrapper based functions select a feature subset by the quality of the performance of a statistical learning algorithm. Thus, when dealing with classification tasks, a wrapper will evaluate attributes based on the classifier's performance (Ang et al., 2015).
- **Embedded methods:** Unlike the two before mentioned techniques, embedded methods perform feature selection during the modelling algorithm's execution (Miao and Niu, 2016).

In this thesis, a filter method called *Principal Component Analysis* (PCA) and a wrapper feature selection technique named *Recursive Feature Elimination* (RFE) are utilized. Although both mentioned techniques receive criticism from practitioners, it is crucial to stress that no ideal method exists, given that each of the methods for variable selection has its pros and cons.

4.3.1 Variable selection with RFE

As Guyon et al. (2002) note, *Recursive Feature Elimination* (RFE) is regarded as a backward selection technique for the predictors. The basic idea behind this method is that it starts by building a model on the entire dataset, i.e. with all predictors, and simultaneously assigns weights or feature importance scores to each of the attributes according to a pre-defined external estimator. Based on those scores, the least important predictor(s) are removed recursively, the model is re-built, and feature importance's scores are re-computed again (Kuhn and Johnson, 2019). That procedure is recursively repeated on the down-scaled dataset until a desired number of attributes is reached (Pedregosa et al., 2011).

Initially, RFE was first proposed to perform feature selection for SVM models (Guyon et al., 2002). However, recursive feature elimination was similarly also applied to random forest estimators, as discussed by Jiang et al. (2004) and Svetnik et al. (2004). In this paper, the variable selection was performed with the help of the popular yellowbrick library, in combination with a random forest classifier as an estimator and *recall* as scoring evaluation.

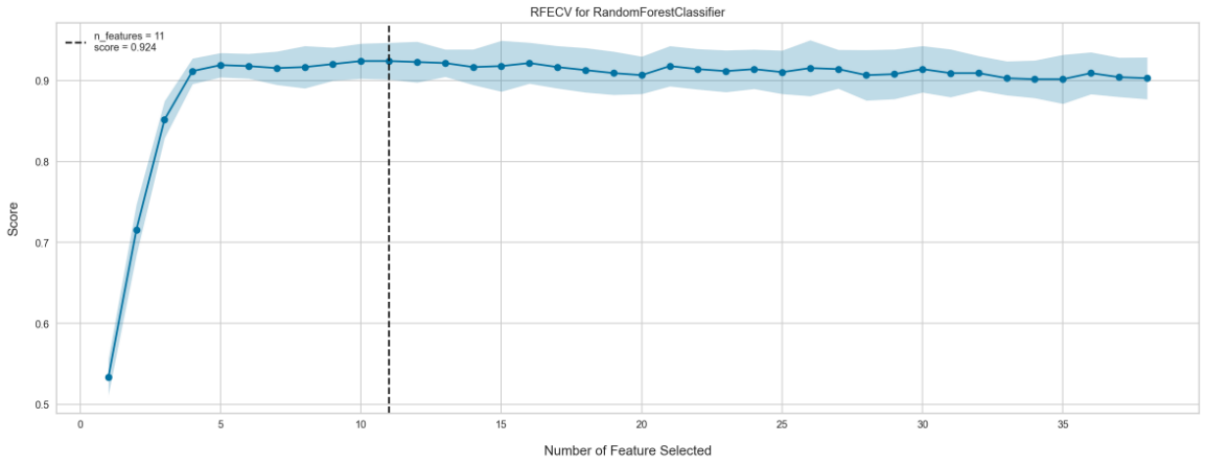


FIGURE 4.3: RFECV for RandomForestClassifier

The result of the cross-validated RFE, in short *RFECF*, is shown in Figure 4.3. From the output, it is noticeable that according to the algorithm, 11 out of the 40 features were selected. This number also constitutes the amount of variables that will be incorporated in the machine learning models. The light blue shaded area displays the variability of cross-validation with one standard deviation above and below the mean accuracy score drawn by the graph (Granström and Abrahamsson, 2019). Furthermore, it is observable that the score hardly varies after 5 features, which might be of interest when computational power is an important factor to consider.

4.3.2 Variable selection with PCA

Principal Component Analysis (PCA), introduced by Pearson (1901), is a popular unsupervised learning technique, based on the linear transformation that is used to reduce the dimensionality of the input data, whilst preserving as much of the relevant information as possible. However, while PCA has many advantages like removing collinear features or reducing overfitting, its biggest drawback is that it lacks interpretability. Given that the new set of features, i.e. *principal components*, are a linear transformation of the features from the original set of attributes, they become less interpretable (Kamperis, 2021).

From a mathematical point of view, as Ma (2014) describes, PCA is defined as a linear transformation that converts the given data to a new coordinate system such that the principal components are uncorrelated, and that the first principal component lies on the coordinate that has the largest variance by the projection of the data, the second component lies on the coordinate with second largest variance, and so on. In a more formal setting, Boehmke (2017) summarized this in the following way:

Suppose the first principal component of a dataset $X_1, X_2, ..., X_p$ is the linear combination of the features

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + ... + \phi_{p1}X_p \tag{4.5}$$

that has the largest variance and where ϕ_1 is the first principal component loading vector, with elements $\phi_{12}, \phi_{22}, \ldots, \phi_{p2}$. Note that ϕ are normalized (see Equation 4.7), resulting in $\sum_{j=1}^{p} \phi_{j1}^2 = 1$. In the next step, after the first principal component (Z_1) has been determined, the second principal component (Z_2) can be computed, which, similar to the first, is a linear combination of $X_1, \ldots, X_p$. Additionally, the second principal component possess maximal variance out of all linear combinations that are uncorrelated with Z_1.

Analogically to the first principal component scores, the second $z_{12}, z_{22}, \ldots, z_{n2}$ take the form as displayed in Equation 4.6:

$$Z_2 = \phi_{12}X_1 + \phi_{22}X_2 + ... + \phi_{p2}X_p \tag{4.6}$$

This procedure is repeated until all principal components $Z_1, Z_2, ..., Z_p$ are determined. The key point in this process is to compute the loadings ϕ, which are calculated by finding the vector that maximizes the variance. To be more concrete, the principal component that maximizes the

variance of all projected points onto a certain feature space is the eigenvector of the covariance matrix associated with the largest eigenvalue (Raschka, 2014).

Therefore, to compute the mentioned principal components, following mathematical steps, as chronologically ordered, are incorporated into this algorithm:

1. **Standardization of the raw data**

   Per definition, PCA incorporates the variance, which by itself depends on a certain unit of measurement. This further implies that the PCA based on the covariance matrix **S** will change if the units of measurement on one or more of the variables change (Jolliffe and Cadima, 2016). Thus, in order to overcome this issue, variable scaling or standardization is applied. This methodology works in a similar way as described in Equation 4.2, where $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$ are the instances of the dataset with n dimensional inputs.

$$x_j^{(i)} = \frac{x_j^{(i)} - \bar{x}_j}{\sigma_j} \quad \forall j \tag{4.7}$$

2. **Covariance matrix computation**

   By far the most important part of PCA are the eigenvectors and eigenvalues of a covariance (or correlation) matrix. The eigenvectors, which constitute the principal components, determine the directions of the new feature space, and the eigenvalues determine their magnitude (Brownlee, 2019). Generally, the covariance matrix is a square matrix $(n \times n)$ that describes the covariance between the elements with each other. The covariance matrix Σ is defined as follows:

$$\Sigma = \frac{1}{m} \sum_{i}^{m} \left(x_{(i)}\right) \left(x_{(i)}\right)^T, \quad \Sigma \in \mathbb{R}^{n \times n} \tag{4.8}$$

3. **Eigendecomposition: computing eigenvectors and eigenvalues**

   In linear algebra, an **eigenvector** is defined as a vector whose direction remains unchanged when a linear transformation is applied to it (Shlens, 2014). Additionally, the corresponding eigenvalue is the factor by which the eigenvector is scaled. The eigenvector $\vec{v}$ of a matrix Σ is the vector for which the following holds:

$$\Sigma \vec{v} = \lambda \vec{v} \tag{4.9}$$

where λ is a scalar called the **eigenvalue**. Furthermore, this can be re-written as:

$$\begin{aligned} \Sigma\vec{v} - \lambda\vec{v} &= 0 \\ \Rightarrow \vec{v}(\Sigma - \lambda I) &= 0 \end{aligned} \tag{4.10}$$

where $\boldsymbol{I}$ is the identity matrix.

4. **Selecting k principal components according to their eigenvalues**

   Since the goal of PCA is to reduce the dimensionality of the original feature space, i.e., projecting the feature space onto a smaller subspace, where the eigenvectors will form the axes of this new feature subspace, it is important to notice that these eigenvectors are unit eigenvectors with length equal to 1 (Raschka, 2015).

   Thus, when it comes to the decision of which eigenvector(s) can be dropped from the feature space without losing too much of the variability in the data, one needs to inspect the eigenvalues: Given that a higher eigenvalue corresponds to a higher variability, the principal axis with the higher eigenvalue will be an axis capturing more variance in the data. By sorting the eigenvectors with respect to their decreasing order of eigenvalues, the desired amount of principal components $\boldsymbol{k}$ can be achieved, where k corresponds to the number of dimensions.

   Hence, the first column in the re-arranged eigenvector matrix constitutes the principal component that captures most of the variance in the data. The eigenvalues in decreasing order are shown in Figure 4.4:

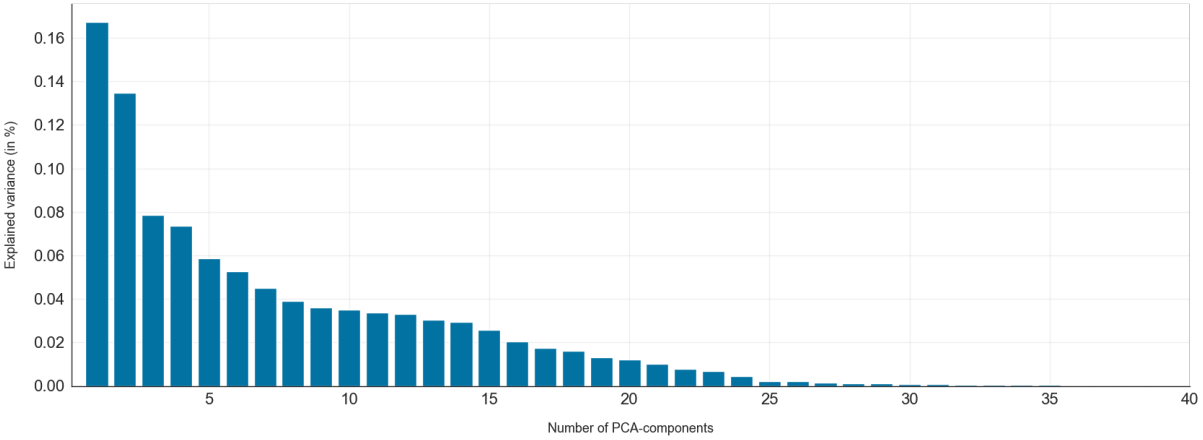


FIGURE 4.4: Variance explained in % by each principal component

The bar chart displays the proportion of variance explained by the principal components in decreasing order. From there it is possible to see that PC1 explains approx. 17% of the variance, PC2 explains 14% of the variance, and so on.

By looking at Figure 4.4, a pronounced drop-off (elbow shape) after the fifth principal component can be seen. Nonetheless, a clear choice regarding how many principal components should be chosen for the new feature subspace is hard to extract from this plot. Thus, often it is advantageous to display the cumulative explained variance ratio as a function of the number of components, as shown in Figure 4.5, which can be calculated from the eigenvalues. In essence, this curve tells how much information (variance) is contained within the first k principal components. For instance, within the first 12 PCs, approx. 80% of the variance can be explained, while 40 components are needed to explain 100% of the variance in the dataset.

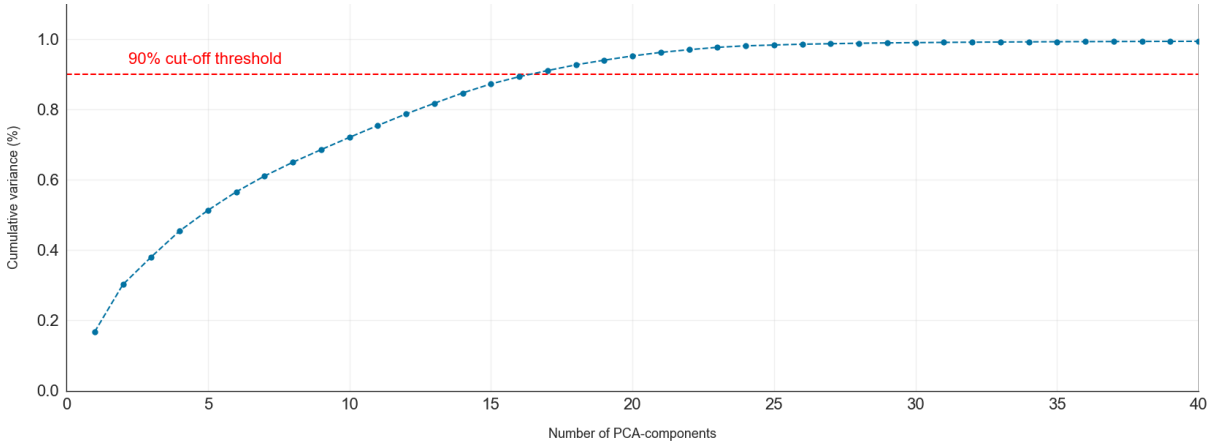


FIGURE 4.5: Cumulative explained variance ratio

An often-used method to decide on the number of principal components is by setting a fixed threshold, for instance, 80% or 90%, and stop when the first k components account for a percentage of the total variation greater than this threshold (Hartmann et al., 2018). Here, a limit of 90% was chosen, which results roughly in a total of 16 PCs.

4.4 Handling of imbalanced data with SMOTE

As Yanminsun et al. (2011) note, an imbalanced classification in a biclass scenario constitutes a problem where the imbalance degree of a class distribution can be denoted by the ratio of the sample size of the small class to that of the prevalent class.

Two widely accepted sampling techniques in machine learning that try to balance skewed datasets are ***random undersampling*** and ***random oversampling***. While the former samples majority class data instances to form a representative set of similar size to the minority class and discard the rest, the latter duplicates synthetic examples by replication or interpolation from the underrepresented class to balance the training set (Mukherjee and Khushi, 2021). A widely used extension to oversampling, developed by Chawla et al. (2002), constitutes **S***ynthetic* **M***inority* **O***versampling* **T***echnique* (SMOTE), which generates synthetic samples from the minority class by applying the KNN approach. Mathematically, as Walimbe (2017) emphasizes, this works as follows: First, the algorithm selects k-nearest neighbors, joins them together, and creates the synthetic samples in the space. Secondly, SMOTE takes the feature vectors in addition to its nearest neighbors and computes the distance between these vectors. In the last step, the difference is multiplied by a random number between (0, 1) and it is added back to the feature, as pictured in Equation 4.11:

$$\mathrm{x}_{\mathrm{new}} = x + \mathrm{rand}(0,1) \times (\overline{\mathrm{x}} - \mathrm{x}), \tag{4.11}$$

where $\mathrm{rand}(0,1)$ refers to a random number between 0 and 1. For a more understandable explanation, Figure 4.6 displays the conceptual overview of the SMOTE framework.

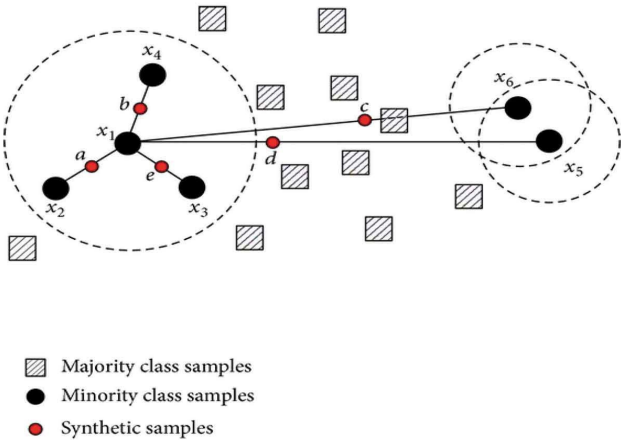


FIGURE 4.6: SMOTE algorithm [Shrivastava et al. (2020)]

Chapter 5

Evaluation Metrics and Feature Interpretability

In the model evaluation section, background information regarding how to correctly evaluate and choose the best model classification-error metric is addressed.

5.1 Model Performance Assessment

Once the data processing and feature engineering part is completed, and the model successfully trained, the next step is to find out how effective the algorithm is based on some pre-defined metric. Generally, the evaluation process in machine learning can be regarded as important as the building procedure, given that it estimates the model prediction accuracy and prediction errors by using new and unseen data.

In this section, some of the most commonly used binary classification metrics and methods for assessing the relative performance of predictive classification models will be addressed, namely:

- Confusion Matrix
- Binary classification measures such as accuracy, precision, and recall
- Receiver Operating Characteristic (ROC)
- Area Under the Curve (AUC)

5.1.1 Confusion Matrix

A confusion matrix is used to determine how many observations were correctly or incorrectly classified - thus, it provides a more detailed breakdown of the results for each class. In a binary classification context, usually, it is common to present the *true positive* (TP), *true negative* (TN), *false positive* (FP) and *false negative* (FN) predictions. By looking at Figure 5.1, one can see that those values are presented in the form of a matrix where the x-axis shows the actual values while the y-axis displays the predicted outcomes. Moreover, a confusion matrix per se does not constitute a metric to evaluate a model, it rather serves as a background foundation to comprehend further classification metrics such as accuracy, precision, and recall.

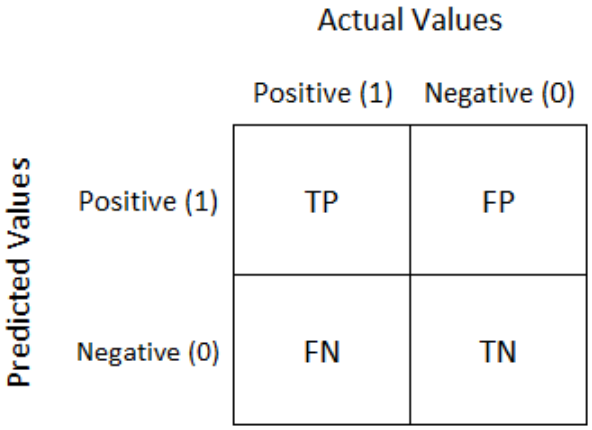


FIGURE 5.1: Binary Confusion Matrix [Shin (2020)]

Each quadrant of the table has its unique meaning, which differs depending on the classification problem. An interpretation of the individual components is given in the following:

- **true positives**: the model says the transaction is fraudulent, and it actually is
- **true negatives**: the model says the transaction is not fraudulent, and it actually isn't
- **false positives**: the model says the transaction is fraudulent, whereas in reality it isn't
- **false negatives**: the models says the transaction isn't fraudulent, whereas in reality it is

The diagonal of the confusion matrix, i.e. TP and TN, represent the correct predictions, where the goal is to maximize those values. Most often, in statistics, FPs are also referred to as *Type I errors*, and FNs as *Type II errors*.

5.1.2 Binary Classification Measures

Accuracy is one metric for evaluating the performance of classification models. It is the measure of how many observations were correctly classified, both positive and negative instances, as shown by the formula:

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

An alternative to using accuracy is to use ***precision*** and ***recall*** as performance metrics. Precision is defined as the number of true positives divided by the total number of positive predictions. Mathematically, this can be expressed as:

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

Finally, ***recall*** (or sensitivity) measures the model's ability to detect positive samples.

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

5.1.3 Receiver Operating Characteristic (ROC)

A ***receiver operating characteristic curve*** (ROC) is a two-dimensional graph that depicts the performance of a classification model at all classification thresholds. Creating a ROC curve is done by plotting the sensitivity on the y-axis and the 1-specificity on the x-axis for all possible thresholds (Ji et al., 2018). While classifiers that are closer to the top-left corner of the plot indicate better relative performance, models that are closer to the 45-degree diagonal of the ROC space indicate less accurate algorithmic performance, as shown in Figure 5.1.

5.1.4 Area Under the Curve (AUC)

While the ROC curve is a performance measure at various thresholds, the ***area under a receiver operating characteristic curve*** (AUC) is a single aggregated scalar that represents the degree or measure of separability. As Moura Oliveira et al. (2019) notes, AUC can be interpreted as the probability that the model ranks a random positive example more highly than a random negative example. In other words, the larger the AUC, the better the model is in predicting 0 classes as 0 and 1 classes as 1.

5.2 Test, training and validation sets

By definition, machine learning is defined as the study and construction of algorithms that learn from given data, where predictions can be made from the trained model (Ron, 1998). Typically, the data utilized in the model for training and testing purposes comes from multiple sources, particularly from three sets, as shown in Figure 5.2.

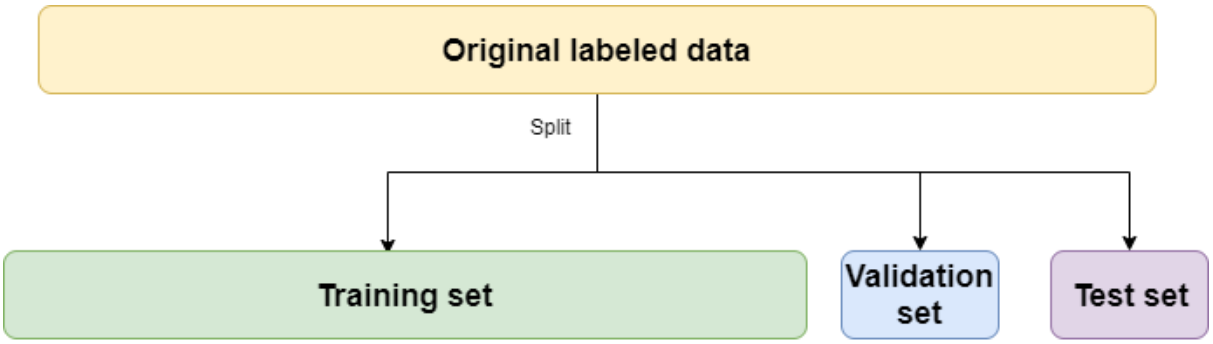


FIGURE 5.2: Test, training and validation set [Villalobos (2020)]

While all three sets are usually split from the original data source, each one normally possesses its own distinct use in statistical learning. A high-level overview is given in the following:

- **Training set**: As the name indicates, the training set constitutes the data source where the algorithm is evaluated repeatedly to learn about the data's behavior and its features. Furthermore, it has the largest corpus from all three mentioned sets.
- **Validation set**: During the training respectively hyperparameter-tuning procedure, the validation set provides an unbiased sample to evaluate the fitted model.
- **Test set**: After the training process is finalized, the test set is used to conduct an unbiased evaluation of the model. It provides a final check if the learned algorithm is capable of correctly predicting unseen data.

In this paper, the original Ethereum fraud dataset was split according to a 70%, 15% and 15% ratio, which is in line with frequently conducted train, validation and test splits, as Draelos (2019) summarizes in the following:

1. Option: 70% train, 15% validation, 15% test
2. Option: 80% train, 10% validation, 10% test
3. Option: 60% train, 20% validation, 20% test

5.3 K-fold cross-validation

K-fold cross-validation is a procedure in statistical learning that is used during the hyperparameter tuning, to evaluate a model's optimal performance respectively its testing error. In essence, it is a resampling technique (without replacement) that involves randomly dividing the set of observations into k groups (or folds) of approximately equal size (Motta, 2020). The main idea is to leave out part k, fit the existing model to the remaining $k-1$ folds, and then obtain predictions for the left out k-*th* part, where each time, a different group of observations is treated as a validation set (Rajan, 2018). The diagram given in Figure 5.3 summarises the concept behind k-fold cross-validation with k = 5 folds. In the case of 5-fold cross-validation, the model is trained on the first four groups in the first iteration and evaluated on the fifth. The next iteration considers the first, second, third, and fifth fold and evaluates on the fourth. This procedure is repeated three more times, where at every point the evaluation takes place on a different fold.

The CV computation for the pre-defined metric, for the sake of this paper, let the metric be MSE, is performed in turn for each part $k = 1, 2, ..., K$, where the results are combined and averaged:

$$\mathrm{CV}_{(K)} = \sum_{k=1}^{K} \frac{n_k}{n} \mathrm{MSE}_k \tag{5.1}$$

The parameter n_k are the observations in part k, $\mathrm{MSE}_k = \sum_{i \in C_k} (y_i - \hat{y}_i)^2 / n_k$, and $\hat{y}_i$ is the fit for the observation i, which is obtained from the data with k removed.

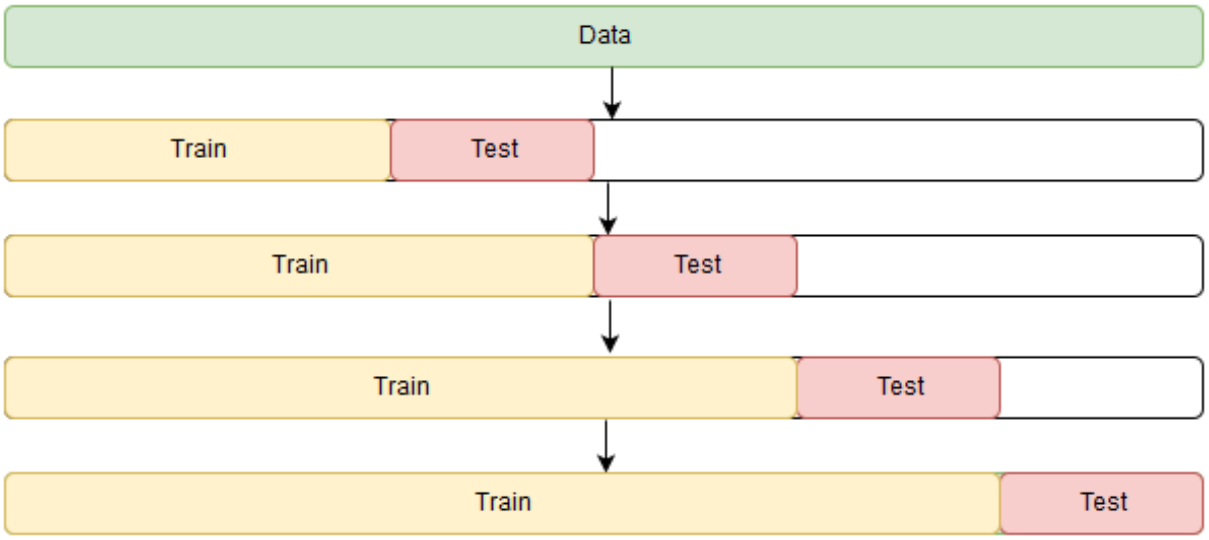


FIGURE 5.3: K-fold cross validation framework [Garg (2017)]

5.4 Feature Importance with SHAP

Frequently, a well-tuned and cleverly designed machine learning model may produce reliable and accurate predictions but notoriously lacks when it comes to feature interpretation since many methods, for example, Support Vector Machine, are regarded as so-called *black-box models* (Dornigg, 2022). To put it simply, a machine learning practitioner knows which parameters go into a model and what comes out, but there is no possible way to fully understand the calculation that takes place under the hood of an algorithm. However, when designing a model, it is essential for the end-user to understand not only which variables have an impact on the final decision, but also how it is making the final prediction. To assist in this matter, the European Commission High-Level Expert Group on AI presented the Ethics Guidelines for Trustworthy Artificial Intelligence in April 2019 that put forward a set of seven key requirements that AI systems should meet in order to be deemed trustworthy. Among this list, the following points discuss the concepts of explainable artificial intelligence (XAI):

- **Human agency and oversight**: While AI systems should empower human beings by making informed decisions, there should also be proper oversight mechanisms enabled, which can be achieved through human-in-the-loop, human-on-the-loop, and human-in-command approaches.
- **Transparency**: AI systems and their decisions should be explained to the concerns of their stakeholders in charge. Moreover, humans need to be aware that they are interacting with an AI system, and must be informed of the system's capabilities and limitations.
- **Accountability**: AI systems should put mechanism in place to ensure responsibility, accountability, and auditability.

To bridge the gap between the proposed ethical guidelines and models that are highly accurate and precise but difficult to interpret, a novel approach that can deal with both of these aspects is proposed in this section.

The suggested methodology, called ***Shapley Additives***, can be applied to any predictive output, irresponsible of the underlying model. Inspired by several other previously conducted work in the field of model interpretability, such as Ribeiro et al. (2016), Štrumbelj and Kononenko (2013) and Lipovetsky and Conklin (2001), Lundberg and Lee (2017) proposed SHapley Additive exPlanations value as a unified approach to explain individual machine learning model outputs.

SHAP is originally based on Shapley values, first introduced by Shapley (1953), a concept incorporating a cooperative game-theoretical approach. As Bussmann et al. (2021) note, it offers a breakdown of variable contributions so that every data point is not only represented by input features but also by variable contributions to the prediction of the trained machine learning model. More formally, based on its game-theoretical origin, the idea of SHAP is to explain the prediction of an instance x by computing the contribution of each feature to the prediction (Molnar, 2021). Lundberg and Lee (2017) specified this as follows:

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i = \text{bias} + \sum \text{contribution of each feature} \tag{5.2}$$

where the explanation model $g(x')$ for the prediction $f(x)$ is constructed by an additive feature attribution method, which decomposes the prediction into a linear function of the binary variables $z' \in \{0,1\}^M$ and the quantities $\phi_j \in \mathbb{R}$ (Bussmann et al., 2021). Furthermore, the SHAP value evaluates the difference to the output f_x made by including the feature i or all the combinations of features other than i (Lu, 2018). This can be expressed as:

$$\phi_i(f, x) = \sum_{z' \subseteq x'} \frac{|z'|!\,(M - |z'| - 1)!}{M!} \left[f_x\left(z'\right) - f_x\left(z' \backslash i\right) \right] \tag{5.3}$$

where $|z'|$ is the number of non-zero entries in z', and $z' \subseteq x'$ represents all z' vectors where the non-zero entries are a subset of the non-zero entries in x' (Lundberg and Lee, 2017).

Also, the authors of SHAP define three desirable properties that only the additive feature attribution method satisfies:

- **Local accuracy**: If x' is the simplified x, $g(x') = f(x)$.
- **Missingness**: If a feature is locally zero ($x'_i = 0$), the Shapley value has no impact.
- **Consistency**: Changing a model in such a way that one of its feature has a larger overall contribution will also change the impact of the Shapley value in the same way.

Due to the rather technical description of the SHAP framework, Molnar (2021) provides a simplified interpretation possibility: *Given the current set of feature values, the contribution of a feature value to the difference between the actual prediction and the mean prediction is the estimated Shapley value.*

Chapter 6

Model Results

This section highlights the main module findings which are based on the methodologies stated in the previous chapters.

6.1 Model performance on training/test-set

Based on the methods developed in section 4 and 5, specifically the feature selection methodologies, and the different performance criteria metrics, the combined results are displayed in Table 6.1 for PCA selected features respectively Table 6.2 for RFECV selected features. In addition, to show the visual performance of the models, ROC curves and their respective AUC values were plotted for both feature selection techniques in Figure 6.1.

From Table 6.1 it can be observed that in the PCA feature selected models, XGBoost and Random Forest yielded relatively similar positive results, while the Support Vector Machine classifier performed the worst, given that it was, among others, not capable of providing outcomes for the false-negative class. Almost the same results are obtained from the models with the RFECV selected features, which can be seen in Table 6.2. The results of the Random Forest and XGBoost classifier are again very much similar in terms of their classification performance. Also, similar to the previous results, the Support Vector Machine model was not able to capture any false negatives of the Ethereum dataset. Interestingly, similar to SVM, logistic regression as the traditional benchmark classifier performed relatively worse than certain more advanced machine learning algorithms. Last but not least, a conclusion can be made that the tree-based models (Decision Tree and Random Forest) performed on average better than the boosting

models (AdaBoost and XGBoost), if several indicators are taken into account, foremost recall, false positive, false negative and true negative.

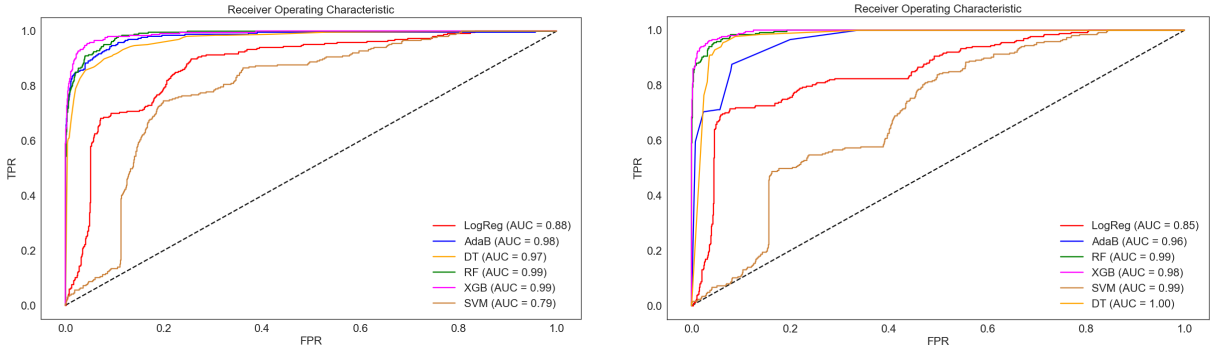
Model	Accuracy (%)	Recall/Sensitivity (%)	Precision (%)	TPs (%)	FPs (%)	FNs (%)	TNs (%)
Logistic Regression	69.00	92.51	32.59	14.42	29.83	1.17	54.58
XGBoost	96.67	92.13	87.23	14.36	2.10	1.23	82.31
AdaBoost	93.35	91.01	72.97	14.19	5.25	1.40	79.16
Decision Tree	91.77	89.89	67.80	14.01	6.65	1.58	77.76
Random Forest	95.04	88.76	81.16	13.84	3.21	1.75	81.20
Support Vector Machine	31.41	100.00	18.52	15.59	68.59	0.00	15.82

TABLE 6.1: Model-performance summary (PCA feature selected)

Model	Accuracy (%)	Recall/Sensitivity (%)	Precision (%)	TPs (%)	FPs (%)	FNs (%)	TNs (%)
Logistic Regression	57.50	89.14	25.40	13.89	40.81	1.69	43.61
XGBoost	97.55	93.63	90.91	14.59	1.46	0.99	82.95
AdaBoost	71.57	100.00	35.41	15.59	28.43	0.00	55.98
Decision Tree	94.92	92.88	78.48	14.48	3.97	1.11	80.44
Random Forest	95.45	95.13	79.62	14.83	3.79	0.76	80.62
Support Vector Machine	28.78	100.00	17.96	15.59	71.22	0.00	13.19

TABLE 6.2: Model-performance summary (RFECV feature selected)

As mentioned in section 5.1.3, to visualize the performance of classification models, ROC curves and their respective AUC values were computed, depicted in Figure 6.1, which support the numerical interpretation performed above. From the two plots, it can be concluded that there is no worth remarking difference in the ROC curves and AUC scores between the two feature selection techniques. For instance, while the area under the curve score for XGB with PCA is 0.99, the same score with RFECV is 0.98. In general, though, the values are slightly lower for the principal component selection technique than for the other. For the sake of completeness, the line segment from (0,0) to (1,1) has an area of 0.5 and is called the chance diagonal (Devos et al., 2007). Every outcome above this threshold can be considered as a performance not happening by pure chance.

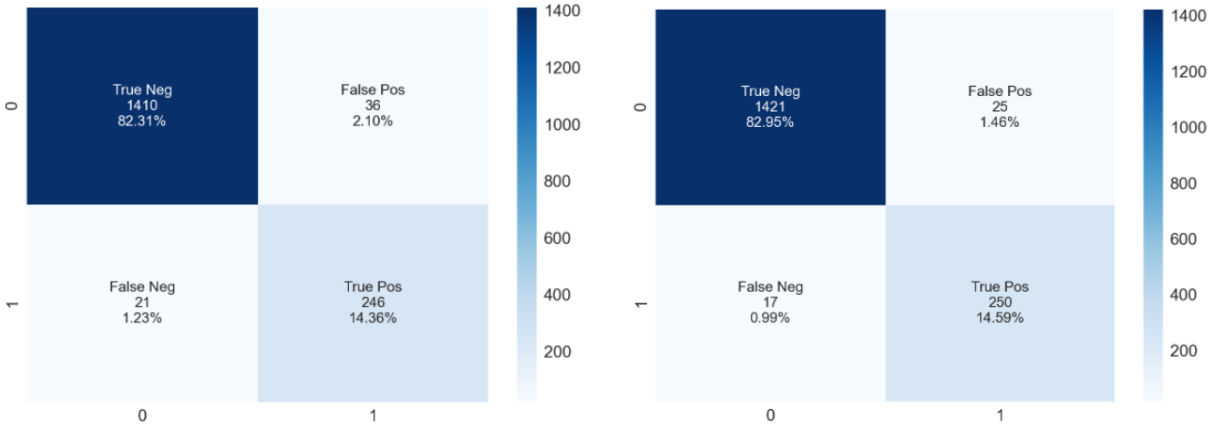


(a) ROC curve with PCA features

(b) ROC curve with RFE features

FIGURE 6.1: Left plot (a) displays ROC curve with PCA selected features, the right plot (b) with RFECV selected features

Furthermore, to summarize the performance of the best classification algorithm, confusion matrices, as explained in section 5.1.1, were configured for the XGB model.



(a) Confusion matrix with PCA features

(b) Confusion matrix with RFE features

FIGURE 6.2: Left plot (a) displays the confusion matrix with PCA selected features, the right plot (b) with RFECV selected features

Similar to the ROC curves, hardly any (significant) difference between the performance and its feature selection technique can be seen, indicating that both methods are suitable for putting into real-life production.

6.2 Model performance on holdout-set

In the last section, the performance of the models was solely tested on an in-sample training/-validation set. Nonetheless, for a classifier to be regarded as stable and well-trained, it also needs to perform well on unseen data (holdout-set). As a result, the respective performance measures are displayed in Table 6.3 and Table 6.4.

Model	Accuracy (%)	Recall/Sensitivity (%)	Precision (%)	TPs (%)	FPs (%)	FNs (%)	TNs (%)
XGBoost	96.94	93.65	87.62	14.64	2.07	0.99	82.30
Random Forest	94.87	88.36	80.68	13.81	3.31	1.82	81.06

TABLE 6.3: Model-performance summary (PCA feature selected)

Model	Accuracy (%)	Recall/Sensitivity (%)	Precision (%)	TPs (%)	FPs (%)	FNs (%)	TNs (%)
XGBoost	98.26	96.30	92.86	15.05	1.16	0.58	83.21
Random Forest	97.44	94.18	89.90	14.72	1.65	0.91	82.71

TABLE 6.4: Model-performance summary (RFECV feature selected)

Given that both outcomes indicate stable results across several performance metrics, it is possible to conclude that XGB, as well as RF, constitute reliable and steady classifiers.

6.3 Model interpretation with SHAP

Based on the definition of SHAP given in section 5.4, two commonly used plots within the SHapley Additive exPlanations framework, (1) **summary beeswarm** and (2) **global feature summary**, are displayed and explained for the optimized Random Forest classifier.

The summary plot depicted in Figure 6.3 sorts variables by their respective Shapley value magnitude and gives a sense of the behavior an individual feature has on the overall classification model. The y-axis corresponds to the features ordered by their importance and the x-axis shows the SHAP values. The color represents the value of the feature where red stands for a high and blue for a low impact. Also, overlapping points that do not fit on the row are jittered or piled up in the y-axis direction, displaying the distribution of the Shapley values per attribute.

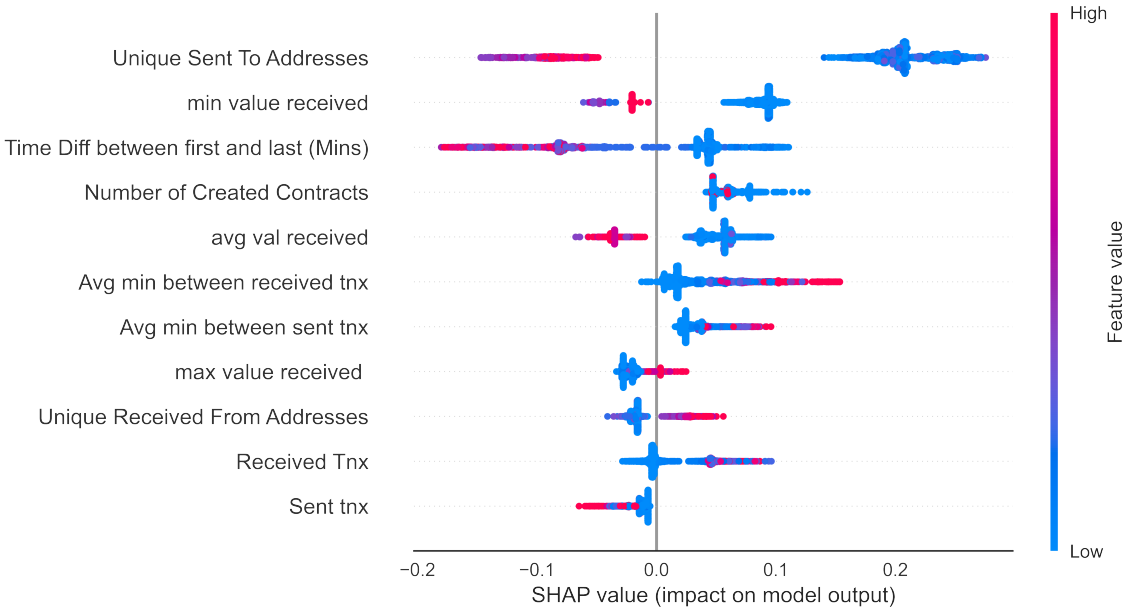


FIGURE 6.3: SHAP summary (beeswarm) plot

For example, it is possible to see from this figure that SHAP in conjunction with RFEECV, selects the eleven features, listed on the y-axis of the plot, as the most important ones of the Random Forest classifier. The first feature, which constitutes the most relevant one, can be interpreted as follows: the larger the amount of unique addresses from which an account sent transactions, the larger its negative impact on the overall model performance (the same logic can be applied to the rest of the attributes). A slightly less technical interpretation is that if a user has more transactions to unique addresses, the probability of a potential fraud case is also higher. However, the biggest drawback of using the beeswarm plot is that it tends to represent values imprecisely, specifically in terms of its total influence on the model.

Fortunately, the Shapley framework offers a large variety of models that can deal with this issue, one of them being the global SHAP feature importance plot. As Molnar (2021) explains, the idea behind this method is simple: attributes with larger absolute Shapley values are regarded as more important. The computation is done by averaging the absolute Shapley values per feature across the data, as shown in Equation 6.1:

$$I_j = \sum_{i=1}^{n} |\phi_j^{(i)}| \tag{6.1}$$

The variable importance plot in Figure 6.4 depicts the most influential variables in descending order, i.e. the top attributes have a higher predictive contribution to the model than the bottom ones. As seen from this plot, the issue of imprecisely showing each attribute's total impact was successfully taken care of.

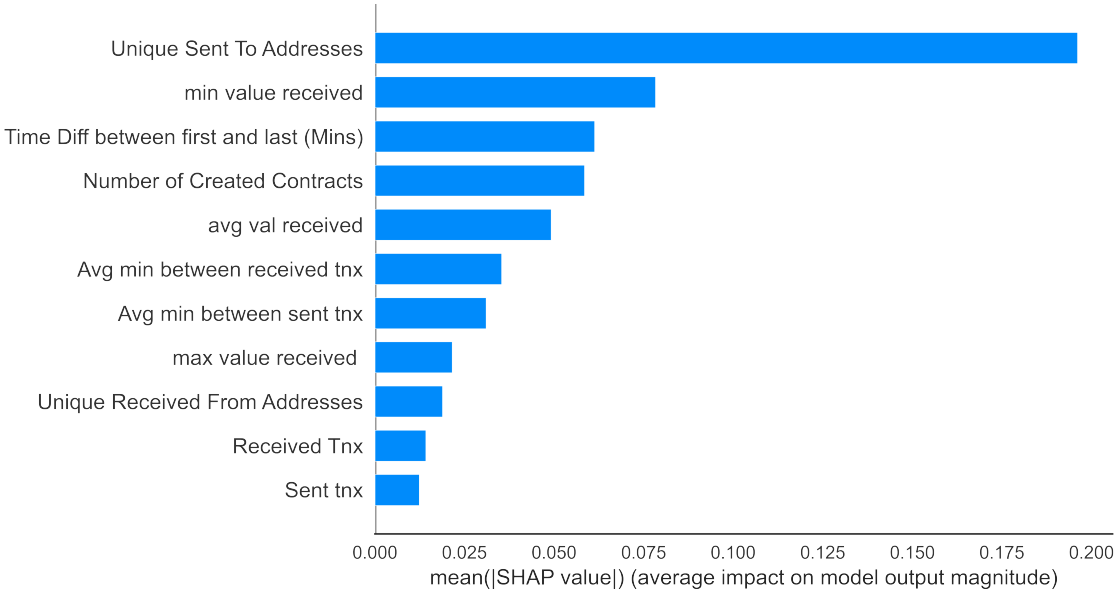


FIGURE 6.4: SHAP (global) feature importance plot

Using the interpretation technique provided by Molnar (2021), interpreting the SHAP feature importance measured as the mean absolute Shapley values becomes an uncomplicated task: the amount of total addresses from which an account sent transactions was the most important feature, changing the predicted absolute probability of detecting fraudulent transactions on average by approx. 20 percentage points (0.200 on x-axis).

6.4 Comparison of results of the suggested approach

Comparing the results of this paper with the outcomes of Farrugia et al. (2020), one can conclude that both are similar in terms of their overall predictive accuracy. For instance, while the XGBoost's accuracy is ca. 0.97, with an AUC value for PCA and RFECV selected features of approx. 0.99 in this thesis, the accuracy score in Farrugia et al. (2020) for a 10 fold cross-validated XGBoost classifier is roughly 0.963 ($\pm$ 0.006) with an average AUC of 0.994 ($\pm$0.0007). In terms of the most significant attributes, the comparison paper utilized the incorporated feature importance function of the XGB classifier, while this paper used the SHAP framework to determine the variables with the largest impact. While the top three predictors in Farrugia et al. (2020) are "Time diff between first and last (Mins)", "Total Ether balance" and "Min value received", the top three variables established with SHAP are "unique sent to addresses", "min value received" and "Time Diff between first and last transaction".

Chapter 7

Potential Impact of Fraud Models and Regulation in the U.S.

In this chapter, practical applications of the discussed fraud detection model and its potential impact on cryptocurrency prices, as well as the general market stability, are briefly highlighted.

7.1 Cryptocurrency market manipulation

While the proponents of cryptocurrencies claim that the technology offers numerous advantages, foremost the disruption of the traditional payment system (see section 1.3 for more details), the crypto market has also been subject to security breaches and wild price fluctuations (Gandal et al., 2018). For instance, the value of Bitcoin skyrocketed during the year 2017, increasing from around US $1,000 to an all-time peak of approx. US $20,000, before an abrupt sell-off destroyed more than half of its market capitalization (see Figure A.6). Given the abnormal price surge at the time, many industry players expressed their concern that the token price was actively inflated or (partly) pushed upwards by activities that can be linked to Bitfinex, one of the largest and least regulated exchanges in the industry, registered in the Caribbean (Popper, 2018).

In fact, according to an investigative study conducted by Finance professors John M. Griffin from the University of Texas with a background in forensics, and Ohio State University professor Amin Shams, who reviewed over 200 gigabytes of transaction history data between Bitcoin (BTC) and Tether (USDT) over a period of March 2017 till March 2018, backed the news rumor that the entire rise of the digital currency at the time is attributable to *"one large player"*, revealing

a manipulation scheme during that boom (Vigna, 2019). Their investigation clearly showed several unique patterns that someone inside the crypto-exchange successfully worked to push up prices when they sagged at other exchanges (Popper, 2018). Interestingly, the study came out after an analysis had been published by the cryptocurrency asset manager Bitwise in March 2019, showing that nearly 95% of Bitcoin's spot trading is faked, i.e. only US $273 million of approx. US $6 billion in average daily Bitcoin transactions was legitimate (Sheetz, 2019). As the analysis of Bitwise highlighted, artificial volume was created by *wash trading*, a term used in the context of simultaneously selling and buying the same underlying - Bitcoin in this case - to create the false impression of vigorous (fake) market activity.

By studying the fraud cases, the question arises, why regulators and the general public should care about manipulation happening in the token world? After all, cryptocurrency valuation is still far behind current stock market capitalization. However, as recent trends clearly show, Bitcoin, Ethereum, and other coins are becoming more and more mainstream as an alternative source to conventional payment systems (Gandal et al., 2018). Thus, it is crucial for state authorities and other related parties to understand the functionalities of the ecosystem. Specifically, knowledge in that domain comes in handy when analyzing volume fraud, which has become increasingly popular since 2017. For example, as Matthew Hougan from Bitwise elaborates, there is an incentive for an exchange to report fake volume since initial coin offerings (ICOs) are commonly performed on exchanges that inherit the largest trading volume, as higher trading levels signals enhanced trustworthiness to investors (Rooney, 2019). The way this works is that transactions that are occurring on exchanges are not directly recorded on the individual blockchain, rather they are maintained by the exchanges themselves, thus, opening a loophole to counterfeit trading volume by wash trading (Chen et al., 2022). As a result, supposedly legit exchanges skim off listing fees that can range from US $1 million to US $3 million per ICO, constituting charges that are up to 10-times more than conventional stock exchanges would demand for securities, according to information from Autonomous Next (Russo, 2018). Also, despite reporting higher trading volumes, dodgy exchanges showed on average systematic higher levels of spreads.

A rather stark example of this is given in the Bitwise report: While at the time of the price comparison (December 12, 2018), Bitcoin was trading at a spread of $34.74 on *CoinBene*, the spread on *Coinbase Pro* was approx. $0.01. However, interestingly, the larger spread was observable on an exchange that claimed to have 18x more trading volume than Coinbase Pro, which is economically not plausible. In total, out of the 81 evaluated exchanges, only 10 provided solid and legitimate volume figures, according to Bitwise. A similar investigation

was also conducted in South Korea in August 2020, where officials found that the third-largest cryptocurrency exchange *Coinbit* was allegedly responsible for earning at least 100 billion won (nearly US $85 million) illegally and forging more than 99% of its trading volume (Palmer, 2020). To tackle those issues, Chen et al. (2022) proposed an empirical data-mining approach to detect whether an exchange is actively faking trading volume by designing seven metrics based on two different kinds of transaction data. Their analysis indicate that at the time the analysis was carried out, Huobi, a Seychelles-based cryptocurrency exchange, faked its trading volume the most, whereas Binance appears to be the least fraudulent one.

The mentioned examples as highlighted above already demonstrate the need for more elaborate fraud detection mechanisms to guarantee a safe and fraudless cryptocurrency trading framework. Particularly, the same issues were already raised by U.S. regulators, such as the SEC Chairman in 2019, arguing that most cryptocurrency trading platforms are not utilizing the same monitoring tools as traditional stock exchanges; thus, no fair assessment of a token's price can be conducted without misleading customers (Liu et al., 2019). Therefore, one could argue that by utilizing fully automatic, state-of-the-art machine learning methods that are capable of scanning millions of transactions for suspicious behavior instantly, the overall cryptocurrency market could become more secure and efficient, thereby serving its original purpose to create a payment system that is self-regulating and independent of outside intervention.

7.2 Cryptocurrency regulation: Enforcement and policy responses in the U.S. with regards to fraud

As discussed, the P2P framework of cryptocurrencies offers multiple benefits and opportunities when compared to conventional monetary systems, such as lower transaction costs, an increase in traceability & security, and a higher level of efficiency (Rejeb et al., 2021). But with any new financial innovation, particularly highly volatile one, crypto-coins are not free of drawbacks, and especially decentralization can be considered a double-edged sword (Eigelshoven et al., 2021). This stems from the fact that no central authority exists that regulates the market and protects investors from fraudulent activities or manipulation. Nonetheless, for a government to oversee this new type of asset class, one needs to define its legality first. As a result, governments around the globe, and also the U.S., are not trying to develop a new set of governing rules, but instead, each case is being tried according to laws that were intended to regulate conventional payment systems, financial services, and investments (Hughes, 2017).

For cryptocurrency regulation in the U.S. this means that both the Federal and state governments have focused to come up with regulatory frameworks. For instance, as Dewey (2022) mentions, at the federal level, the following agencies are involved: The Securities and Exchange Commission (the "**SEC**"), the Commodity Futures Trading Commission (the "**CFTC**"), the Federal Trade Commission and the Department of the Treasury, through the Internal Revenue Service (the "**IRS**"), the Office of the Comptroller of the Currency (the "**OCC**") and the Financial Crimes Enforcement Network ("**FinCEN**"). Due to the involvement of multiple agencies, and the different area of responsibility per public authority, each one of them categorizes cryptocurrencies differently. In particular, the SEC views cryptos as a security, the CFTC calls it a commodity, the IRS a property, and FinCEN regards it as a convertible virtual currency.

Independent of the various legal definitions, the subsumption of the new digital asset class into already existing financial law makes it possible for state authorities to charge fraudsters and scammers. Most often, the starting point for prosecutors seeking indictments related to cryptocurrency activity is the US wire fraud statute since those who are found guilty of criminal activities while transacting with a cryptocurrency largely fall into this category, which is legally defined in 18 USC § 1343 (Levi, 2021):

> *Whoever, having devised or intending to devise any scheme or artifice to defraud, or for obtaining money or property by means of false or fraudulent pretenses, representations, or promises, transmits or causes to be transmitted by means of wire, radio, or television communication in interstate or foreign commerce, any writings, signs, signals, pictures, or sounds for the purpose of executing such scheme or artifice, shall be fined under this title or imprisoned not more than 20 years, or both. If the violation occurs in relation to, or involving any benefit authorized, transported, transmitted, transferred, disbursed, or paid in connection with, a presidentially declared major disaster or emergency (as those terms are defined in section 102 of the Robert T. Stafford Disaster Relief and Emergency Assistance Act (42 U. S. C. 5122)), or affects a financial institution, such person shall be fined not more than $ 1,000,000 or imprisoned not more than 30 years, or both.*

The federal criminal charge of wire fraud can be applied to various criminal activities given that the term *wire* involves any form of telecommunication, such as phone, fax, text message, radio, television, social media message, email, or any other form of cable communication that falls within the remit of this statute (Fisher, 2021). As Levi (2021) further notes, the US legal

system has an extensive list of regulations affiliated with financial crimes of fraud and attempted fraud, which, among others include the following:

- Section 32(a) of the Securities Exchange Act of 1934 (Exchange Act)
- Section 24 of the Securities Act of 1933 (Securities Act)
- Sarbanes-Oxley Act of 2002 (SOX)
- Mail and wire fraud statutes (18 U.S. C. §§1341, 1343)
- Bank fraud statute (18 U.S.C. §1344)
- Misapplication and embezzlement statute (18 U. S. C. § 656)
- Fraudulent and fictitious claims statute (also known as the Criminal False Claims Act)
- Tax evasion and fraud provisions of the Internal Revenue Code (26 U.S. C. §§7201, 7206)
- Computer Fraud and Abuse Act of 1986 (18 U. S. C. §1030$(a)(4)$)
- False statements statute (18 U.S.C. §1001)
- Major Fraud Act of 1988 (18 U.S.C. §1031)

Moreover, within the broad legal definition of *wire fraud*, several subcategories exist that are used in conjunction for crypto related topics. For instance, wire fraud & scams [see court examples: DOJ (2022a), DOJ (2022b)], wire fraud & theft [see court examples: DOJ (2022c), DOJ (2022d)], or wire fraud & ICOs [see court examples: DOJ (2021), DOJ (2020)].

Chapter 8

Discussion

The final chapter of this thesis discusses the posed research question and potential limitations & future work with regards to fraud models.

8.1 Research question

The goal of this master thesis was to analyze the potential effects of using state-of-the-art machine learning models versus traditional fraud models to detect fraudulent transactions in the Ethereum blockchain, by using a similar setting to Farrugia et al. (2020) as the baseline. Even though advanced statistical models have yielded excellent performance in the financial fraud area, there are still some ongoing issues that remain unsolved, which need further, more thorough investigation. As a result, the following chapter briefly answers the posed research question, and further information is provided with regard to major challenges and future work from a model-oriented perspective.

The answer to the research question is as follows:

By utilizing state-of-the-art machine learning modeling techniques, how well do they perform in predicting fraudulent transactions in Ethereum, respectively, which attributes have the largest impact on the outcome?

As highlighted in section 6, more advanced artificial intelligence models can outperform traditional, previously used classifiers, foremost Random Forest and XGBoost, measured on several statistical metrics, such as accuracy, precision, and false-negative rate. Superior performance was demonstrated by both feature selection techniques, providing almost identical outcomes. In

addition, similar to the numerical conclusion, a visual comparison (ROC-AUC plot) yields the same result. It is also worth mentioning that the superior performance was not only demonstrated in an in-sample dataset, but also on a holdout-set (out-of-the-sample). Furthermore, based on the game-theoretical feature interpretability methodology provided by SHAP (see section 6.3), the top three variables that have the largest predictive explainability are "unique sent to addresses", "min value received" and "Time Diff between first and last transaction".

8.2 Limitations and future work

The future tasks with respect to fraud detection modeling via machine learning methods are:

- A major issue that persists in the fraud detection domain is that fraudulent activities are most often complex, and non-typical financial actions. With the steady increase in digital transformation over the last decade, this process accelerated, making the detection procedure even more challenging. Thus, methods that are regarded as significantly more sophisticated than supervised learning techniques, such as *Artificial Neural Nets* (ANN), which originate from the Deep Learning framework, may have to be utilized.
- In many cases, fraud detection can be considered as a binary classification task, which requires a training sample with fraud as well as non-fraudulent entries. However, given the first bullet point, fraud activities are increasingly difficult to detect, making the holistic gathering of representative fraud data samples virtually impossible for market regulators.
- As indicated in section 4, the open-source Ethereum dataset was directly used from Kaggle, thus, potential (incorrect) assumptions about the data and its representativeness of the overall population were taken as a given.

The future tasks related to legal issues in cryptocurrencies:

- One of the main ideas of the blockchain technology, as highlighted in the beginning of this thesis, is the concept of enhanced privacy for any type of transaction, thereby no third party can pinpoint the actual location of a ledger. However, at the same time, this poses a complex jurisdictional challenge, as the individual nodes of the blockchain can be located in various countries or jurisdictions, thus, conflicts may arise, which legal framework is ultimately applicable when cross-border transactions are carried out.
- Financial fraud and privacy concerns go hand in hand when it comes to challenges within the world of cryptos. Due to the promise of anonymity, and the resulting belief in total

freedom, criminals have switched to cryptocurrencies to conduct their financial transactions, providing them a new, and almost untraceable way to either commit or use this new form of technology as a means of payment service.

Bibliography

Shivani Agarwal. Introduction. Binary Classification and Bayes Optimal Classifier. *University of Pennsylvania: CIS 520 - Machine Learning*, Spring 2019.

Michael Akers and Jodi Bellovary. What is fraud and who is responsible? *Accounting Faculty Research and Publications*, January 2006.

W.S. Albrecht, C.O. Albrecht, C.C. Albrecht, and M.F. Zimbelman. *Fraud Examination*. Cengage Learning, 2015. ISBN 9781305840485.

JC Ang, Andri Mirzal, Habibollah Haron, and Haza Nuzly Abdull Hamed. Supervised, Unsupervised, and Semi-Supervised Feature Selection: A Review on Gene Selection. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 13(5):971–989, September 2015. doi: 10.1109/TCBB.2015.2478454.

Farrukh Nizam Arain. Decision Tree Classification Algorithm. https://www.devops.ae/decision-tree-classification-algorithm/, April 2021. Accessed: 2022-04-02.

Manuel Artís, Mercedes Ayuso, and Montserrat Guillen. Detection of Automobile Insurance Fraud with Discrete Choice Models and Misclassified Claims. *Journal of Risk and Insurance*, 69:325 – 340, September 2002. doi: 10.1111/1539-6975.00022.

Martin Birch, Roy Katzenberg, Judy Finn, Stephen Hill, Richard Sharp, Allan McDonagh, Martin Robinson, and Mia Campbell. Fraud risk management: A guide to good practice. *Chartered Institute of Management Accountants*, 2008.

Brad Boehmke. UC Business Analytics R Programming Guide: Principal Components Analysis. http://uc-r.github.io/pca, February 2017. Accessed: 2022-03-24.

Bradley Boehmke and Brandon M. Greenwell. *Hands-On Machine Learning with R*. The R Series. Taylor & Francis Group, Boca Raton, Florida, 1 edition, 2020. ISBN 978-1138495685.

Philip Boucher, Susana Nascimento, and Mihalis Kritikos. How blockchain technology could change our lives. *STOA - Science and Technology Options Assessment*, February 2017. doi: 10.2861/926645.

Leo Breiman. Bagging predictors. *Machine Learning*, 24:123–140, August 1996. doi: 10.1007/BF00058655.

Leo Breiman. Random Forests. *Machine Learning*, 45:5–32, October 2001. doi: 10.1023/A:1010933404324.

Leo Breiman, Jerome Friedman, Charles J. Stone, and Richard A. Olshen. *Classification and Regression Trees*. Taylor & Francis Group, Boca Raton, Florida, January 1984. ISBN 9780412048418.

Jason Brownlee. How to Choose a Feature Selection Method For Machine Learning. https://tinyurl.com/feature-selection-techniques, November 2019. Accessed: 2022-04-18.

Jason Brownlee. *Machine Learning Mastery with Python: Understand your Data, Create Accurate Models and Work Projects End-To-End*. Machine Learning Mastery. 1.20 edition, August 2021. ISBN 979-8540446273.

Niklas Bussmann, Paolo Giudici, Dimitri Marinelli, and Jochen Papenbrock. Explainable Machine Learning in Credit Risk Management. *Computational Economics*, 57:203–216, January 2021. doi: 10.1007/s10614-020-10042-0.

Kristof Böhmer and Stefanie Rinderle-Ma. Mining association rules for anomaly detection in dynamic process runtime behavior and explaining the root cause to users. *Information Systems*, 90:101438, September 2019. doi: 10.1016/j.is.2019.101438.

Recep Can, Sultan Kocaman, and Candan Gokceoglu. A Comprehensive Assessment of XGBoost Algorithm for Landslide Susceptibility Mapping in the Upper Basin of Ataturk Dam, Turkey. *Applied Sciences*, 11(11):4993, May 2021. doi: 10.3390/app11114993.

Yongtao Cao and Roland Francis. On forecasting the community-level COVID-19 cases from the concentration of SARS-CoV-2 in wastewater. *Science of the Total Environment*, 786:147451, September 2021. doi: 10.1016/j.scitotenv.2021.147451.

Rich Caruana and Alexandru Niculescu-Mizil. An Empirical Comparison of Supervised Learning Algorithms. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML

'06, pages 161–168, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933832. doi: 10.1145/1143844.1143865.

Rich Caruana, Nikolaos Karampatziakis, and Ainur Yessenalina. An empirical evaluation of supervised learning in high dimensions. pages 96–103, January 2008. doi: 10.1145/1390156.1390169.

N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16:321–357, June 2002. doi: 10.1613/jair.953.

Jialan Chen, Dan Lin, and Jiajing Wu. Do cryptocurrency exchanges fake trading volumes? An empirical analysis of wash trading based on data mining. *Physica A: Statistical Mechanics and its Applications*, 586:126405, 2022. ISSN 0378-4371. doi: 10.1016/j.physa.2021.126405.

Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794, New York, USA, August 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939785.

Marine Collery. Rule Learning from Time-Dependent Data Applied to Fraud Detection. In *RuleML+RR*, 2021.

Corda. About R3: Blockchain/DLT 101. https://www.r3.com/blockchain-101/#:~:text=DLT%20is%20a%20decentralized%20database,cryptographic%20signature%20called%20a%20hash., 2021. Accessed: 2022-04-12.

Don Cowan. Random Forest Models. https://www.ml-science.com/random-forest, 2020. Accessed: 2022-004-05.

Andy. K Devos, Sabine van Huffel, Arjan W. Simonetti, Marinette van der Graaf, Arend Heerschap, and Lutgarde M.C. Buydens. Chapter 11 - Classification of Brain Tumours by Pattern Recognition of Magnetic Resonance Imaging and Spectroscopic Data. In Azzam F.G. Taktak and Anthony C. Fisher, editors, *Outcome Prediction in Cancer*, pages 285–318. Elsevier, Amsterdam, 2007. ISBN 978-0-444-52855-1. doi: 10.1016/B978-044452855-1/50013-1.

Josias N. Dewey. *Blockchain & Cryptocurrency Laws and Regulations 2022*, volume 4th Edition. Latham & Watkins LLP, Global Legal Insights, 2022.

Department of Justice DOJ. Co-Founder Of Cryptocurrency Company Pleads Guilty For Role In ICO Fraud Scheme, June 2020. URL https://www.justice.gov/usao-sdny/pr/co-founder-cryptocurrency-company-pleads-guilty-role-ico-fraud-scheme. Accessed: 2022-05-06.

Department of Justice DOJ. Leading Co-Founder of Cryptocurrency Company Sentenced to 8 Years in Prison for ICO Fraud Scheme, March 2021. URL https://tinyurl.com/ICO-fraud-8-years-prison. Accessed: 2022-05-06.

Department of Justice DOJ. Disbarred Attorney Pleads Guilty To $5 Million Cryptocurrency Fraud, February 2022a. URL https://www.justice.gov/usao-sdny/pr/disbarred-attorney-pleads-guilty-5-million-cryptocurrency-fraud. Accessed: 2022-05-05.

Department of Justice DOJ. San Francisco Man Charged In Alleged Cryptocurrency Investor Fraud Scheme, April 2022b. URL https://www.justice.gov/usao-ndca/pr/san-francisco-man-charged-alleged-cryptocurrency-investor-fraud-scheme. Accessed: 2022-05-05.

Department of Justice DOJ. Two Arrested for Alleged Conspiracy to Launder $4.5 Billion in Stolen Cryptocurrency, February 2022c. URL https://www.justice.gov/opa/pr/two-arrested-alleged-conspiracy-launder-45-billion-stolen-cryptocurrency. Accessed: 2022-05-05.

Department of Justice DOJ. Ceo of Major Online Cryptocurrency Exchange Company Indicted for Defrauding Company's Customers, Destroying Evidence, and Tax Evasion, January 2022d. URL https://www.justice.gov/opa/pr/two-arrested-alleged-conspiracy-launder-45-billion-stolen-cryptocurrency. Accessed: 2022-05-05.

Thomas Dornigg. Credit Risk Modeling: Predicting Customer Loan Defaults with Machine Learning Models. Master's thesis, Nova School of Business and Economics, January 2022. URL https://run.unl.pt/bitstream/10362/140582/1/2021_22_fall_41727_thomas-dornigg.pdf.

Rachel Draelos. Best Use of Train/Val/Test Splits, with Tips for Medical Data. https://bit.ly/train_test_split_best_use, September 2019. Accessed: 2022-04-09.

ECB. Virtual currency schemes: a further analysis. *Policy Department for Economic, Scientific and Quality of Life Policies*, February 2015.

Bradley Efron. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*, 7(1):1–26, January 1979. doi: 10.1214/aos/1176344552.

Felix Eigelshoven, André Ullrich, and Douglas Parry. Cryptocurrency Market Manipulation: A Systematic Literature Review. December 2021.

Steve Farrugia, Joshua Ellul, and George Azzopardi. Detection of illicit accounts over the Ethereum blockchain. *Expert systems with applications*, 150, 2020. ISSN 0957-4174. doi: 10.1016/j.eswa.2020.113318.

Primavera De Filippi. Blockchain Technology and Decentralized Governance: The Pitfalls of a Trustless Dream. Decentralized Thriving : Governance and Community on the Web 3.0. February 2020. doi: 10.2139/ssrn.3524352.

Evan Fisher. Wire Fraud, December 2021. URL https://www.findlaw.com/criminal/criminal-charges/wire-fraud.html. Accessed: 2022-05-05.

Fraud.net. Rules-Based Fraud Detection, 2021. URL https://fraud.net/d/rules-based-fraud-detection/. Accessed: 2022-04-27.

Y. Freund and R. Schapire. Experiments with a new Boosting Algorithm. ICML, page 148–156. Machine Learning: Proceedings of the Thirteenth International Conference, July 1996.

Neil Gandal, Jt Hamrick, Tyler Moore, and Tali Oberman. Price manipulation in the Bitcoin ecosystem. *Journal of Monetary Economics*, 95(C):86–96, 2018. ISSN 0304-3932. doi: 10.1016/j.jmoneco.2017.12.004.

Jatin Garg. Using k-fold cross-validation for time-series model selection. https://bit.ly/k_fold_cross_validation, March 2017. Accessed: 2022-04-14.

R. Y. Goh and L. S. Lee. Credit Scoring: A Review on Support Vector Machines and Metaheuristic Approaches. *Advances in Operations Research*, 2:1–30, March 2019. doi: 10.1155/2019/1974794.

Daria Granström and Johan Abrahamsson. Loan Default Prediction using Supervised Machine Learning Algorithms. Master's thesis, KTH, School of Engineering Sciences (SCI), 2019. URL http://kth.diva-portal.org/smash/get/diva2:1319711/FULLTEXT02.pdf.

Roger Grosse. Lecture 3: Linear Classification. *CSC 411: Introduction to Machine Learning*, January 2019a.

Roger Grosse. Lecture 4: Training a Classifier. *CSC 411: Introduction to Machine Learning*, January 2019b.

Xinjian Guo, Yilong Yin, Cailing Dong, Gongping Yang, and Guangtong Zhou. On the Class Imbalance Problem. *Fourth International Conference on Natural Computation, ICNC '08*, 4, October 2008. doi: 10.1109/ICNC.2008.871.

Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene Selection for Cancer Classification Using Support Vector Machines. *Machine Learning*, 46:389–422, January 2002. doi: 10.1023/A:1012487302797.

K. Hartmann, J. Krois, and B. Waske. *E-Learning Project SOGA: Statistics and Geospatial Data Analysis*. Department of Earth Sciences. Freie Universitaet Berlin, 2018.

Barbara Jane Holland. *Handbook of Research on Knowledge and Organization Systems in Library and Information Science*. Brooklyn Public Library & Independent Researcher, June 2021. ISBN 9781799872580. doi: 10.4018/978-1-7998-7258-0.

Robby Houben and Alexander Snyers. Cryptocurrencies and blockchain: Legal context and implications for financial crime, money laundering and tax evasion. *Policy Department for Economic, Scientific and Quality of Life Policies*, July 2018.

Scott D. Hughes. Cryptocurrency Regulations and Enforcement in the u.s. *Western State Law Review*, 45:1, 2017.

IBM-Cloud-Education. Boosting, May 2021. URL https://www.ibm.com/cloud/learn/boosting. Accessed: 2022-04-15.

Rahmeh Fawaz Ibrahim, Aseel Mohammad Elian, and Mohammed Ababneh. Illicit Account Detection in the Ethereum Blockchain Using Machine Learning. In *2021 International Conference on Information Technology (ICIT)*, pages 488–493, 2021. doi: 10.1109/ICIT52682.2021.9491653.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning with Applications in R*. Springer Science+Business Media, 2013. ISBN 978-1-4614-7137-0.

Yuhe Ji, Guangsheng Zhou, Qijin He, and Lixia Wang. The Effect of Climate Change on Spring Maize: Suitability across China. *Sustainability*, 10(10), October 2018. ISSN 2071-1050.

H. Jiang, Y. Deng, H. S. Chen, L. Tao, Q. Sha, J. Chen, C. J. Tsai, and S. Zhang. Joint analysis of two microarray gene-expression data sets to select lung adenocarcinoma marker genes. *BMC bioinformatics*, 5(81), June 2004. doi: 10.1186/1471-2105-5-81.

Ian T. Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A*, 374(2065), April 2016. doi: 10.1098/rsta.2015.0202.

Michael I. Jordan. Binary Classification. *CS281A/Stat241A: Statistical Learning Theory*, September 2004.

A. Jović, K. Brkić, and N. Bogunović. A review of feature selection methods with applications. In *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1200–1205, 2015. doi: 10.1109/MIPRO.2015.7160458.

Eunjin Jung, Marion Le Tilly, Ashish Gehani, and Yunjie Ge. Data Mining-Based Ethereum Fraud Detection. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 266–273, 2019. doi: 10.1109/Blockchain.2019.00042.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Prentice Hall, 2nd edition edition, May 2008. ISBN 978-0131873216.

Stathis Kamperis. Principal component analysis limitations and how to overcome them. https://ekamperi.github.io/mathematics/2021/02/23/pca-limitations.html, Feb 2021. Accessed: 2022-04-29.

Jakub Karczewski. Machine Learning Models vs. Rule Based Systems in fraud prevention, 2022. URL https://tinyurl.com/machine-learning-vs-rule-based. Accessed: 2022-04-27.

Murat Kaya, F. Gurgen, and Nesrin Okay. An Analysis of Support Vector Machines for Credit Risk Modeling. In *Proceedings of the 2008 conference on Applications of Data Mining in E-Business and Finance*, volume 177, pages 25–33, June 2008. doi: 10.3233/978-1-58603-890-8-25.

Sihem Khemakhem and Younes Boujelbene. Artificial Intelligence for Credit Risk Assessment: Artificial Neural Network and Support Vector Machines. *ACRN Oxford Journal of Finance and Risk Perspectives*, 6(2):1–17, Jan 2017.

Taghi M. Khoshgoftaar, Moiz Golawala, and Jason Van Hulse. An Empirical Study of Learning from Imbalanced Data Using Random Forest. In *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007)*, volume 2, pages 310–317, 2007. doi: 10.1109/ICTAI.2007.46.

Ron Kohavi and Foster Provost. Glossary of terms. Machine Learning—Special Issue on Applications of Machine Learning and the Knowledge Discovery Process. *Machine Learning*, 30: 271–274, January 1998. doi: 10.1023/A:1017181826899.

Ilker Koksal. The Benefits of Applying Blockchain Technology in any Industry. https://tinyurl.com/forbes10042022, October 2019. Accessed: 2022-04-08.

I.B. Kraiem. *Détection d'Anomalies Multiples par Apprentissage Automatique de Règles dans les Séries Temporelles*. PhD dissertation, Universitè de Toulouse-Jean Jaures, January 2021.

D. Kuhn, Dylan Yaga, and Jeffrey Voas. Rethinking Distributed Ledger Technology. *Computer*, 52:68–72, February 2019. doi: 10.1109/MC.2019.2898162.

Max Kuhn and Kjell Johnson. *Applied Predictive Modeling*, volume 1st ed. Springer Science+Business Media LLC, New York, 2013. doi: 978-1461468486.

Max Kuhn and Kjell Johnson. *Feature Engineering and Selection: A Practical Approach for Predictive Models*, volume 1st ed. Chapman and Hall/CRC, Boca Raton, FL, 2019. doi: 978-1138079229.

Thomas Kulnigg. Together on the blockchain: Finding consensus in a decentralised network. https://tinyurl.com/schoenherr-blockchain, January 2020. Accessed: 2022-04-11.

Juan Laborda and Seyong Ryoo. Feature Selection in a Credit Scoring Model. *Mathematics*, 9: 746, 2021. doi: 10.3390/math9070746.

James Le. R Decision Trees Tutorial. https://tinyurl.com/Decision-Tree-in-R, June 2018. Accessed: 2022-04-09.

Erik G. Learned-Miller. Introduction to Supervised Learning, February 2014. URL https://tinyurl.com/Introduction-to-ML. Accessed: 2022-04-12.

Michael Levi. *Understanding cryptocurrency fraud: The challenges and headwinds to regulate digital currencies*. Batten-Corbet-Lucey Handbooks in Alternative Investments. de Gruyter, Berlin/Boston, Germany, December 2021. ISBN 9783110716887.

Gangmin Li. *Do A Data Science Project in 10 Days*. July 2021. URL https://bookdown.org/gmli64/do_a_data_science_project_in_10_days/.

Weizhang Liang, Suizhi Luo, Guoyan Zhao, and Hao Wu. Predicting Hard Rock Pillar Stability Using GBDT, XGBoost, and LightGBM algorithms. *Mathematics*, 8(5):765, April 2020. doi: 10.3390/math8050765.

Andreas Lindholm, Niklas Wahlström, Fredrik Lindsten, and Thomas B. Schön. *Machine Learning - A First Course for Engineers and Scientists*. 2021. URL https://smlbook.org.

Stan Lipovetsky and Michael Conklin. Analysis of Regression in Game Theory Approach. *Applied Stochastic Models in Business and Industry*, 17:319 – 330, 10 2001. doi: 10.1002/asmb.446.

Zachary Lipton. The Mythos of Model Interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16, May 2018. doi: 10.1145/3236386.3241340.

M. Liu and DC. Lin. Commercial bank credit risk assessment model based on support vector machine. *J Xiamen university (natural science edition)*, 44(1):29–32, 2005.

Yukun Liu, Aleh Tsyvinski, and Xi Wu. Common Risk Factors in Cryptocurrency. Working Paper 25882, National Bureau of Economic Research, May 2019.

Meichen Lu. SHAP for explainable machine learning, November 2018. URL https://tinyurl.com/SHAP-explainable-ML. Accessed: 2022-04-25.

Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *CoRR*, abs/1705.07874, 2017. URL http://arxiv.org/abs/1705.07874.

Zee Ma. A Tutorial on Principal Component Analysis. Feb 2014. doi: 10.13140/2.1.1593.1684.

David Martins. XGBoost: A Complete Guide to Fine-Tune and Optimize your Model. http://bit.ly/xgboost_a_complete_guide, May 2021. Accessed: 2022-04-06.

Francisco Melo. *Area under the ROC Curve*, pages 38–39. Springer New York, New York, NY, 2013. ISBN 978-1-4419-9863-7. doi: 10.1007/978-1-4419-9863-7_209. URL https://doi.org/10.1007/978-1-4419-9863-7_209.

Jianyu Miao and Lingfeng Niu. A Survey on Feature Selection. *Procedia Computer Science*, 91: 919–926, 12 2016. doi: 10.1016/j.procs.2016.07.111.

Christoph Molnar. *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. 2021. URL https://christophm.github.io/interpretable-ml-book/.

Patrick Monamo, Vukosi Marivate, and Bhekisipho Twala. A Multifaceted Approach to Bitcoin Fraud Detection: Global and Local Outliers. pages 188–194, December 2016. doi: 10.1109/ICMLA.2016.0039.

Alfredo Motta. Cross Validation done wrong, July 2020. URL https://tinyurl.com/Cross-validation-done-wrong. Accessed: 2022-04-15.

Paulo Moura Oliveira, Paulo Novais, and Luís Reis. *Progress in Artificial Intelligence*. Part II: 19th EPIA Conference on Artificial Intelligence, EPIA 2019. January 2019. ISBN 978-3-030-30243-6. doi: 10.1007/978-3-030-30244-3.

Mimi Mukherjee and Matloob Khushi. Smote-enc: A Novel SMOTE-Based Method to Generate Synthetic Data for Nominal and Continuous Features. *Applied System Innovation*, 4(1), March 2021. doi: 10.1155/2013/694809.

Yuri Musienko. How Secure is Blockchain: Security of Blockchain Technology. https://tinyurl.com/How-secure-is-Blockchain, November 2021. Accessed: 2022-04-12.

Ghayur Naqvi. A Hybrid Filter-Wrapper Approach for Feature Selection. Master's thesis, Department of Technology at Örebro University, 2012. URL http://www.diva-portal.org/smash/get/diva2:567115/FULLTEXT01.pdf.

Harish Natarajan, Solvej Krause, and Helen Gradstein. Distributed Ledger Technology (DLT) and Blockchain. *FinTech Note*, 1, 2017.

Huy Nguyen Duc, Innocent Kamwa, Louis-A Dessaint, and Huy Cao-Duc. A Novel Approach for Early Detection of Impending Voltage Collapse Events based on the Support Vector Machine. *International Transactions on Electrical Energy Systems*, 27, March 2017. doi: 10.1002/etep.2375.

Byungjoo Noh, Changhong Youm, Eunkyoung Goh, Myeounggon Lee, Hwayoung Park, Hyojeong Jeon, and Oh Yoen Kim. XGBoost based machine learning approach to predict the risk of fall in older adults using gait outcomes. *Scientific Reports*, 11:12183, June 2021. doi: 10.1038/s41598-021-91797-w.

Maxwell I. Nye, Armando Solar-Lezama, Joshua B. Tenenbaum, and Brenden M. Lake. Learning Compositional Rules via Neural Program Synthesis, 2020. URL https://arxiv.org/abs/2003.05562. Accessed: 2022-04-27.

Daniel Palmer. South Korean Crypto Exchange Coinbit Seized Over Allegations of Massive Wash Trading, August 2020. URL https://tinyurl.com/south-korea-crypto-exchange. Accessed: 2022-05-01.

Karl Pearson. On Lines and Planes of Closest Fit to Systems of Points in Space. *Philosophical Magazine*, Series 6(2):559—-572, 1901.

Ravi Sankar Pediredla, Vadlamani Ravi, G. Rao, and Indranil Bose. Detection of financial statement fraud and feature selection using data mining techniques. *Decision Support Systems*, 50:491–500, January 2011. doi: 10.1016/j.dss.2010.11.006.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Peng Peng, Yi Zhang, Yinan Wu, and Heming Zhang. An Effective Fault Diagnosis Approach based on gentle AdaBoost and AdaBoost.MH. In *2018 IEEE International Conference on Automation, Electronics and Electrical Engineering (AUTEEE)*, pages 8–12, 2018. doi: 10.1109/AUTEEE.2018.8720764.

Thai Pham and Steven Lee. Anomaly Detection in Bitcoin Network Using Unsupervised Learning Methods. November 2016.

Nathaniel Popper. Bitcoin's Price Was Artificially Inflated, Fueling Skyrocketing Value, Researchers Say, June 2018. URL https://www.nytimes.com/2018/06/13/technology/bitcoin-price-manipulation.html. Accessed: 2022-04-27.

Gil Press. Cleaning Big Data: Most Time-Consuming, Least Enjoyable Data Science Task, Survey Says, March 2016. URL https://tinyurl.com/forbes-cleaning-data. Accessed: 2022-04-16.

Amit Rajan. ISLR Chapter 5: Resampling Methods (Part 1: Cross-Validation), May 2018. URL https://tinyurl.com/Resampling-Methods-Cross-Val. Accessed: 2022-04-15.

Sridhar Ramamoorti and William Olsen. Fraud: The Human Factor. *Financial Executive*, 23: 53–55, January 2007.

Sebastian Raschka. About feature scaling and normalization – and the effect of standardization for machine learning algorithms. https://sebastianraschka.com/Articles/2014_about_feature_scaling.html, July 2014. Accessed: 2022-04-18.

Sebastian Raschka. Principal Component Analysis in 3 simple steps. https://sebastianraschka.com/Articles/2015_pca_in_3_steps.html, January 2015. Accessed: 2022-04-23.

Abderahman Rejeb, Karim Rejeb, and John Keogh. Cryptocurrencies in Modern Finance: A Literature Review. *ETIKONOMI*, 20:93–118, February 2021. doi: 10.15408/etk.v20i1.169111.

Arjan Reurink. Financial Fraud: A Literature Review. *MPIfG Discussion Paper*, 16(5), 2016.

Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?": Explaining the Predictions of Any Classifier. *CoRR*, abs/1602.04938, 2016. URL http://arxiv.org/abs/1602.04938.

Benjamin Ricaud. A simple explanation of entropy in decision trees. https://bricaud.github.io/personal-blog/entropy-in-decision-trees/, Aug 2017. Accessed: 2022-04-03.

Dana Ron. Guest Editor's Introduction. *Machine Learning*, 30:5–6, January 1998. doi: 10.1023/A:1007411609915.

Kate Rooney. Majority of bitcoin trading is a hoax, new study finds, March 2019. URL https://tinyurl.com/Majority-of-bitcoin-is-hoax. Accessed: 2022-04-30.

Saharon Rosset, Uzi Murad, Einat Neumann, Yizhak Idan, and Gadi Pinkas. Discovery of Fraud Rules for Telecommunications - Challenges and Solutions. pages 409–413, January 1999. doi: 10.1145/312129.312303.

Camila Russo. Crypto Exchanges Charge Millions to List Tokens, Autonomous Says, April 2018. URL https://news.bloomberglaw.com/tech-and-telecom-law/crypto-exchanges-charge-millions-to-list-tokens-autonomous-says. Accessed: 2022-04-30.

Marco Sanchez, Luis Urquiza Aguiar, and José Estrada-Jiménez. Fraud Detection Using the Fraud Triangle Theory and Data Mining Techniques: A Literature Review. *Computers*, 10: 121, September 2021. doi: 10.3390/computers10100121.

Iqbal Sarker. Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Computer Science*, 2(160), May 2021. doi: 10.1007/s42979-021-00592-x.

Mihaela van der Schaar. Classification and Regression Trees. March 2017. URL http://www.stats.ox.ac.uk/~flaxman/HT17_lecture13.pdf.

L. S. Shapley. Stochastic Games. *Proceedings of the National Academy of Sciences*, 39(10): 1095–1100, 1953. ISSN 0027-8424. doi: 10.1073/pnas.39.10.1095. URL https://www.pnas.org/content/39/10/1095.

Ryan Sheehy. New Course: Machine Learning with Tree-Based Models in Python, July 2018. URL https://tinyurl.com/ML-with-Tree-based-Models. Accessed: 2022-04-15.

Michael Sheetz. A single anonymous market manipulator caused bitcoin to top $20,000 two years ago, study shows, November 2019. URL https://www.cnbc.com/2019/11/04/study-single-anonymous-market-manipulator-pushed-bitcoin-to-20000.html. Accessed: 2022-04-29.

Terence Shin. Understanding the Confusion Matrix and how to implement it in Python. https://bit.ly/binary_confusion_matrix, May 2020. Accessed: 2022-04-07.

Jonathon Shlens. A Tutorial on Principal Component Analysis, April 2014. URL https://arxiv.org/abs/1404.1100. Accessed: 2022-04-30.

Santosh Shrivastava, Prem Mary Jeyanthi, and Sarbjit Singh. Failure prediction of Indian Banks using SMOTE, Lasso regression, bagging and boosting. *Cogent Economics Finance*, page 8, 2020. doi: 10.1080/23322039.2020.1729569.

Yan-Yan Song and Ying Lu. Decision tree methods: applications for classification and prediction. *Shanghai Archives of Psychiatry*, 27(2):130–5, April 2015. doi: 10.11919/

j.issn.1002-0829.215044. URL https://www.researchgate.net/publication/279457799_Decision_tree_methods_applications_for_classification_and_prediction.

Jerzy Stefanowski and Szymon Wilk. Evaluating business credit risk by means of approach-integrating decision rules and case-based learning. *Int. Syst. in Accounting, Finance and Management*, 10:97–114, June 2001. doi: 10.1002/isaf.197.

Madhumita Sushil, Simon Suster, and Walter Daelemans. Rule induction for global explanation of trained models. *CoRR*, abs/1808.09744, 2018. URL http://arxiv.org/abs/1808.09744.

Vladimir Svetnik, Andy Liaw, Christopher Tong, and Ting Wang. Application of Breiman's Random Forest to Modeling Structure-Activity Relationships of Pharmaceutical Molecules. In *Multiple Classifier Systems*, pages 334–343, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-25966-4.

Armando Teixeira-Pinto. Machine Learning for Biostatistics. https://bookdown.org/content/4818/, Aug 2020. Accessed: 2022-04-04.

Sue Troy and Mary K. Pratt. Distributed Ledger Technology (DLT). https://www.techtarget.com/searchcio/definition/distributed-ledger, June 2021. Accessed: 2022-04-08.

Paul Vigna. Large Bitcoin Player Manipulated Price Sharply Higher, Study Says, November 2019. URL https://tinyurl.com/Large-Bitcoin-Manipulated. Accessed: 2022-04-27.

Juan Orozco Villalobos. Test, training and validation sets, January 2020. URL https://www.brainstobytes.com/test-training-and-validation-sets/. Accessed: 2022-03-23.

Athina Voulgari. Ethereum Analytics. Master's thesis, ETH Zürich, June 2019.

Rohit Walimbe. Handling imbalanced dataset in supervised learning using family of SMOTE algorithm. https://tinyurl.com/Handling-imbalanced-data-SMOTE, April 2017. Accessed: 2022-04-23.

Chen Wang, Chengyuan Deng, and Suzhen Wang. Imbalance-XGBoost: Leveraging weighted and focal losses for binary label-imbalanced classification with XGBoost. *Pattern Recognition Letters*, 136:190–197, 2020. doi: 10.1016/j.patrec.2020.05.035.

Hans-Dieter Wehle. Machine Learning, Deep Learning, and AI: What's the Difference? Conference: Data Scientist Innovation Day, July 2017.

Guangnian Xiao, Qin Cheng, and Chunqin Zhang. Detecting Travel Modes Using Rule-Based Classification System and Gaussian Process Classifier. *IEEE Access*, PP:1–1, August 2019. doi: 10.1109/ACCESS.2019.2936443.

Yanminsun, Andrew Wong, and Mohamed S. Kamel. Classification of imbalanced data: a review. *International Journal of Pattern Recognition and Artificial Intelligence*, 23, 11 2011. doi: 10.1142/S0218001409007326.

Walid Yassin, Hironori Nakatani, Yinghan Zhu, Masaki Kojima, Keiho Owada, Hitoshi Kuwabara, Wataru Gonoiand Yuta Aoki, Hidemasa Takao, Tatsunobu Natsubori, Norichika Iwashiro, Kiyoto Kasai, Yukiko Kano, Osamu Abe, Hidenori Yamasue, and Shinsuke Koike. Machine-learning classification using neuroimaging data in schizophrenia, autism, ultra-high risk and first-episode psychosis. *Translational Psychiatry*, 10(278), Aug 2020. doi: 10.1038/s41398-020-00965-5.

Ivy Yeh and Che-Hui Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, 36: 2473–2480, March 2009. doi: 10.1016/j.eswa.2007.12.020.

Lei Yu and Huan Liu. Efficient Feature Selection via Analysis of Relevance and Redundancy. *The Journal of Machine Learning Research*, 5:1205–1224, December 2004.

L. Zhang, H. Hu, and D Zhang. A credit risk assessment model based on SVM for small and medium enterprises in supply chain finance. *Financial Innovation*, 1(14), 2015. doi: 10.1186/s40854-015-0014-5.

Xiaoqian Zhu, Xiang Ao, Zidi Qin, Yanpeng Chang, Yang Liu, Qing He, and Jianping Li. Intelligent financial fraud detection practices in post-pandemic era. *The Innovation*, 2(4): 100176, 2021. ISSN 2666-6758. doi: 10.1016/j.xinn.2021.100176.

Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems*, 41:647–665, August 2013. doi: 10.1007/s10115-013-0679-x.

Appendix

A Figures

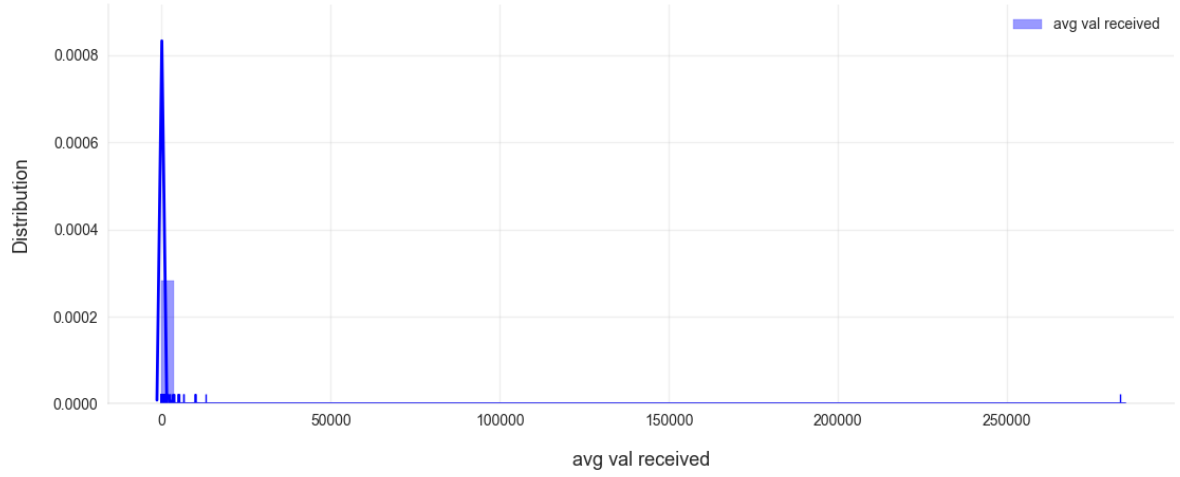


FIGURE A.1: Distribution plot for *avg_value_received*

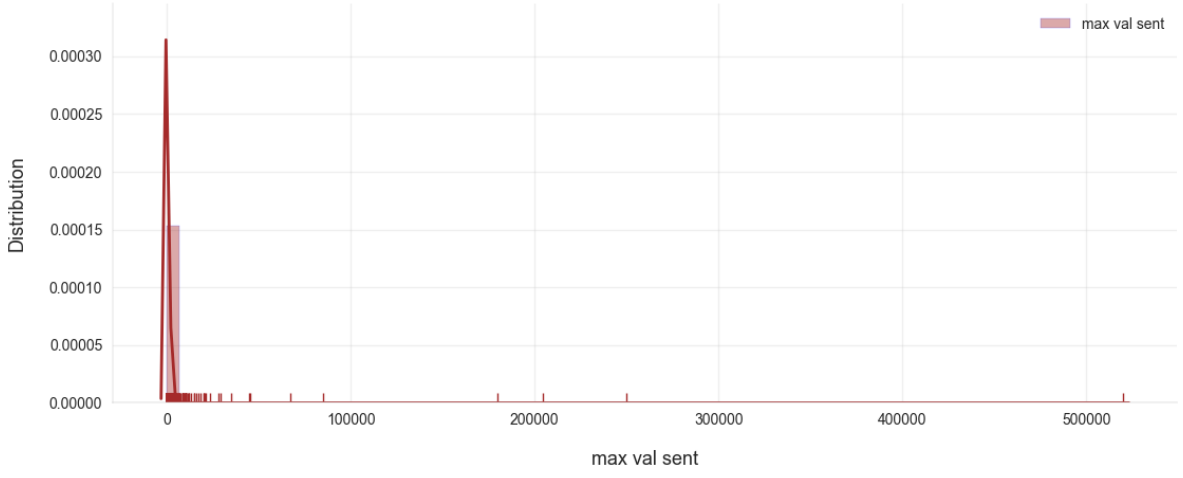


FIGURE A.2: Distribution plot for *max_value_sent*

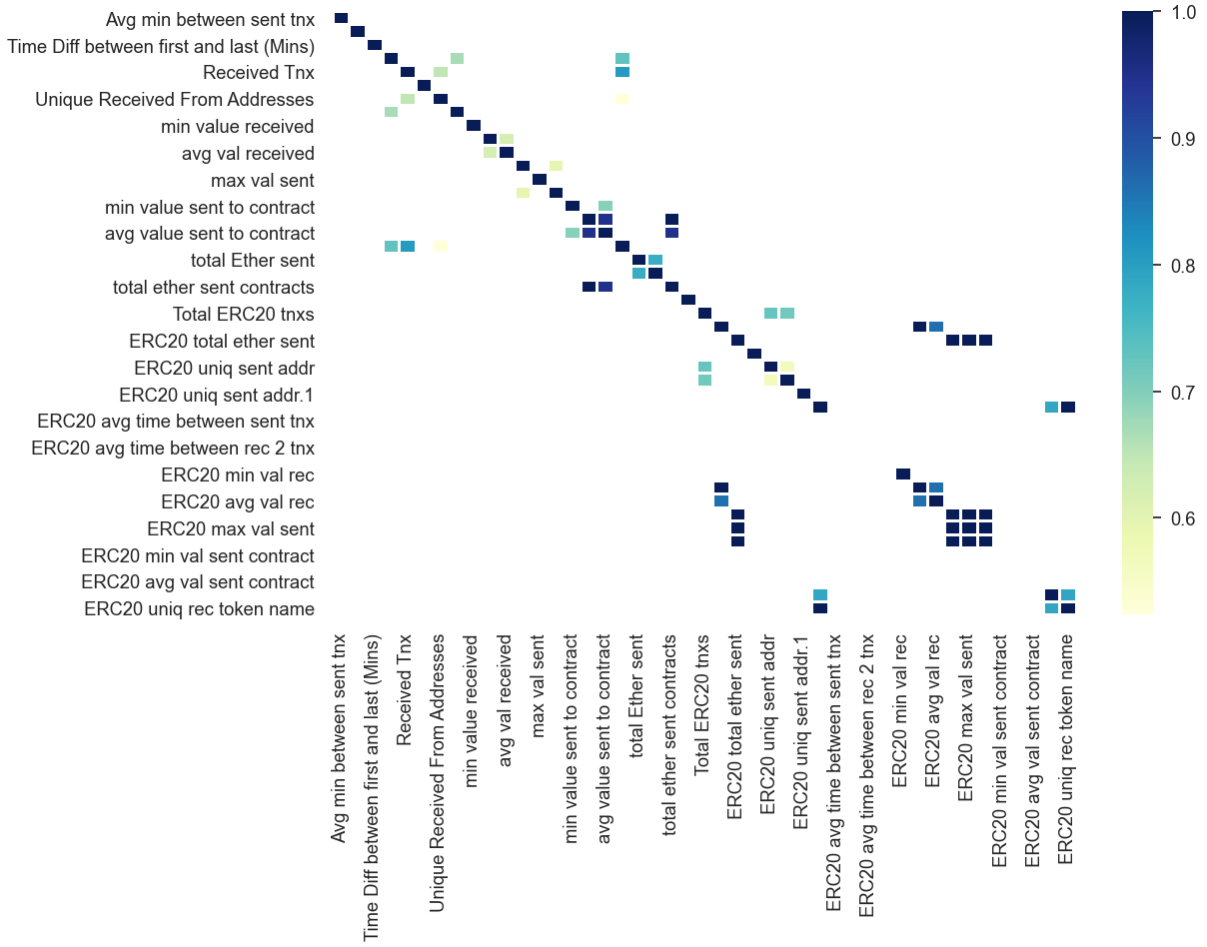


FIGURE A.3: Correlation matrix (heatmap)

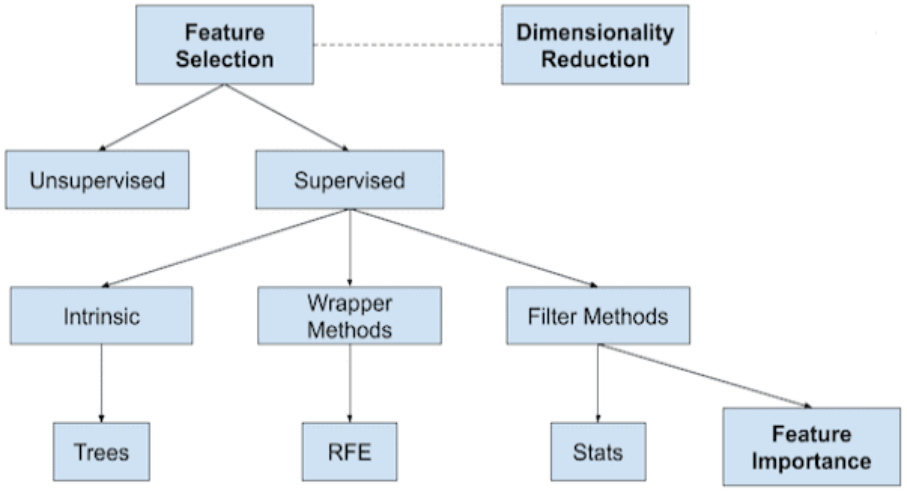


FIGURE A.4: Feature Selection Techniques [Brownlee (2019)]

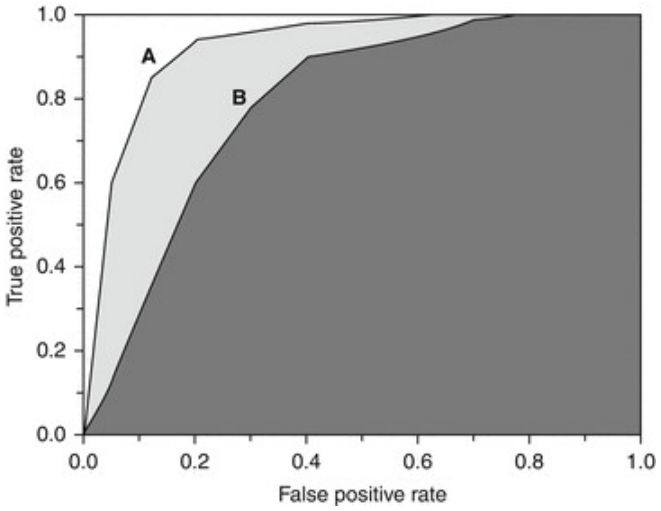


FIGURE A.5: ROC-curve [Melo (2013)]

FIGURE A.6: Bitcoin (BTC) price history

B Tables

Variables	Type	Detail
Address	Categorical	Is the address of the Ethereum account
Avg min between received tnx	Numeric	Average time between received transactions for account in minutes
Avg min between sent tnx	Numeric	Average time between sent transactions for account in minutes
avg val received	Numeric	Average value in Ether ever received
avg val sent	Numeric	Average value of Ether ever sent
avg value sent to contract	Numeric	Average value of Ether sent to contracts
FLAG	Categorical	Answers whether whether the transaction is fraud or not
ERC20 avg time between contract tnx	Numeric	Average time ERC20 token between sent token transactions
ERC20 avg time between rec tnx	Numerical	Average time between ERC20 token received transactions in minutes
ERC20 avg time between sent tnx	Numeric	Average time between ERC20 token sent transactions in minutes
ERC20 avg val rec	Numeric	Average value in Ether received from ERC20 token transactions for account
ERC20 avg val sent	Numeric	Average value of Ether ever sent
ERC20 avg val sent contract	Numeric	Average value in Ether sent from ERC20 token transactions for account
ERC20 max val rec	Numeric	Maximum value in Ether received from ERC20 token transactions for account
ERC20 max val sent	Numeric	Maximum value in Ether sent from ERC20 token transactions for account

Table B1: Variable description of the dataset (1/3)

Variables	Type	Detail
ERC20 max val sent contract	Numeric	Maximum value in Ether sent from ERC20 token transactions for account
ERC20 min val rec	Numeric	Minimum value in Ether received from ERC20 token transactions for account
ERC20 min val sent	Numeric	Minimum value in Ether sent from ERC20 token transactions for account
ERC20 min val sent contract	Numeric	Minimum value in Ether sent from ERC20 token transactions for account
ERC20 min val rec	Numeric	Minimum value in Ether received from ERC20 token transactions for account
ERC20 min val sent	Numeric	Minimum value in Ether sent from ERC20 token transactions for account
ERC20 min val sent contract	Numeric	Minimum value in Ether sent from ERC20 token transactions for account
ERC20 most sent token type	Numeric	Most sent token for account via ERC20 transaction
ERC20 total Ether received	Numeric	Total ERC20 token received transactions in Ether
ERC20 total ether sent	Numeric	Total ERC20 token sent transactions in Ether
ERC20 total Ether sent contract	Numeric	Total ERC20 token transfer to other contracts in Ether
ERC20 uniq rec addr	Numeric	Number of ERC20 token transactions received from Unique addresses
ERC20 uniq rec contract addr	Numeric	Number of ERC20 token transactions received from Unique contract addresses
ERC20 uniq rec token name	Numeric	Number of Unique ERC20 tokens received
ERC20 uniq sent addr	Numeric	Number of ERC20 token transactions sent to Unique account addresses
ERC20 uniq sent token name	Numeric	Number of Unique ERC20 tokens transferred
ERC20_most_rec_token_type	Numeric	Most received token for account via ERC20 transaction
Total ERC20 tnxs	Numeric	Total number of ERC20 token transfer transactions

Table B2: Variable description of the dataset (2/3)

Variables	**Type**	**Detail**
max val sent	Numeric	Maximum value of Ether ever sent
max val sent to contract	Numeric	Maximum value of Ether sent to a contract
max value received	Numeric	Maximum value in Ether ever received
min val sent	Numeric	Minimum value of Ether ever sent
min value received	Numeric	Minimum value in Ether ever received
min value sent to contract	Numeric	Minimum value of Ether sent to a contract
Number of Created Contracts	Numeric	Total Number of created contract transactions
Received Tnx	Numeric	Total number of received normal transactions
Sent tnx	Numeric	Total number of sent normal transactions
Time Diff between first and last (Mins)	Numeric	Time difference between the first and last transaction
total ether balance	Numeric	Total Ether Balance following enacted transactions
total ether received	Numeric	Total Ether received for account address
total Ether sent	Numeric	Total Ether sent for account address
total ether sent contracts	Numeric	Total Ether sent to Contract addresses
total transactions	Numeric	Total number of transactions
Unique Received From Addresses	Numeric	Total Unique addresses from which account received transactions
Unique Sent To Addresses	Numeric	Total Unique addresses from which account sent transactions

Table B3: Variable description of the dataset (3/3)

	count	mean	std	min	25%	50%	75%	max
FLAG	9,841.00	0.22	0.42	0.00	0.00	0.00	0.00	1.00
Avg min between sent tnx	9,841.00	5,086.88	21,486.55	0.00	0.00	17.34	565.47	430,287.67
Avg min between received tnx	9,841.00	8,004.85	23,081.71	0.00	0.00	509.77	5,480.39	482,175.49
Time Diff between first and last (Mins)	9,841.00	218,333.26	322,937.93	0.00	316.93	46,637.03	304,070.98	1,954,860.95
Sent tnx	9,841.00	115.93	757.23	0.00	1.00	3.00	11.00	10,000.00
Received Tnx	9,841.00	163.70	940.84	0.00	1.00	4.00	27.00	10,000.00
Number of Created Contracts	9,841.00	3.73	141.45	0.00	0.00	0.00	0.00	9,995.00
Unique Received From Addresses	9,841.00	30.36	298.62	0.00	1.00	2.00	5.00	9,999.00
Unique Sent To Addresses	9,841.00	25.84	263.82	0.00	1.00	2.00	3.00	9,287.00
min value received	9,841.00	43.85	325.93	0.00	0.00	0.10	2.00	10,000.00
max value received	9,841.00	523.15	13,008.82	0.00	1.00	6.00	67.07	800,000.00
avg val received	9,841.00	100.71	2,885.00	0.00	0.43	1.73	22.00	283,618.83
min val sent	9,841.00	4.80	138.61	0.00	0.00	0.05	1.00	12,000.00
max val sent	9,841.00	314.62	6,629.21	0.00	0.16	5.00	61.52	520,000.00
avg val sent	9,841.00	44.76	239.08	0.00	0.09	1.61	22.00	12,000.00
min value sent to contract	9,841.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02
max val sent to contract	9,841.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05
avg value sent to contract	9,841.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02
total transactions (including tnx to create con...	9,841.00	283.36	1,352.40	0.00	4.00	8.00	54.00	19,995.00
total Ether sent	9,841.00	10,160.92	358,322.71	0.00	0.23	12.49	101.00	28,580,960.89
total ether received	9,841.00	11,638.32	364,204.77	0.00	2.67	30.53	101.00	28,581,590.07
total ether sent contracts	9,841.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05
total ether balance	9,841.00	1,477.40	242,425.42	-15,605,352.04	0.00	0.00	0.04	14,288,636.26
Total ERC20 tnxs	9,012.00	36.26	447.53	0.00	0.00	1.00	3.00	10,001.00

Table B4: Descriptive statistic of the dataset (1/2)

	count	mean	std	min	25%	50%	75%	max
ERC20 total Ether received	9,012.00	129,620,673.24	10,538,584,109.43	0.00	0.00	0.00	100.34	1,000,020,000,000.00
ERC20 total ether sent	9,012.00	13,868,492.61	1,180,389,999.86	0.00	0.00	0.00	0.00	112,000,000,000.00
ERC20 total Ether sent contract	9,012.00	110.94	6,128.63	0.00	0.00	0.00	0.00	416,000.00
ERC20 uniq sent addr	9,012.00	5.64	105.25	0.00	0.00	0.00	0.00	6,582.00
ERC20 uniq rec addr	9,012.00	7.60	81.82	0.00	0.00	1.00	2.00	4,293.00
ERC20 uniq sent addr.1	9,012.00	0.00	0.07	0.00	0.00	0.00	0.00	3.00
ERC20 uniq rec contract addr	9,012.00	4.90	17.25	0.00	0.00	1.00	2.00	782.00
ERC20 avg time between sent tnx	9,012.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ERC20 avg time between rec tnx	9,012.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ERC20 avg time between rec 2 tnx	9,012.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ERC20 avg time between contract tnx	9,012.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ERC20 min val rec	9,012.00	485.61	16,883.28	0.00	0.00	0.00	0.00	990,000.00
ERC20 max val rec	9,012.00	125,252,360.15	10,537,407,457.23	0.00	0.00	0.00	99.00	1,000,000,000,000.00
ERC20 avg val rec	9,012.00	4,346,203.07	214,119,242.01	0.00	0.00	0.00	29.46	17,241,810,275.00
ERC20 min val sent	9,012.00	11,741.26	1,053,567.12	0.00	0.00	0.00	0.00	100,000,000.00
ERC20 max val sent	9,012.00	13,035,935.15	1,179,905,145.26	0.00	0.00	0.00	0.00	112,000,000,000.00
ERC20 avg val sent	9,012.00	6,318,389.13	591,476,414.69	0.00	0.00	0.00	0.00	56,147,560,976.00
ERC20 min val sent contract	9,012.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ERC20 max val sent contract	9,012.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ERC20 avg val sent contract	9,012.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ERC20 uniq sent token name	9,012.00	1.38	6.74	0.00	0.00	0.00	0.00	213.00
ERC20 uniq rec token name	9,012.00	4.83	16.68	0.00	0.00	1.00	2.00	737.00

Table B5: Descriptive statistic of the dataset (2/2)