Master Thesis

Blockchain Clustering with Machine Learning

Jessica Siegenthaler

Supervised by:
Prof. Dr. Fabian Schär
Credit Suisse Asset Management (Switzerland) Professor for
Distributed Ledger Technologies and Fintech
Center for Innovative Finance, University of Basel

Submission date: August 11th, 2020

Abstract

Public bitcoin blockchain stores a lot of historical data around transactions on the blockchain. With unsupervised machine learning, this data was analyzed in order to detect if an algorithm is able to differentiate between shady / illegal owners and clean owners. Data over 11 years has been analyzed to see if patterns exist in the transaction behavior. K-means, PCA and DBSCAN algorithms were used in this thesis. PCA and DBSCAN algorithms were not able to detect a desirable clustering. However, k-means algorithms was able to cluster the data were one cluster contained significant attributes of potentially black wallets, which was compared to an external group of black wallets labeled by the US government. With a recall of 88%, this k-means algorithm performed well in separating black wallets in a cluster.

Keywords: Blockchain, Bitcoin, Bitcoin Anonymity, Blockchain Clustering
JEL: C38, E42, O31

Contents

Acknowledgement

I would first like to thank my thesis advisor Prof. Dr. Fabian Schär for supporting me during this thesis.
My acknowledge also goes to Sein Coray, who supported me with the necessary infrastructure and his knowledge in information technology.
Also I would like to thank Joby Jose and Orkan Sahin for the great discussions around the bitcoin blockchain technology.
In addition, I wish to express my gratitude to Joel Hubeli, Thibaut Vernay and Julia Matas for their valuable inputs around machine learning.
Finally, I thank my family and my friends for their continuous support throughout my thesis. This accomplishment would not have been possible without them. Thank you.

University of Basel

Center for Innovative Finance

Plagiatserklärung

Ich bezeuge mit meiner Unterschrift, dass meine Angaben über die bei der Abfassung meiner Arbeit benutzten Hilfsmittel sowie über die mir zuteil gewordene Hilfe in jeder Hinsicht der Wahrheit entsprechen und vollständig sind. Ich habe das Merkblatt zu Plagiat und Betrug vom 22. Februar 2011 gelesen und bin mir der Konsequenzen eines solchen Handelns bewusst.

Basel, August 11, 2020

J. Siegenthaler

Jessica Siegenthaler

1 Introduction

The blockchain technology promotes transparency, decentralization, immutability and security. An often believed feature is the anonymity discussion. Transacting on a blockchain does not guarantee anonymity. With pseudonymous addresses, anonymity can be compromised if they are reused. Professional service firms started to identify blockchain participants, by using techniques as web-scraping, tracking payments or IP address monitoring (Conti et al. (2018)). Having such an accurate database, which constantly needs to be updated, is time-consuming and not cost-efficient. Therefore, this thesis follows another approach, by identifying the nature of a certain group of people on the bitcoin network. Bitcoin is the first blockchain ever, and with a current market capitalization of $172 billion also the market leader (YahooFinance (2020), Nakamoto et al. (2008)). In order to gather enough data to verify the hypotheses, this blockchain has been selected. Instead of gathering all available information about all users of the bitcoin blockchain ever recorded in the internet, this thesis focuses on applying unsupervised machine learning on the bitcoin data, to find clusters of similar behavior from the users. Behaviors could be to transact in shady / illegal activities, being a trader to benefit from currency variances or by buying goods from companies and many more behaviors. Unsupervised machine learning groups similar data points together in clusters without knowing the target variable. The research question therefor is:
Can unsupervised machine learning clustering identify the nature of the wallets?
The focus lies on wallets with fraudulent, shady behavior. The term wallet used here stands for a user, who is transacting on the bitcoin blockchain. This user (or entity) can have several addresses, but all owners addresses belong to the same wallet. Gathering knowledge in this area can support governments by gaining information in regard to money laundering, terrorist financing or illegal market activities.
In this thesis real data has been analyzed from the bitcoin network. The time horizon of the collected data is between January 16, 2009 and June

23, 2020, with a total of 636’000 blocks.

The thesis is structured as following: The 2nd chapter begins with specific background information around the bitcoin network related to this thesis, so mainly focuses on the anonymity and the address structure. This chapter also informs about the current market situation of private companies, which head into a similar direction as this thesis. Chapter 3 is a literature review about similar papers. Their research focus is in the area of anonymity on the bitcoin network and the usage of machine learning on blockchain data. Chapter 4 guides through the steps of the pre-processing of the data, as the key for all machine learning algorithm is cleared data. This means, the pre-processing part of the data, before using any machine learning algorithm, is essential in order to get correct results. Chapter 5 introduces the unsupervised machine learning part, where the algorithm k-means is implemented as the most popular algorithm, along with a principal component analysis and DBSCAN. After the conclusion chapter, there is an outlook for further research possibilities in this area.

2 Background

This chapter gives some background information on the bitcoin blockchain data, which is key to understand the later parts of this thesis. Furthermore, this chapter will cover a basic understanding of the anonymity in the network, to bring insights on this topic of the thesis.

2.1 Raw data

The whole bitcoin network is publicly available by participating as a full node (downloading the whole history of bitcoin[1] on a machine or cloud), or by viewing over one of the online bitcoin explorers[2]. When downloading the blockchain over a bitcoin client (284 Giga Byte (GB) per June 23, 2020) to get a full node running on the local machine or cloud, the raw data is saved in blk*.dat files. This data in blk*.dat files is in binary format and does not necessarily need to be in the order of the blocks itself. Binary data (values of 0 and 1) can be viewed in hexadecimal format (values from 0 till 9 and from A till F) which then gives the possibility to use American Standard Code for Information Interchange (ASCII) characters as a support for the human readable eye (Sommer (2019)). The data for this thesis is based on this ASCII human readable format and saved as JavaScript Object Notation (JSON) format. Each bitcoin block is a separate JSON file. To skip these steps of converting from binary to hexadecimal to ASCII format, there are several bitcoin explorers which offer an Application Program Interface (API) to get these JSON files for each bitcoin block. One of the popular explorers is blockchain.info with an API call possibility: "www.blockchain.info/rawblock/block_hash", where simply the "block_hash" needs to be replaced by the favorable block height. For the genesis block (the first block of bitcoin blockchain #0) the url link would be: `https://blockchain.info/rawblock/0` with given output as JSON format. The JSON file structure used for this thesis can be found in the appendix A for the genesis block 0.

[1] `https://bitcoin.org/en/download`
[2] `https://www.blockchain.com/explorer`

2.2 Addresses

In bitcoin, with its triple entry bookkeeping system, each transaction is visible together with the corresponding addresses, transferring Bitcoins to another address, without mentioning any names of users. These addresses are used to send Bitcoin from and to people or organizations. An address symbolizes a unique string of digits and characters (around 26 to 42 alphanumeric identifiers) which can be remotely compared to a bank account number. Users can generate this kind of addresses by themselves and as many as they want. There are no costs associated in creating bitcoin addresses (Wang and Liu (2015)). These addresses are mostly based on hashes from the public key, however there are also other possibilities. For example, some address types are based on hashes generated from scripts. A script defines some properties, which get executed during a transaction and decides, if the transaction will be accepted as valid or not. A public key is a result of an intelligent mathematical elliptic multiplication based on a private key. With asymmetric encryption when having the private key as the owner, the public keys can be decrypted but not in the opposite way. There are many research papers in this area, which explains this concept in detail, see one example Koblitz et al. (2000).

Valid bitcoin address types can have one of the following three structures:

P2PKH - Pay-to-Public-Key-Hash: This is the most common bitcoin address and is characterized by beginning with the number 1. There are multiple possibilities to create such an address, as for example over (`https://www.bitaddress.org/`). The first such address is contained in the genesis block #0:

$$\mathbf{1}A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa$$

This kind of address is based on a number of steps to get from the public key to the address. It is explained in Harris and Zero (2019).

P2SH - Pay-to-Script-Hash or SegWit: This type of address is not hashed from a public key but instead of a whole script. They are often called multi-signature / multi-sig addresses. This feature was introduced in January 2012 with the bitcoin improvement proposal 16. The first occurrence was with the following address:

342ftSRCvFHfCeFFBuz4xwbeqnDw6BGUey

How to create such an address can be seen for example on: `https://segwitaddress.org/` (QuantaBytes (2020), Harris and Zero (2019)).

Bech32 (P2WPKH or P2WSH): This Bech32 address was used after the soft-fork Segregated Witness in August 2017 and does not rely on the variant which uses P2PSH address format to enclose SegWit data. Thanks to the segregated witness fork, the transaction malleability problem could be solved and it increased the blocksize (Bitcoin.Wiki (2019)). The Bech32 addresses start with bc1 and are case insensitive (upper and lowercase letters treated the same). This is contrary to the two examples above. An address of this type is:

bc1qvjxyegg05zt73t6dehvzdndlpuamkxywqgf955x8aqg43z62w9lsnd59lm

To check or create such an address, sites like this can be used: `https://segwitaddress.org/bech32/` (Harris and Zero (2019)).

When studying a group of transactions, there is a chance to cross transactions, which contain null data or OP_Returns where not receiver gets the Bitcoins, but instead they are "buried". With null data, people can store information on the bitcoin network forever. One example was in block 325'001, where somebody wrote "hello world" in one of the outputs of a transaction[3]. In an online explorer, this kind of output is marked as OP_Returns. A closer look in the JSON file gives more information. For such transactions the rest of the output addresses, which are valid, were considered for this thesis. On the whole bitcoin blockchain, there has been one empty block, which had a coinbase transaction to an in-

[3] 6dfb16dd580698242bcfd8e433d557ed8c642272a368894de27292a8844a4e75

valid output. For this reason, block 501'726 was not taken into the data volume for this thesis, as no address was involved. The 12.5 BTC which were generated by the system in this block is lost forever. This is one example how Bitcoins can be burned.

2.3 Anonymity

It is highly advised from multiple sources, for users in the bitcoin network, to generate a new address, every time a transaction is made. Later in the thesis it will be clear that many services reuse addresses and therefore generate super-clusters. The reuse of addresses helps identify much of their on-chain activity for external viewers (Harrigan and Fretter (2016)). This thesis will describe in detail how to uncover the anonymity of the users in the network. The subchapter will highlight methods of how users can stay anonymous.
Privacy enabling techniques in bitcoin can be broadly categorised into three classes:

- Peer-to-peer (P2P) mixing protocols (ex. CoinJoin):
  The most popular P2P mixing protocols is CoinJoin and CoinShuffle. CoinJoin adds transactions together without a third party involved in a single transaction. CoinJoin uses multi-signature system to enhance privacy and prevent thefts. Therefore, each input has its own signature and is independent from all other inputs. The anonymity level depends on the number of participants in the CoinJoin transaction. This method is vulnerable to denial-of-service (DoS), intersection and Sybil attacks. Sybil attacks are, when nodes illegitimately claim multiple identities (Douceur (2002)).

  CoinShuffle is a decentralized protocol for coordinating CoinJoin transaction through cryptographic mixing protocols. Their advantage is internal unlinkability, robustness against DoS attacks and theft resistance. On the other hand, this technique has a lower anonymity level. There are further P2P protocols which can be viewed in Conti et al. (2018).

- Distributed mixing networks (ex. Tumbler)
  The principal of distributed mixing networks is built as follows: The user transacts his/her Bitcoin to a third-party mixing service, and receives the Bitcoin amount minus service fees back from the mixing service. The received Bitcoins were originally submitted by some other users. Due to this service, it will be difficult to match where the funds went, once submitted to the mixing service. This kind of strategy provides a strong anonymity from external observers. Example of such services would be MixCoin, BlindCoin and TumbleBit. Many mixers are available through the Tor network (The Onion Router), to prevent a linkage to the IP address (Conti et al. (2018)). Tor network is a browser dedicated to protect the privacy of the users (Jardine (2015)).

- Altcoins (ex. ZeroCoin)
  With invention of the blockchain through bitcoin, there has been a wave of other new cryptocurrencies being created on the basis of the decentralized P2P network. These other currencies inspired by bitcoin are collectively called altcoins. Altcoins gained on popularity because of multiple reasons. Ethereum enabled the usage of smart contracts, where an automated script is executed, when coins are paid. Blockchains for example Hyperledger Fabric were built for private groups.
  Altcoins like ZeroCoin, ZeroCash and Zcash are cryptographic extensions to bitcoin, which are unlinkable and untraceable transactions. They are built on the zero knowledge proof protocol. They provide assurance, that internal connections are not possible and provide theft- and DoS resistance. However, they rely on a trusted setup, and blockchain pruning is not possible. Pruning a blockchain means that only certain information of the blockchain are saved in order to work around the scalability issue (Matzutt et al. (2020)).
  A famous altcoin for anonymity is Monero blockchain, which is based on a CryptoNote protocol, and improves user privacy by ring signatures. There is a slight chance that transaction linking could be achieved by leveraging the ring signature size of zero, to-

gether with an output merging. For further altcoin differentiating see Conti et al. (2018).

Techniques or alternatives how to ensure anonymity when transacting on the blockchain is not completed with the list above, and over time new ways are developed. Nevertheless, as the bitcoin blockchain stores all transaction historically a great deal of anonymity can still be uncovered, which this thesis is about.

2.4 Private companies with user data

From an economic point of view, private companies could have an interest in knowing, who is behind which address on the bitcoin network. Hence, several private companies have their business model in selling labeled data about the bitcoin network users. However, it is strongly assumable that large centralized players on the bitcoin network, for example exchange platforms and wallet services, would be capable of identifying and observing a large amount of user activity as well. Here an overview regarding the most present firms in this area:

- The Elliptic company is a midsized company with 51-200 employees, based in London and founded in 2013 (Elliptic (2020)). They are specialized in fighting financial crypto crime, and won the "2020 Technology Pioneer" award from the World Economic Forum. With web-scraping in the clear and the dark web they search for publicly available information to corresponding cryptocurrency addresses. They work with the Federal Bureau of Investigation (FBI) and Central Intelligence Agency (CIA) (Computerworld (2019)). A published research paper documented supervised machine learning with the Elliptic labeled data set (Weber et al. (2019)).

- The Chainalysis company with headquater in New York was founded in 2014, is a midsized company with 51-200 employees. They established a partnership with one of the major cryptocurrency trading platforms, Binance, in 2018 and have cooperations with the FBI, the Drug Enforcement Administration (DEA) and the Internal Revenue Service (IRS). Furthermore, they offer cryptocurrency investigation and compliance solutions for global law enforcement agencies (Chainalysis (2020)). Due to the labeled data set which they provide for customers, some research papers have used this data for supervised machine learning, as per example Harlev et al. (2018), Yin et al. (2019) and Nie et al. (2017).

- The CipherTrace company from California was founded in 2015 and is a small company with 11-50 employees. CipherTrace develops forensic services on the blockchain with intelligence threat solutions, called Maltego. By tracing transaction flows, CipherTrace supports cryptocurrency exchanges, banks, investigators, regulators and digital asset businesses. They assure compliance with regulatory anti-money laundering requirements. Also, the anti-money laundering report has become an authoritative industry data source[4] (Cipher Trace (2020)).

- Bitfury was founded in Amsterdam in 2011 and with 501-1000 employees is one of the larger companies in this offering (Crystal Blockchain (2020)). Bitfury is a full-service blockchain technology company. One service which goes into the anonymity aspect is the web-based software tool Crystal, which supports financial institutions and law enforcement to manage investigations on blockchain (Bitfury (2020*a*)). In April 2020 the Crystal Blockchain analytics team published an international bitcoin flow analytics report, which shows countries who received and sent the most Bitcoins in their exchanges. In the first quarter 2020, the volume of bitcoins transferred between exchange platforms was $15 billion (Bitfury (2020*b*)).

[4] `https://ciphertrace.com/resources/?reports`

3 Related Research

In this chapter related work has been analyzed in two main areas: anonymity on the bitcoin blockchain and address clustering. Both topics are deeply related and therefore relevant for this thesis.

Zola et al. (2019) wanted to detect illegal activities on the bitcoin blockchain network with machine learning. They used a method to attack bitcoin anonymity through entity characterization with implementing a cascade of classifiers. With this characterization, outgoing classification results could be grouped and used to enrich a final classification. Machine learning models Adaboost, RandomForest and Gradient Boosting were applied. With a global accuracy score of 99.68%, their algorithm classifies very precisely.
The researcher Chawathe (2019) considered the bitcoin blockchain data, while applying well-known clustering algorithms like k-means and tried address merging based on patterns in the blockchain. As an evaluation metrics, he used the Mahalonobis distance, which helped to evaluate the quality of the output clustering algorithm.
The research team Huang et al. (2017) focused on clustering the bitcoin nodes and invented a new clustering method "Behavior Pattern Clustering" (BPC). They extracted the sequences according to the transaction amount changing over time. The research was conducted on 1'321 nodes, and their algorithm out-performed the state-of-the-art algorithm in the behavior pattern clustering tasks.
Researchers Ermilov et al. (2017) defined that they can identify owners by two ways, behavior patterns in the blockchain data and publicly available information from off-chain sources. To gain knowledge from patterns, they define different heuristics like change heuristics of the output. This part will be described in detail in the later chapter 4.1. For analyzing the off-chain information, they again have two ways: a) passive approach by using web-crawling of public forums together with user profiles and b) active approach, where they perform manual analysis of bitcoin companies and data actualisation procedures. A greedy additive

clustering method was used by the research team for their analysis.
Maesa et al. (2016) considered blockchain data till December 2015 and used them in Protocol Buffers format from Google. They confirmed the small-world hypothesis, where everything is strongly connected and analyzed, if there is a wealth concentration.
The research team Harrigan and Fretter (2016) used publicly available information to cluster different groups on the bitcoin network. They used data up till February 2016. In their papers conclusion was describe that especially super-clusters are primary targets for identification attacks.
ShenTu and Yu (2015) focused on both sides of the anonymity, the identification and the anonymization of users from a qualitative aspect.
The researchers Fleder et al. (2015) tried to link bitcoin public keys to real people, either definitely or statistically. Furthermore, they created a transaction graph to find and summarize activities of both known and unknown users. Raw bitcoin data until December 13, 2013 was considered. They extracted the raw data from a full node and parsed it into LevelDB. For the web-scraping they used the Python package Scrapy.
Kondor et al. (2014) researchers analyzed the transaction graph from bitcoin data until May 2013. They found that the wealth of rich bitcoin users increased faster than the wealth of other nodes.
Their (Meiklejohn et al. (2013)) approach was to open accounts and make purchases from a broad range of known bitcoin merchants and service providers to label their identity afterwards. Additionally, they collected known or assumed addresses they found in the web. To analyze the bitcoin data, this team downloaded a full node by April 13, 2013 and later uploaded the raw data into a PostgreSQL database.
With raw bitcoin data until May 2012, Ron and Shamir (2013) discovered that the network contains a huge number of small transactions, while a subset of transactions has moved a large amount of money. Following this observation, they focused on the large transactions in order to detect the ways, how these amounts are bundled or dispersed.
Researchers Reid and Harrigan (2011) paper was one of the first to analyze the anonymity related to bitcoin, which is why their paper is one of the most cited in this area. The raw data which was considered for the

research was until July 2011. The researchers investigated bitcoin thefts with external information and techniques like context discovery and flow analysis. They separated their analysis in transaction network and the user network. They built a partial user directory to associate bitcoin users.

The literature knowledge was taken into account for this thesis. Chapter 4 will guide through the approach of the pre-processing stage, which has been modified to previous research in order to generate an accurate clustering for a later stage. While some papers used labeld data, others focused on a specific timeline in bitcoin history. This thesis includes all blocks accurately until June 23, 2020. Beyond the scope of this thesis was an IP address mapping, to get the knowledge who is behind which wallet. The focus is on which wallet owners behave in a similar way to detect patterns.

4 Pre-processing

To use machine learning on the bitcoin data, the data needs to be preprocessed in a way, that existing algorithms can use the underlying data and detect patterns. As blockchain data is stored on multiple nodes due to the distributed ledger system, and does not contain a standard database table as structured data, this process has to be completed first. Therefore, an architectural concept has been styled in figure 1.

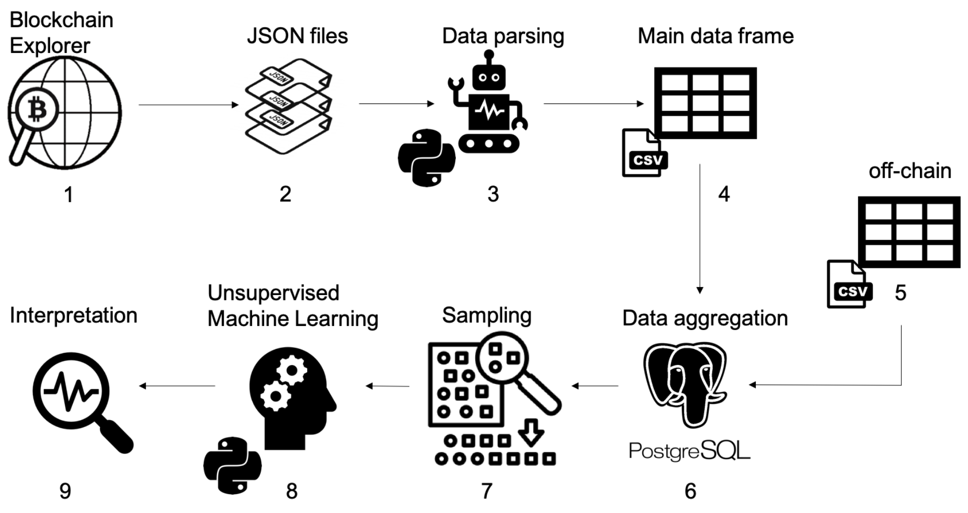


Figure 1: Architecture for data pre-processing with analyzing

All the computations have been done on a machine with 32 CPUs Intel(R) Xeon(R) CPU E5-2600 @2.20 Ghz, 128 GB RAM DDR3 1067 Mhz ECC and 6TB Disk SATA from Western Digital. The software system was Ubuntu 18.04.4 LTS. PostgreSQL v10.12 was used to manage stored data. The various classifier models were implemented and evaluated using Python's scikit-learn library (version 0.23.1).

The first step was to implement an API to the blockchain.info website, to download all 636'000 bitcoin blocks, which was mined up till June 23, 2020[5]. This could be done by a short PHP script (acronym stands

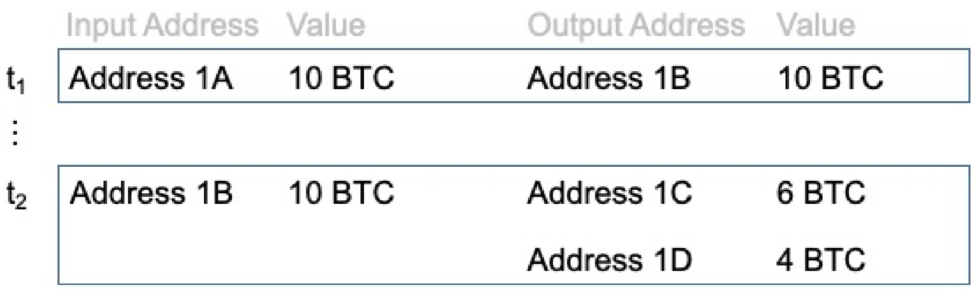
[5] `https://www.blockchain.com/btc/block/636000`

for hypertext preprocessor) which downloaded all the 636'000 blocks in approximately a week. Figure 1 visually represent this process from step one to step two. All 636'000 JSON files sum up to 1.1 TB. As a comparison to download a full node from bitcoin core blockchain the total size per June 23, 2020 was 284 GB (Blockchain.com (2020)). At the time of this process 31 blocks[6] were corrupted and could not been retrieved from the blockchain.info API. For this reason, these blocks have been manually filled with the block information in the respective structure in order to have a complete data set. These 31 blocks are located in a short period between July and September 2015.

After extracting the data, the JSON files had to be processed in order to turn them into a table structure for analyzing purposes (figure 1 step 3). To do so, some heuristics had to be defined to make a clustering process in the machine learning part easier. The clustering part will be explained in chapter 5. As the main purpose of this thesis is to find patterns in the bitcoin data, it is interesting to know which players are on the blockchain network. For that, some bitcoin blockchain theory needs to be highlighted first. As already mentioned, the blockchain participants names are anonymous, but their respective address is publicly visible. Furthermore, some patterns are constructible in the transaction structure itself. Each bitcoin transaction can contain 1 to N input addresses, as well as 1 to N output addresses. One person can own one or many addresses and has the freedom to create as many new addresses as desired. Every transaction creates a new unspent transaction output (UTXO). The age of such UTXOs gives information, in which block this UTXO was first created and therefore can lead to more pattern analysis. When making bitcoin transactions as an input owner, one or several UTXOs will be used. When the total amount of the UTXOs are exceeding the desired transaction amount, some rest amount needs to be returned to the input owner. Always a whole UTXO (which can contain any amount, depending on the previous transaction) needs to be inserted as input in

[6] 364675 364797 364799 364807 364831 364837 364845 364847 364886 364946 364951 364960 364965 364968 364972 364990 364994 365015 365026 365047 365049 365063 365065 365068 365074 365077 367851 367853 367859 373979 374940

a transaction. To handle such a partial pay-out to some other bitcoin participant, the rest of the unused UTXO amount stays in the owners possession. It needs to be specified to which addresses the partial bitcoin value flows. To make a short example see figure 2.



	Input Address	Value	Output Address	Value
t_1	Address 1A	10 BTC	Address 1B	10 BTC
⋮				
t_2	Address 1B	10 BTC	Address 1C	6 BTC
			Address 1D	4 BTC

Figure 2: Two connecting bitcoin transaction with an UTXO, neglecting transaction costs

In time period t_1 address 1A sends 10 BTC to address 1B. In time period t_2 the owner of address 1B wants to send address 1C 6 BTC. To do that, the owner must insert the whole UTXO value of the 10 BTC which was received on t_1 into the transaction. As the rest of the 10 BTC should stay in possession of address 1B owner, he/she needs to insert a second output in the same transaction indicating to which address this money should flow. This is usually a newly created address, as it is strongly suggested by the bitcoin community to do so, in order to stay more anonymous. However, the same address can be inserted again to return this 4 BTC to address 1B owner, who is also address 1D owner. In this example transaction fees have been neglected.
With this knowledge seven rules have been defined, and each transaction should fall into one of these seven rules. The goal is to map the same owner of multiple addresses together. In the example in figure 2, address 1B and address 1D are mapped as the same owner. The term wallet is used for this. One wallet can have multiple addresses and each wallet gets a unique identification (ID) number. Consequently, when the JSON files will be processed, every transaction and address gets registered in a data frame with the corresponding walletID. In the end this will lead to the structured table for machine learning.

Transaction hash	Bitcoin address	WalletID	Time	Value	Rule

Table 1: Main table

In the next subsection the rules for this walletID generation will be explained.

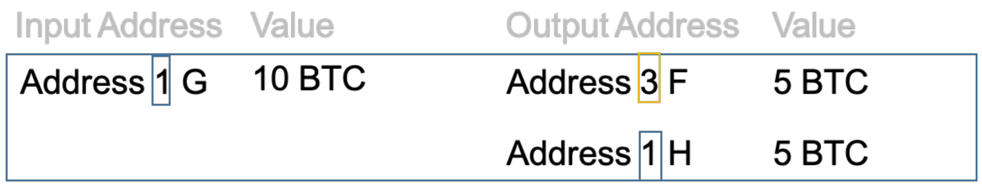
4.1 Rule heuristics

Rule 0 - Coinbase transaction
In each block the first transaction is from the system generated transaction, where the miner gets his/her block reward. A miner can create Bitcoins by packing and verifying new transactions into a block while using computation power and publishing the newly mined blocks on the network. For this effort the miner gets the block reward paid out in the first transaction of the block (Wang and Liu (2015)). So there is no input address (as the bitcoin system created these Bitcoin values) but the transaction contains output address(es) in this coinbase transaction. Newly generated Bitcoins belong to the miner, who mined the block. Therefore, this rule takes the output addresses and cross-checks them with the newly created database table (main table 1) which grows after each transaction. If this output address from the miner has already been used in a previous transaction, meaning that the owners walletID already exists, then this address will be labeled with the same walletID and added in the data frame. Otherwise a new walletID will be assigned to this output address.

Rule 1 - A 1:1 transaction
This transaction type contains one input and one output address. Each input address in the system has by definition some output in a previous transaction, as the bitcoin system is historically built. Therefore, the input address can be easily mapped to the existing walletID of the main data frame. The output address is compared to the main data frame, and if it exists, the same walletID will be copied. Otherwise a newly created walletID is mapped to this output address.

Rule 2 - A 1:2 transaction
In the frequently occurring situation, where the UTXO needs to be splitted, a partial amount goes back to its original owner. By paying partially to the creditor with the UTXO some change to the owner of the input address occurs. In theory, different versions of these "shadow" addresses can be identified. Shadow address means a change of the Bitcoin from the original owner and this should land in the same cluster. Androulaki et al. (2013) state the heuristic, in the case that the transaction has two outputs: one address, which already exists in the main data frame and one address which is newly created. They say this is an indication, that the new address belongs to the input owner who created a new address to return the change value. The other output owner (in this example the creditor) already used his/her address before. Although this approach has been confirmed by Chawathe (2019), there has been no proof that seven years after their analysis, the same heuristic still is true. Concluding on this, the heuristic approach has been taken out of consideration.

Another possibility, which was mentioned in the paper Kalodner et al. (2017) has been graphically displayed in figure 3. This transaction contains one output address starting with the number 3 and one output address starting with the number 1. The address starting with 3 is a P2SH and therefore most likely a vendor, which leads to the assumption that the other output address can be linked as the shadow address (change from the input owner). The paper introducing this heuristic is newer (Kalodner et al. (2017)), however the usage of P2SH addresses has increased, and now normal users (not only vendors) use the same address starting with the number 3. For example, one of the best promoted hardware wallets called Trezor (invented by Satoshi Lab in 2014) is only using addresses starting with 3 since 2017 (Mackay (2019)). Considering this uncertainty about the user of such addresses, this heuristic has not been implemented either, as there is no clear majority for P2SH being only vendors.



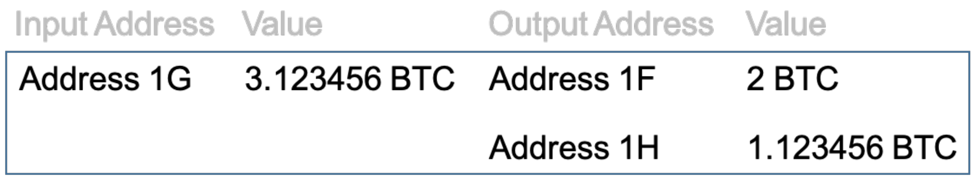
Input Address	Value	Output Address	Value
Address 1 G	10 BTC	Address 3 F	5 BTC
		Address 1 H	5 BTC

Figure 3: Bitcoin transaction to a P2SH address, neglecting transaction costs

Another way to recognize the shadow address was introduced by Spagnuolo et al. (2014), who noted, that there has been a bug in the official bitcoin core system, where the change address has been in the first position and the other address in the second. Bitcoin developers recognized this anonymity issue and fixed it on February 3, 2013. This left the first 218’995 blocks with this edition. Spagnuolo et al. (2014) state that only 6.8% of the shadow addresses have been provably in the second slot of the two output transactions. This seems to be very accurate, which is why for this period, between the genesis block and the last block before the fix, the first output address has been considered as a change address. The document fix was CVE-2013-2273 and stated by the developer team "bitcoind and Bitcoin-Qt before 0.4.9rc1, 0.5.x before 0.5.8rc1, 0.6.0 before 0.6.0.11rc1, 0.6.1 through 0.6.5 before 0.6.5rc1, and 0.7.x before 0.7.3rc1 make it easier for remote attackers to obtain potentially sensitive information about returned change by leveraging certain predictability in the outputs of a bitcoin transaction" (Bitcoin.Wiki (2020)). As the documentation on the internet and the research paper from Spagnuolo et al. (2014) does not exactly cite the exact time of the fix, the blocks until end of January 31, 2013 have been considered to be affected by this bug, which includes blocks till height 218’995. Therefore, always the first output address has been mapped with the walletID of the input address. After block 218’995, a new way of recognizing the change address had to be implemented which includes the following steps:

1. First the output addresses are compared with the input address. If one matches, it is clear that this is the change of the input address. Leading the change address to get the same walletID as the input address.

2. Should that not be the case, the next check would be to compare both values of the outputs. An example is shown in figure 4: One value of the two outputs has no partial numbers after the forth decimal point and the other output value has a decimal number after the 4th place. In this case the heuristic assumes, that the partial Bitcoin value belongs to the input owner, which has a partial change. Most transactions spend whole Bitcoins to another address, but then have some change (which can be seen if it has an amount after the 4th decimal point). In case both values have numbers after the 4th decimal point, or alternatively if for both addresses this is not the case, then step 3 is followed. Therefore, to come back to figure 4, address 1H (output address) most likely belongs to the same owner as 1G (input address).



Input Address	Value	Output Address	Value
Address 1G	3.123456 BTC	Address 1F	2 BTC
		Address 1H	1.123456 BTC

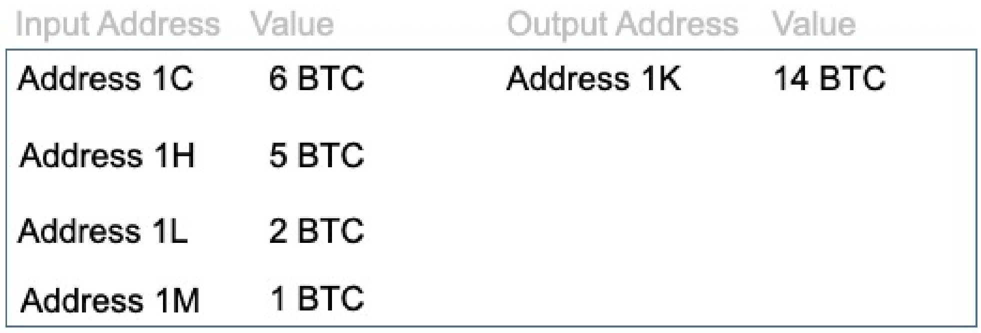
Figure 4: Bitcoin transaction with decimal values, neglecting transaction costs

3. Here both output addresses are checked with the main data frame. If they exist, the walletID is copied from there, if they are newly created addresses, then a new walletID is created.

Rule 3 - A 1:N transaction

Here the output addresses are compared to the main data frame and if they already exist (have been used in the past), then they get the same walletID, otherwise a new walletID is created.

Rule 4 - A N:1 transaction
In the literature Ron and Shamir (2013) state that multiple sending addresses can be reasonable assumed to have the same owner. This heuristic is underlined as well by two other literatures Meiklejohn et al. (2013) and Androulaki et al. (2013) which say, if two or more addresses are inputs of the same transaction, they are controlled by the same owner. As with a high likelihood the input owner of the addresses is the same person, who authorized the transaction and had access to the corresponding private keys (Hirshman et al. (2013)). This does not need to be necessarily true with CoinJoin transactions or mixed services. However, the probability that different input owners only pay to one other person is rather small. Therefore, this heuristic has been implemented in this thesis, as in practice different users rarely contribute in a single, collaborative transaction (Conti et al. (2018)). All input addresses have been mapped to the same walletID, which led to updating the main data frame, as the input addresses may have already been noted to different walletIDs. Graphically displayed in figure 5, addresses 1C, 1H, 1L and 1M belong to the same owner and will be mapped to the same walletID.



Input Address	Value	Output Address	Value
Address 1C	6 BTC	Address 1K	14 BTC
Address 1H	5 BTC		
Address 1L	2 BTC		
Address 1M	1 BTC		

Figure 5: Input addresses mapped to the same owner

The output address is checked with the main data frame and if the address already existed, it gets the same walletID, otherwise a new walletID is created.
Especially when transaction fees are low, Bitcoin owners merge many of their addresses into a single address, for usability and cheaper transac-

tion fees in the future. Bitcoin transaction fees are higher, if multiple addresses are involved. In this use case, the output address should get the same walletID as the input address. However, this use case could not be verified by literature research, and was therefore not implemented to avoid wrong clusters.

Rule 5 - A N:2 transaction
This case is similar to rule 4 when considering the input addresses. All input addresses have been linked to the same walletID. For the two output addresses, the same applies like in rule 2. The repetition of the steps from rule 2 are listed below:

1. Check if block is lower or equal to height 218'995.
2. Check if output address is same to one of the input addresses.
3. Check if only one output address has a value after the fourth decimal place or lower.
4. Map existing addresses with same walletID from main data frame or create new walletID.

Rule 6 - A N:N transaction
Here no heuristic can be used, because it is not clear how many involved parties there are in this transaction. For this rule 6, the input addresses have been matched with the historical addresses and received the respective walletID. The other output addresses were compared to the main data frame, and, if not yet existing, received a new walletID.
A real life transaction with the same amount of inputs and outputs as described for each of this seven rules above has been saved in the appendix A.

After setting up rules and heuristics, a Python script ran through all 636'000 blocks and parsed the data into a data frame format which was saved as a CSV file. This file contained 2.7 billion rows. The code performed 24 hours on the server. Attributes like transaction hash, bitcoin address, walletID, time, value and which rule have been saved in the main table, as already displayed in table 1. The first few instances of running the code took very long (several hours for only 100 blocks). The newly generated lines not only needed to be appended in the main data frame, but also had to be constantly updated with walletID mappings as well. To get around this problem, the Python code had to be modified, where the first script processed all the JSON data and saved dependencies in another data frame. In a second procedure a new script updated the main data frame with support of the dependency data frame. This technique lead to very fast processing and enabled the end data frame for the next step.

Rule	Transaction occurrences	Percentage of rule
Rule 0, coinbase txid	18'410'043	3.4 %
Rule 1, 1:1 txid	43'536'302	8 %
Rule 2, 1:2 txid	316'271'452	58.4 %
Rule 3, 1:N txid	32'099'583	6 %
Rule 4, N:1 txid	21'660'855	4 %
Rule 5, N:2 txid	103'171'731	19 %
Rule 6, N:N txid	6'514'357	1.2 %
Total transactions	541'664'323	100 %

Table 2: Distribution of rules in the bitcoin blockchain on transaction (txid) basis

The described heuristic rules from above have the following distribution as displayed in table 2 along the bitcoin data set. The numbers from the table are distributed on transaction level. With 542 million transactions per June 23, 2020, the most prevalent transaction rule was rule 2 with a 1 to N transaction. The rule least likely to occur was a transaction with N inputs and N outputs, marked as rule 6. If the distribution of the rule heuristics would be measured on the basis of addresses, then the percentages of rule 3 until rule 6 would increase, as they have more addresses to count for a rule. The main data frame contains 2.7 billion rows of addresses and has a distribution of rule 0: 3.5%, rule 1: 3.2%, rule 2: 34.1%, rule 3: 12.2%, rule 4: 11.1%, rule 5: 26.2% and rule 6: 9.7%. This percentages compared to table 2 shows, that rule 3, rule 4, rule 5 and rule 6 have a percentage increase whereas the other three rules have lower percentages. This can be explained that the mentioned four rules with percentage increase all have a N number of addresses involved. Because every address for every transaction is listed in the data frame with the 2.7 million rows, the overall rule's involvement goes up when a transaction with N in- or outputs is involved.

4.2 Off-chain analysis

Parallel to the data parsing, research for off-chain information has been gathered as indicated in figure 1 point 5. This off-chain information brings value to the bitcoin transactions, by showing what kind of activities they have been involved in.

This additional expansion of the data should give abundant information to the machine learning part in a later stage. The hypothesis is, that in some constellation of the dimensions, a cluster would appear, which would imply that the wallet has been blacklisted, either by illegal activities or by suspicious behavior. In many research papers a web-crawling app was built for this, which mapped published bitcoin address in the internet with associated names or entities. Such a process would need additional checks on eligibility of the results, which is why this method has not been included in the thesis. However, a good reference page is Janda (2020), where services/entities are mapped with the bitcoin address: `https://www.walletexplorer.com/`. The owner of this page provided an API access to gain this data. It turned out that the amount of data which needed to be processed exceeded this thesis. Additionally, this information could not be proven with external sources, as most information gathered was from own small transactions with these services by the website host. Information from this website were not incorporated into the thesis.

For concentration purpose, the research on the off-chain data had to be focused on a few categories as follows:

- Gambling
- Mining
- US blacklisted wallets
- Whale wallets

These categories have been selected, as they contribute with the off-chain analysis into a quite significant category and the underlying data found on the internet is trustworthy. In the category gambling, 20 addresses have been used from the famous bitcoin gambling platform SatoshiDice (2020), which uses the same addresses since 2017. Furthermore, the most used gambling addresses starting to appear in 2013 on the blockchain were added from the list Cdecker (2017) and from the conference Burks et al. (2017).
The category mining is important, as in every single block on the blockchain, one miner has to be involved at least once. 93 addresses have been selected from the github repository of BTC.com (2020). BTC.com is an online bitcoin explorer and offers different statistics on the bitcoin network like the mapping of several miners, as F2Pool, Poolin, etc.
The FBI is tracking wallets with illegal background as well. However, these lists are not publicly accessible. In substitution of the FBI blacklist, the US Office of Foreign Assets Control (OFAC) has a sanction list of people, which must not be traded with. This is also applicable for their cryptocurrency addresses. The "Specially Designated Nationals And Blocked Persons List" (SDN) contains 33 bitcoin addresses which have been added to the category US blacklisted wallets (U.S.Government (2020*a*)). In addition, 61 addresses have been published in an official press release by the United States district court (U.S.Government (2020*b*)). The last category called Whale wallets has been included in the analysis. A bitcoin account with a value of 1'000 BTC or higher is considered a Whale. Therefore,

addresses above 1'000 BTC have been downloaded from the website Bitinfocharts (2020), which summed up to 1'200 addresses defined as Whale accounts.

All addresses are compared with the total overall main list to get the transaction strings. Each transaction containing one of the above mentioned addresses, will get an indication in this category. The table 3 shows the categories and the amount of transactions that are affected by them.

Category	Count of Addresses	Transactions affected
Gambling	20	10'145'481
Mining	93	676'259
US blacklisted wallets	94	10'489
Whale	2100	2'926'887

Table 3: Off-chain addresses mapped to transactions on bitcoin network

Surprisingly the category mining contains only 676'259 affected transactions in the whole population and the US blacklisted wallets contained even less with 10'489 affected transactions. Therefore, these two categories have been excluded for the machine learning part. In general, most machine learning algorithms are quite sensitive to features (dimensions) which do not contribute to additional variation of the model. Therefore, less features which explain the model properly are preferred. Since the idea for this thesis is to detect clusters which could lead to some sort of illegal activities, the US blacklisted wallets will later be compared with the clusters generated from the algorithms. With this approach, similarities can be detected.

4.3 Data aggregation

After the data parsing, the main data frame and the off-chain information need to be combined and aggregated as displayed in step 6 of the architecture from figure 1. As the main data frame contains 320 GB of data, a database software which can handle such a value of data needs to be used. Since PostgreSQL is open source, user-friendly and widely used, this database was chosen. PostgreSQL is a structured database software and uses a syntax similar to the Structured Query Language (SQL) (Drake and Worsley (2002)). Seven steps were processed in the PostgreSQL before the final data output was available for the machine learning part.

1. The main data frame with 2.7 billion rows has been loaded into a table in PostgreSQL. The column transaction hash displays the transaction number. Since a transaction can contain several addresses, a transaction is listed multiple times with the corresponding address in the next column. The walletID was pre-given in the first part of the pre-processing when converting from a JSON to a CSV file. The row value contains the Satoshi amounts and for each input address, the value is noted as 0, only output addresses in transactions contain values. Satoshi is the smallest currency unit in the bitcoin network and is converted with 100'000'000 Satoshis to 1 BTC (Lemieux (2013)). The rule column shows in which category the transaction belongs.

Transaction hash	Bitcoin address	WalletID	Time	Value	Rule
5dfaabc04b692a668393be60146a2ca748d21c76b57b0d284033b8d81bd47f75	15bDeG2PfXG1yAaYtQhEvjno7EN2JbFgNA	65737053	1435801555	4252221	2
7f66bcfa68e6121df428140dff3f3cd9036db8a797a7a2d9185946e70b121dae	1DiceoejxZdTrYwu3FMP2Ldew91jq9L2u	252743292	1505844750	10000	2
7f66bcfa68e6121df428140dff3f3cd9036db8a797a7a2d9185946e70b121dae	1BxBMRGtiwkGfWo9xk3RFG2yr5ohdETFLd	3777	1505844750	0	2
7f66bcfa68e6121df428140dff3f3cd9036db8a797a7a2d9185946e70b121dae	1MrZAgkGJ9MC4KVJMerLzwk3p7hk9f2L8q	3777	1505844750	3684	2
0d5e19f00b6d6d1a31ae74969c9e17a83b586e87e3f59e6d07a88bc18cf40a3f	3BMEXywsrKNhMguQSuxaztWKhXWh7MRV8b	255128346	1508482882	7078729	1
520484a42459b61fe65ac95c61f44f3d0bf7b552290ddf7d7accc450ab533cdc	33QoG5ioV4hseifKT9iaqrmD2eis7DicWA	289893470	1513786900	1E+12	1

Table 4: Main data frame with each address and each transaction

2. All the addresses from the off-chain table 5 have been searched in the main data frame and the corresponding transaction numbers are noted in the separate table 6.
   Table 6 contains all transactions involved in an activity like gambling or a Whale wallet transaction. This list has been filtered for duplicates to make sure a transaction is listed only once with its respective off-chain information.

Bitcoin address	Gambling	Whale
33QoG5ioV4hseifKT9iaqrmD2eis7DicWA		1
1DiceoejxZdTrYwu3FMP2Ldew91jq9L2u	1	

Table 5: Off-chain list of addresses and corresponding activity

Bitcoin address	Gambling	Whale
de6df5d2d92ecbfb4449b74f3d5d66279207598e493e2c0e45cb8410c2933cff	1	
7f66bcfa68e6121df428140dff3f3cd9036db8a797a7a2d9185946e70b121dae	1	
520484a42459b61fe65ac95c61f44f3d0bf7b552290ddf7d7accc450ab533cdc		1

Table 6: Off-chain transaction numbers and activity

3. All the transactions noted with the respective category, have been joined with the main data frame as showed in table 7.

Transaction hash	Bitcoin address	WalletID	Time	Value	Rule	Gambling	Whale
5dfaabc04b692a668393be60146a2ca748d21c76b57b0d284033b8d81bd47f75	15bDeG2PfXG1yAaYtQhEvjno7EN2JbFgNA	65737053	1435801555	4252221	2		
7f66bcfa68e6121df428140dff3f3cd9036db8a797a7a2d9185946e70b121dae	1DiceoejxZdTrYwu3FMP2Ldew91jq9L2u	252743292	1505844750	10000	2	1	
7f66bcfa68e6121df428140dff3f3cd9036db8a797a7a2d9185946e70b121dae	1BxBMRGtiwkGfWo9xk3RFG2yr5ohdETFLd	3777	1505844750	0	2	1	
7f66bcfa68e6121df428140dff3f3cd9036db8a797a7a2d9185946e70b121dae	1MrZAgkGJ9MC4KVJMerLzwk3p7hk9f2L8q	3777	1505844750	3684	2	1	
0d5e19f00b6d6d1a31ae74969c9e17a83b586e87e3f59e6d07a88bc18cf40a3f	3BMEXywsrKNhMguQSuxaztWKhXWh7MRV8b	255128346	1508482882	7078729	1		
520484a42459b61fe65ac95c61f44f3d0bf7b552290ddf7d7accc450ab533cdc	33QoG5ioV4hseifKT9iaqrmD2eis7DicWA	289893470	1513786900	1E+12	1		1

Table 7: Main data frame including off-chain activities on respective transactions

4. The "rule" column has been split into seven separate columns and marked with a 1 if it is in the corresponding column. This is visible in table 8 and had to be distributed in order for a later aggregation of the data frame.

Transaction hash	Bitcoin address	WalletID	Time	Value	Rule	Gambling	Whale	rule0	rule1	rule2	rule3	rule4	rule5	rule6
5dfaabc04b692a668393be6014...	15bDeG2PfXG1yAaYtQhEvjno7E...	65737053	1435801555	4252221	2					1				
7f66bcfa68e6121df428140dff3f...	1DiceoejxZdTrYwu3FMP2Ldew91...	252743292	1505844750	10000	2	1				1				
7f66bcfa68e6121df428140dff3f...	1BxBMRGtiwkGfWo9xk3RFG2yr...	3777	1505844750	0	2	1				1				
7f66bcfa68e6121df428140dff3f...	1MrZAgkGJ9MC4KVJMerLzwk3...	3777	1505844750	3684	2	1				1				
0d5e19f00b6d6d1a31ae74969c...	3BMEXywsrKNhMguQSuxaztWK...	255128346	1508482882	7078729	1				1					
520484a42459b61fe65ac95c61...	33QoG5ioV4hseifKT9iaqrmD2eis7...	289893470	1513786900	1E+12	1		1		1					

Table 8: Main data frame with distributed rule columns

5. The main data frame has been aggregated on walletID level, see table 9. For the column "time", always the first occurrence of this wallet on the bitcoin network has been saved in this aggregation.

WalletID	Value	Time	Gambling	Whale	rule0	rule1	rule2	rule 3	rule 4	rule 5	rule6
65737053	4'252'221	1435801555					2				
252743292	3'561'686	1505844750	31	1		3	22	2	2	2	
3777	195'225'048 billion	1505844750	37'122'931	27'861'827	17'992'372	18'823'432	238'267'144	138'843'519	166'874'727	305'662'573	108'814'426
255128346	259'991'050	1508482882				46	5		1	81	
289893470	1E+12	1513786900		13		2	4	7			

Table 9: Main data frame after aggregation on walletID

6. The total value per wallet has been classified into three groups:

   Value below 1 BTC (100'000'000 Satoshis)

   Value of 1 BTC until 10 BTC

   Value greater than 10 BTC

   The values which were below one Bitcoin, were classified with a 1 in the "lowvalueflag" category. Values between 1 and 10 Bitcoins where classified with a one in the "midvalueflag" category, and if the value is larger than 10 BTC, both categories are 0, see table 10.

WalletID	Value	Time	Gambling	Whale	rule0	rule1	rule2	...	lowvalueflag	midvalueflag
65737053	4'252'221	1435801555					2		1	0
252743292	3'561'686	1505844750	31	1		3	22		0	1
3777	195'225'048 billion	1505844750	37'122'931	27'861'827	17'992'372	18'823'432	238'267'144		0	0
255128346	259'991'050	1508482882				46	5		0	1
289893470	1E+12	1513786900		13		2	4		0	0

Table 10: Main data frame with value categorization

   14% of the wallets belong to the category "lowvalueflag" (44'007'954 wallets). In the category "midvalueflag", 79% of the wallets are located (245'613'757 wallets). The rest of the wallets (22'349'902 wallets) have values higher than 10 BTC, compared with the total amount of all wallets they are a minority with only 7%.

7. Each observation (row) sums up the values from rule 0 until rule 6. This amount is saved in the "total" column. All values from the rule categories, gambling and Whale are divided through this total. So the only ones untouched by this transformation are the two dummy variables for the values. This leads to a data frame, where the observation shows the percentage involved in a category. Such a transformation on observation level is necessary in order to not influence the result of behavior for users with high or low volumn of transactions. The data frame is visualized in table 11.

   The column "total rules" is just for supporting reasons and is not considered a new dimension.

WalletID	Time	Gambling	Whale	rule0	rule1	rule2	rule 3	rule 4	rule 5	rule 6	lowvalueflag	midvalueflag	total rules
65737053	1435801555					100%					1	0	2
252743292	1505844750	100%	3%		10%	71%	6%	6%	7%		0	1	31
3777	1505844750	4%	3%	2%	2%	24%	14%	17%	30%	11%	0	0	995'278'193
255128346	1508482882				33%	3%		1%	59%	4%	0	1	259'991'050
289893470	1513786900		100%		15%	31%	54%				0	0	13

Table 11: Main data frame with percentages based on sum of all rules

The final aggregated data frame is the size of 13 GB. The largest wallet contained 1'952'250'477 BTC with the walletID 3'777, this cluster is considered as supercluster. On this data set the data sampling from chapter 4.4 has been performed.

4.4 Data sampling

This chapter will explain the data sampling process which is referenced in the architecture figure 1, point 7. The first pre-processing part contained 2.7 billion rows in the CSV file, which summed up to a total of 320 GB. After including off-chain information and aggregating the walletID the final pre-processing file contained 311'971'613 rows, with a total of 13 GB. Those 312 million rows represent all wallets which were identified in the bitcoin network and already mapped in the pre-processing part. One wallet symbolizes one user/entity. This leads to the hypothesis, that approximately 312 million users/entities have been involved in bitcoin transactions. To handle this volume of data, without getting memory errors in the machine learning part, a representative sample had to be selected. In this context a memory error means, that the given RAM space is not sufficient while performing the computation. With a server space of 128 GB RAM and a file size of 13 GB, problems will likely occur, especially for the machine learning algorithms with the SciPy library. Usually the SciPy algorithms contain the package pandas library, where it is common that the whole data set is loaded in the data frame and taking up RAM space (pandas development team (2020)). During computing with this large volume the RAM space quickly reaches its limit and returns a memory error in the best case, or it just calculates for a very long time until the loading process breaks down. To find a feasible solution instead, a representative sample has been selected from the

312 million wallets. A sample means, a subset from the whole quantity, reflecting the entire data set accurately. The first step was to choose a statistically approved method with the z-score, in order to choose the correct sample size. To determine such a sample size four key considerations need to be made.

N	The population size is the number of wallets in the data set. As each wallet represents a user or entity, the population size is 312 million (Maple Tech. International LLC. (2020)).
ε	To calculate a sample, a margin of error needs to be chosen, indicating how much higher or lower the sample mean could deviate from the actual mean of the population. A small margin of error leads to a better precision. Therefore, an error margin of 1% has been defined (Maple Tech. International LLC. (2020)).
Z	Confidence level, representing the percentage of which the actual mean of the sample group falls within the defined margin of error. The most common confidence level in research and statistics lies at either 90%, 95% or 99%. To make this sample most representative, a level of 99% has been chosen (Maple Tech. International LLC. (2020)).

p The standard deviation shows variance that is expected in the data sample. As this is not clear, given the data set a value of 0.5 is preferred, as this value will lead to a large enough sample size. Although, there is no knowledge of the correct distribution of the data set (uniform, binominal, random or normal distribution), there is an important mathematical theorem called central limit theorem, which proves, that in a large enough sample, the distribution tends towards normal distribution. The mathematical formulation of the central limit theorem is as follows: $\frac{\bar{x}-\mu}{\sigma/\sqrt{n}}$. Here the mean is 0 and the variance 1 as n approaches infinity. In order to get a statistically significant result, that the central limit theorem holds, a large sample has to be selected. The proportion p is 0.5, as the area on either side of the mean is 0.5 (50%).

With this information the z-score can be calculated. A z-score, also called standard score, represents how many standard deviations a value falls from the mean. To calculate a z-score the population mean as well as the standard deviation needs to be known. With a confidence interval of 99% for both sides, the calculation is $\frac{1+0.99}{2}$ which returns 0.995. This value can be checked in a standard normal distribution table for z-scores and returns the rounded value of 2.58. With this z-score the following formula 1 will be used (Daniel and Cross (2018) Maple Tech. International LLC. (2020)).

$$\text{Sample size} \quad = \frac{\frac{z^2 * p(1-p)}{e^2}}{1 + (\frac{z^2 * p(1-p)}{e^2 N})} \tag{1}$$

With the mentioned values inserted, a minimum sample of 16’641 returns. Going forward, this value has been rounded to the next 10k, for which a sample of 20’000 has been chosen. As the bitcoin blockchain is a time series, where a simple cut in-between would not represent the whole data properly. Therefore, the data for wallet creation per year has been filtered, and the 20’000 samples distributed accordingly. The table

13 represents the amount of wallets created in each year and the sample size for later analysis.

year	wallets	sample
2010	33’525	2
2011	195’777	13
2012	540’478	35
2013	2’455’384	157
2014	12’131’927	778
2015	23’036’055	1’477
2016	33’994’234	2’179
2017	48’204’669	3’090
2018	58’435’689	3’746
2019	58’711’393	3’764
2020	74’232’482	4’759
Total	311’971’613	20’000

Table 13: Created wallets per year

The sample size does not only represent the created wallets per year, but also correlates with the number of transactions per day.
On `https://www.blockchain.com/charts/n-transactions` there is a graphical overview on how the bitcoin networks transaction per day are distributed. Graph 6 shows this correlation together with the chosen samples.

Each year is calculated from June 24, 20xx until June 23, 20xx+t_1. The first year starting on January 3, 2009 is calculated until June 23, 2010. The last year is therefore June 24, 2019 till June 23, 2020 in this data set. With PostgreSQL all the wallet data has practically been split into 11 tables, for each year one table. The tables have been sorted randomly and the number of samples per year according to table 13 have been exported into CSV for the machine learning part. These 20’000 samples weighted 866 KB in the CSV file.

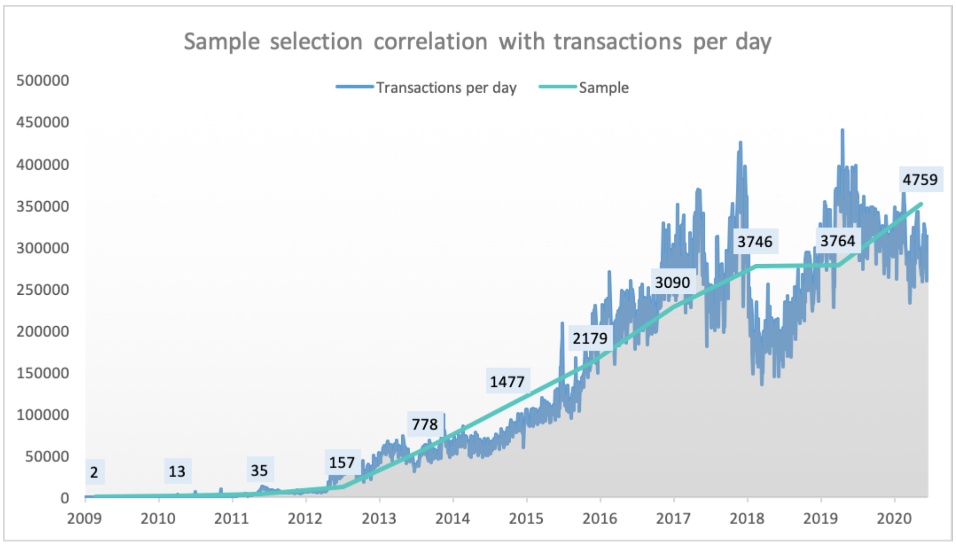


Figure 6: Sample selection line plot correlating with transactions per day line plot

5 Machine Learning

"An intelligent being cannot treat every object it sees as a unique entity unlike anything else in the universe. It has to put objects in categories so that it may apply its hard-won knowledge about similar objects, encountered in the past, to the object at hand." Pinker (1997)

Machine learning is a hype word in nowadays business environment and means basically learning from data. Machine learning is designing algorithms that enable computers to learn. The field of machine learning can be divided into multiple areas like supervised or unsupervised learning, as well as semi-supervised, reinforcement learning, etc. Supervised learning algorithm maps inputs to the desired outputs and is therefore devoted to the classification problem. It divides the data in a test and training set, and as there is a target variable, it is possible to calculate how precise the algorithm is. Due to this learning approach, the computer gets to learn how to classify the respective problems reasonably accurately. Unsupervised machine learning, on the other hand, has only a set of input data but no labels are available. There is no fixed target variable. This makes the analysis more difficult, as the computer has to learn how to cluster something, but it is not defined in which way (Zhang (2010)).

Since the bitcoin data has no labeled categories to whom the address belongs, unsupervised machine learning needs to be applied. That is why this thesis is focusing on unsupervised machine learning. This reflects step number 8 on the architecture landscape from figure 1. There are different kinds of unsupervised machine learning algorithm being explained below:

- Clustering: This category divides the entire data into groups when no pre-defined categories/classes are available. Clustering methods can be further classified into roughly four major categories: partitioning-based methods, density-based methods, hierarchical methods and grid-based methods (Huang et al. (2017)).
- Dimensionality reduction: The name already states that these algorithms reduce the number of dimensions. The most popular ones are principal component analysis (PCA), independent component analysis or non-negative matrix factorization (Huang et al. (2017)).
- Outlier detection: This unsupervised method finds unusual events (e.g. malfunctions). The most well-known local anomaly detection algorithm is called the local outlier factor (Goldstein and Uchida (2016)).
- Novelty detection: This category is specialized on finding changes in the data (Huang et al. (2017)).

There is no algorithm that works for every data set. The first step is to analyze the data set and then choose the algorithm which works best on it. Depending on factors like size, structure or noise, different algorithms perform better. Having a look at the bitcoin data set, there are 20'000 observations, which are the aggregated wallets, and 11 features, which represent the dimensions. This number of observations is very high, and the number of dimensions is also high. However, it is not in the category of high dimensional data, as high dimensional data is classified as the number of features that can exceed the number of observations. The next impression is to find out how much noise there is in the data as well as the shape of the data. This can be outlined by algorithms like DBSCAN for noise and PCA for the shape. Due to the large number of observations, a computationally effortless algorithm is preferred. Therefore, machine learning algorithms which can deal with a large number of observations work best (Huang et al. (2017)).

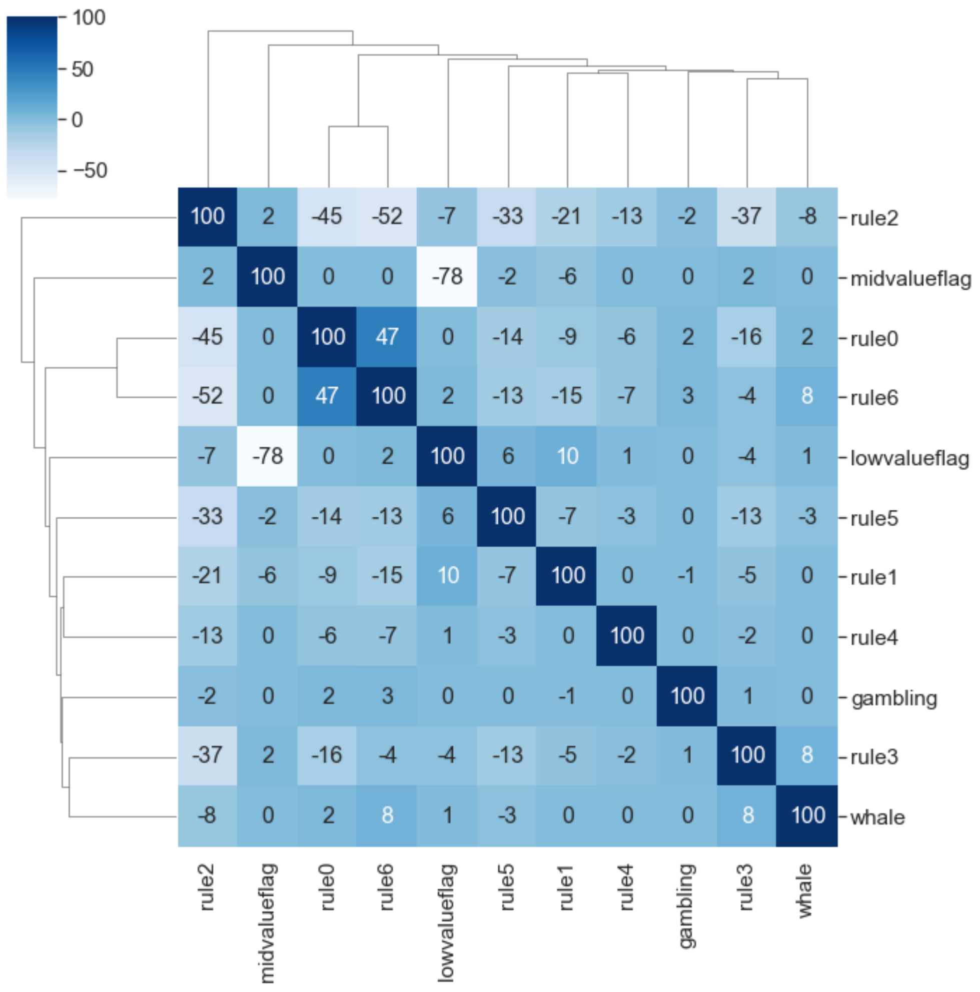
Before having a deep dive into the different machine learning algorithms applied in this thesis, the different features (dimensions) are summarized in table 14. Important to notice is that, the features selection had to be defined in an early stage of the whole research, as the pre-processing part took up several weeks. All the values within a dimension rule 0 till rule 6 are standardized between 0 and 1. A value of 0 means no address of the specific walletID from owner X was involved in a transaction with this rule. A value of 1 means, all the addresses from the walletID from owner X where involved in this rule. A value of 0.5 in one rule means, 50% of all addresses from this walletID where involved in that rule. Together all the rule dimensions sum up to 1 (100%).

Dimensions	Description
Rule 0	This dimension represents the transaction structure in the bitcoin network where the miner receives newly generated bitcoins from the bitcoin network, when verifying the transactions within one block. This transaction is called coinbase transaction and this Bitcoin money can be considered as clean, as this is newly generated and has not been involved in any kind of illegal activity before. Overall 3.4% of all transactions are involved in such coinbase transactions.
Rule 1	This is a 1 to 1 transaction, where one address sends money to another address. This kind of transaction occurs in 8% of all transactions.

Rule 2	With 58.4%, this transaction is the most prevalent type in the bitcoin network. It is a 1 to 2 transaction, meaning one address sends Bitcoins to two addresses. Usually one of the two output addresses belong to the input address owner, as this signals the change of the Bitcoin.
Rule 3	This is a 1 to N transaction. One input address sends Bitcoins to multiple addresses. This type of transaction is occurring in 6% of all transactions.
Rule 4	Rule 4 stands for N input addresses and 1 output address. This type of transaction occurs 4% of all Bitcoin transactions.
Rule 5	This transaction includes N input addresses and 2 output addresses. The frequency of occurrence in the Bitcoin network is 19%.
Rule 6	This is a N to N transaction and occurs in 1.2% of all transactions in the Bitcoin network.
Gambling	This dimension shows which addresses have been involved in gambling activities. 1.9% of all transactions have been labeled as gambling related.
Whale	All wallets which contain over 1'000 BTC are considered Whale wallets. These rich wallets have been identified and 0.5% of all transactions are involved in transactions with Whales.
Lowvalueflag	This is a dummy variable, where the value 1 means that a wallet contains less than 1 Bitcoin. From all the wallets involved in the bitcoin network, 14% had a total value of less than 1 BTC.
Midvalueflag	This is a dummy variable, where the value 1 means that the wallet contains between 1 and 10 Bitcoins. If both value flags are indicating 0, then the total value of the wallet is over 10 Bitcoins. In 79% of the wallets, they were marked as midvalueflag.

Table 14: Dimensions explained

As an overview, figure 7 shows the correlation between every dimension in a correlation matrix.



	rule2	midvalueflag	rule0	rule6	lowvalueflag	rule5	rule1	rule4	gambling	rule3	whale
rule2	100	2	-45	-52	-7	-33	-21	-13	-2	-37	-8
midvalueflag	2	100	0	0	-78	-2	-6	0	0	2	0
rule0	-45	0	100	47	0	-14	-9	-6	2	-16	2
rule6	-52	0	47	100	2	-13	-15	-7	3	-4	8
lowvalueflag	-7	-78	0	2	100	6	10	1	0	-4	1
rule5	-33	-2	-14	-13	6	100	-7	-3	0	-13	-3
rule1	-21	-6	-9	-15	10	-7	100	0	-1	-5	0
rule4	-13	0	-6	-7	1	-3	0	100	0	-2	0
gambling	-2	0	2	3	0	0	-1	0	100	1	0
rule3	-37	2	-16	-4	-4	-13	-5	-2	1	100	8
whale	-8	0	2	8	1	-3	0	0	0	8	100

Figure 7: Correlation of dimensions

- Interestingly, the dimensions of rule 0 and rule 6 correlate strongly. Rule 0 represents the coinbase transactions (generated from the system) and rule 6 the N to N transactions.
- Lowvalueflag and midvalueflag correlate negatively, as each wallet is either categorized in a lowvalue or a midvalue. If none of them are applicable, these dimensions have a value of 0 and will be interpreted as a high value. The negative correlation also impli-

cates that most of the transactions have either a value below one or between one and ten Bitcoins.

- Category gambling does not correlate with the dimensions rule 4, rule 5, lowvalueflag, midvalueflag and Whale.
- In general, the correlation matrix displays low correlation values between the features. This heatmap, together with a dendrogram out of the hierarchical clustering side, has been combined with the Python seaborn library function clustermap. For the clustermap no NA values (missing values) are allowed. Hence all the values of the data set which have been left blank are filled with 0's using the function df.fillna(0) from pandas library (pandas development team (2020)).

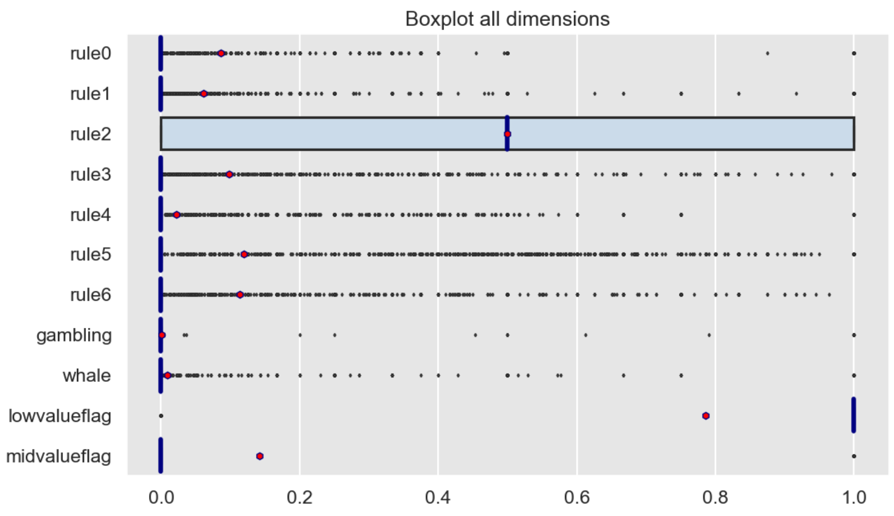


Figure 8: Boxplot over all dimensions

Figure 8 presents the distribution of each dimension as a boxplot. The median values are marked with a blue stroke and the mean values represented by the red dots for each dimension. It is visible that rule 2 has quite a considerable impact and shows a kind of a uniform distribution. Rule 2 is a 1 to 2 transaction and is the most used transaction type in the bitcoin data. As shown in table 2 the total amount of rule 2 over all transactions was approximately 60%.

5.1 K-Means

The most common benchmark algorithm in clustering is k-means. It makes clusters based on geometric distances. On large data sets, the algorithm performs well when the structure is “round” or spherical, equally sized clusters, equally dense, most dense in the center of the sphere and not contaminated by noise/outliers. K-means count as a fast, flexible and straightforward algorithm and clusters with centroids. The algorithm starts with randomly initiating k points in the data and then assigns the rest of the points to the nearest of the k centroid. When all points are assigned to one cluster, the centroids move to the average of all the assigned points in a cluster and then the process repeats. All points are again assigned to its nearest neighbor centroid. This will be repeated, until no point changes position anymore, meaning the algorithm converges (Dabbura (2018)). The distance-based measurement calculation is to determine the similarity between data points. To calculate the distance measures properly, the data needs to be standardized. This is the case, because the standardizing step has been performed in the aggregation chapter 4.3.

K-means has some problems: The main disadvantage is the difficulty to define an optimal k (number of clusters). To handle this problem, there are internal validation methods like the Calinski-Harabasz method or the Silhouhette method, which can support in choosing the right k. Both these methods and some others will be explained later in this chapter. A second problem with k-means is that in very high dimensional space, the Euclidean distance becomes inflated. This phenomenon is called "curse of dimensionality". For the curse of dimensionality, a method called principal component analysis (PCA) exists, which reduces the dimensions. Therefore, PCA will be used in this thesis as well. Additional problems with k-means occur when the data clusters are not globular. K-means will have a hard time identifying those. Overlapping data will result in incorrect clustering, as k-means has to distribute all the data points to one cluster.

5.1.1 Internal validation measures

In order to define the k-clusters which suits the data set best, internal cluster validation techniques can be applied. There exist dozens of validation measures, so only a few were selected for this data set. Choices were based on stated success in the literature and efficiency of the technique. Two universal criteria for validity measures in general are homogeneity and separation. This means, that good clustering brings together similar points and separates other data points, which do not belong to that cluster. Randomness is one cause, why different outcomes are generated.

Elbow method

To visually show the validity of the number of clusters, an elbow method was used. For each k, the sum of squared errors (SSE) are calculated and the plot shows a chart, which demonstrates an arm, where the "elbow" should indicate the best k. The SSE in general tends towards 0 with increasing k number of clusters. And the elbow displays a compromise between finding a small number of k with a low SSE score.

$$\mathrm{SSE} = \sum_{i=1}^{n} (x_i - \bar{x}) \tag{2}$$

n is the number of observations, x_i represents the value of the i observation and $\bar{x}$ is the mean of all the observations (Baarsch and Celebi (2012)).

It is not always easy to identify the elbow with the best amount of k clusters, which can be seen in figure 9. As a trade-off between the value of inertia and the rising number of k, the optimal point is where the elbow lies. Inertia is a multivariate measure for the data set where the variance is explained (Johnson et al. (2002)). It is not always clear where

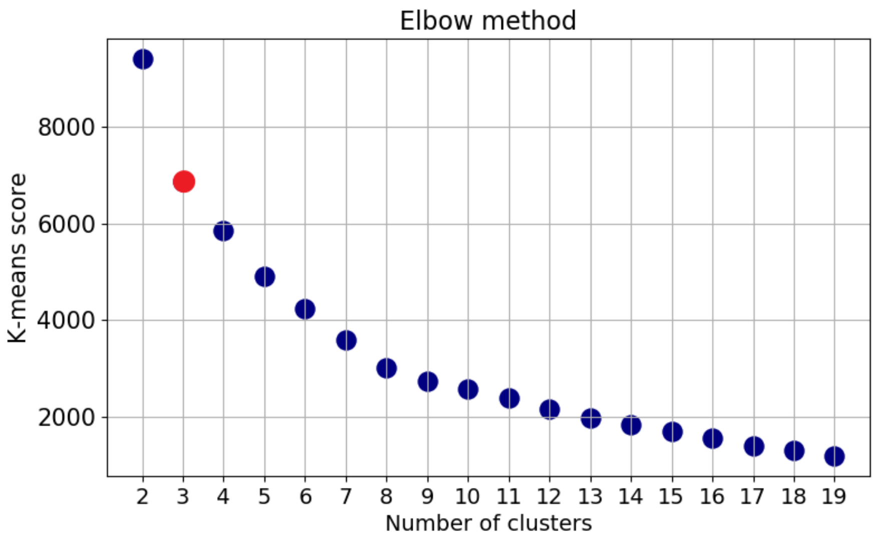


Figure 9: K-means elbow plot

the elbow lies, so other validation methods are recommended instead or additionally. In this thesis k = 3 was selected. Overall, with the other inter cluster quality measures, total of three clusters are preferred, see methods below.

Davies-Bouldin score

The Davies-Bouldin (DB) method links compactness to separation. This is done by calculating the sum of the average distances from each point to its respective center (compactness) and dividing by the distance between the cluster centers (separation). The formula for the Davies-Bouldin score is:

$$\mathrm{DB} = \frac{1}{k}\sum_{i=1}^{k} R_i \tag{3}$$

Where R_i is maximized $max(R_{ij})$, $i \neq j$ and R_{ij} is calculated $(S_i{+}S_j)/M_{ij}$. S_i is the sum of the average distances from each point in the cluster i to its centroid of its cluster. M_{ij} stands for the distance between the two clusters in this case. According to formula 3, a lower score in DB is preferred. This means the cluster will be more compact itself and more separated from others (Baarsch and Celebi (2012)).

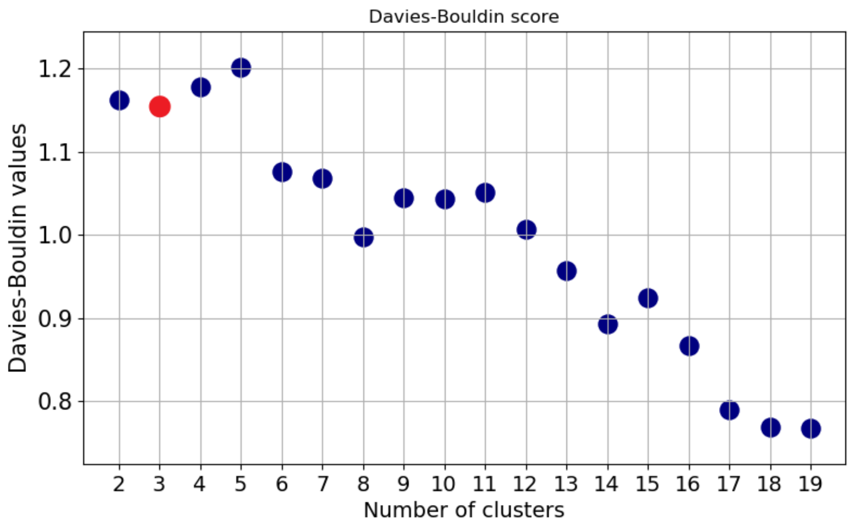


Figure 10: Davies-Bouldin plot

In figure 10 the Davies-Bouldin score is bumpy. As mentioned, a low amount of the Davies-Bouldin score is preferred. The first lower amount is at cluster k = 3, the lower value is around k = 6 (Baarsch and Celebi (2012)).

Silhouette coefficient

The Silhouette method is based on the mean score for each data point. The point's individual score is based on the difference between the average distance from that point and all the other points in one cluster. Then the minimum average distance is calculated between that point and the other points of each other cluster. Afterwards, the difference is divided by normalizing the term, which results in the greater value of the two averages.

$$\text{Silhouette coefficient} = \frac{1}{N}\sum_{i=0}^{N} s_{x_i} \tag{4}$$

N are the data points. $s_{x_i} = (b_{q,i} - a_{p,i})/max(a_{p,i}, b_{p,i})$. If x_i is a point in the cluster q, then $b_{q,i} = min\ d_{q,i}$ where $d_{q,i}$ is the average distance between point x_i and every point of cluster p. To know which cluster size is optimal, the Silhouette score should approach 1 (Baarsch and Celebi

(2012)).

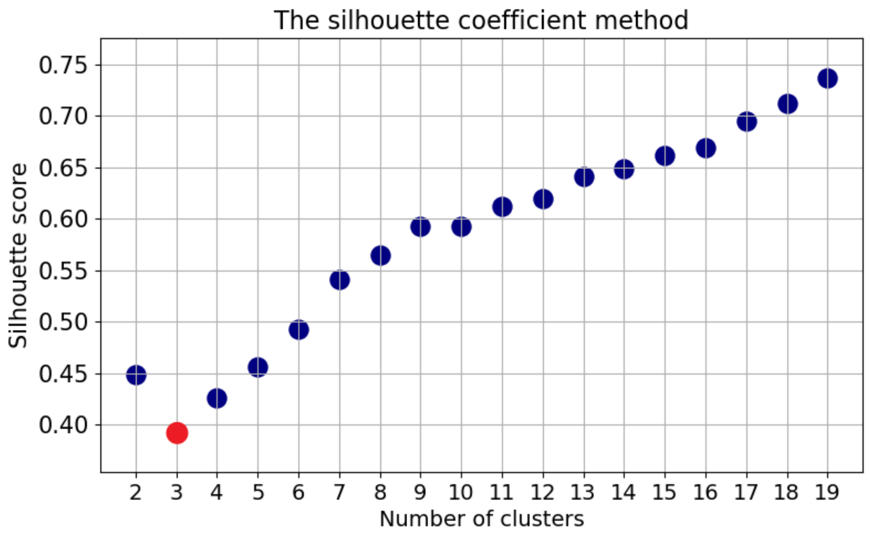


Figure 11: Silhouette plot

Here the Silhouette score in figure 11 is increasing, which should be the case, but at k = 3 the score drops before it starts to rise again. K = 3 is not a suitable choice according to this internal validation method. As all the validation methods are calculated differently, a perfect match for all of them on the same number of k is unlikely to be the case. It is essential to choose the best k according to the data set and the internal validation method which displays the situation of the data point most properly. A reliable validation method is the Calinski-Harabasz method described below (Baarsch and Celebi (2012)).

Calinski-Harabasz

The Calinski-Harabasz (CH) method compares the between cluster scatter matrix (BCSM) and the within cluster scatter matrix (WCSM). The second fraction of the formula is a normalization factor which diminishes the score as the number of clusters (k) increases.

$$\text{CH} = \frac{trace(BCSM)}{trace(WCSM)} * \frac{N-k}{k-1} \tag{5}$$

This method uses cluster centers of the data to calculate separation, where the centers of the data set is used instead of the cluster centers. A good cluster size is when the CH score is maximized (Baarsch and Celebi (2012)).

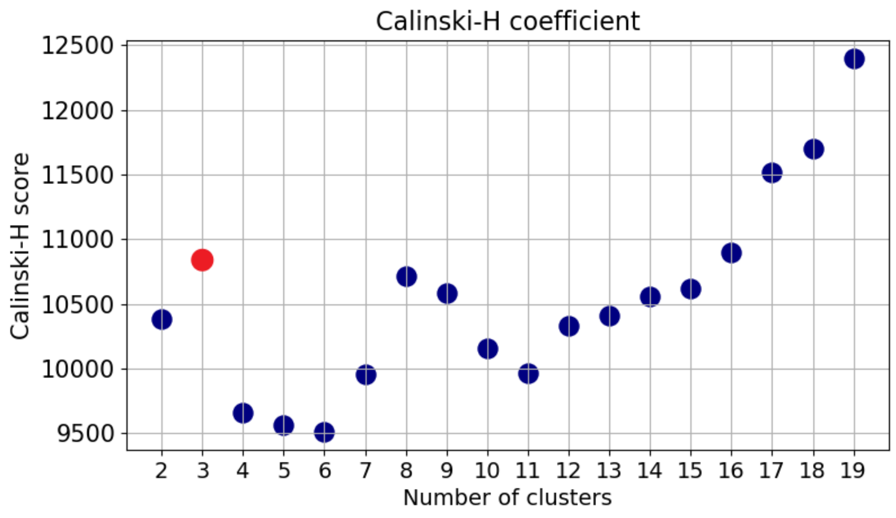


Figure 12: Calinski-Harabasz plot

A good cluster size in figure 12 is k = 3, as it produces the highest amount of the first fifteen k's. As the formula 5 points out, the division of the "between cluster scatter matrix" and the "within cluster scatter matrix" is a decent mathematical standpoint to choose the number of clusters, in this thesis k = 3 has been chosen (Baarsch and Celebi (2012)).

5.1.2 Excecution

With the scikit-learn library from Python, the k-means algorithm with three clusters has been performed. To get the same clusters in a repeated run, the following parameters need to be used:

- As k-means always starts randomly at a point from where it calculates the centroids, a seed generator needs to be set, in order to get the same result when performing the experiment again. From the numpy library the *numpy.random.seed(0)* was set at the beginning.

- In the command sklearn.cluster.KMeans() the number of clusters needs to be set n_clusters = 3. The method for initialisation (init) is by default the "k-means++". This one selects the initial cluster centers for the k-means in a smart way to speed up convergence by careful seeding (Arthur and Vassilvitskii (2006)). In the algorithm itself the random_state which is also a seed indication is set to 0 again. This is important for replication of the algorithm. The whole command is below (Pedregosa et al. (2011)).

  ```
  sklearn.cluster.KMeans(n_clusters=3, init='k-means++',
  randome_state=0)}
  ```

When performing the k-means algorithm, the output of the sample data gives three clusters according to table 15.

Cluster	Number of datapoints	Percentage of total
Cluster 1	9’281	47 %
Cluster 2	4’274	21 %
Cluster 3	6’445	32 %
Total	20’000	100 %

Table 15: K-means clustersize

Table 15 shows a typical attribute of k-means. All points belong to one of the clusters. There is no noise detection and no fuzzy clustering by overlapping clusters. So all the data points are mapped to only one cluster.

To get a deeper understanding about the clusters, each dimension has been listed in relation to the clusters in boxplot figure 13.

- Figure 13 shows that cluster one has a significant higher amount in dimension rule 2. Cluster one only includes wallets that have low-value transactions (below 1 BTC), as the dimension lowvalueflag has a median and mean of 1.

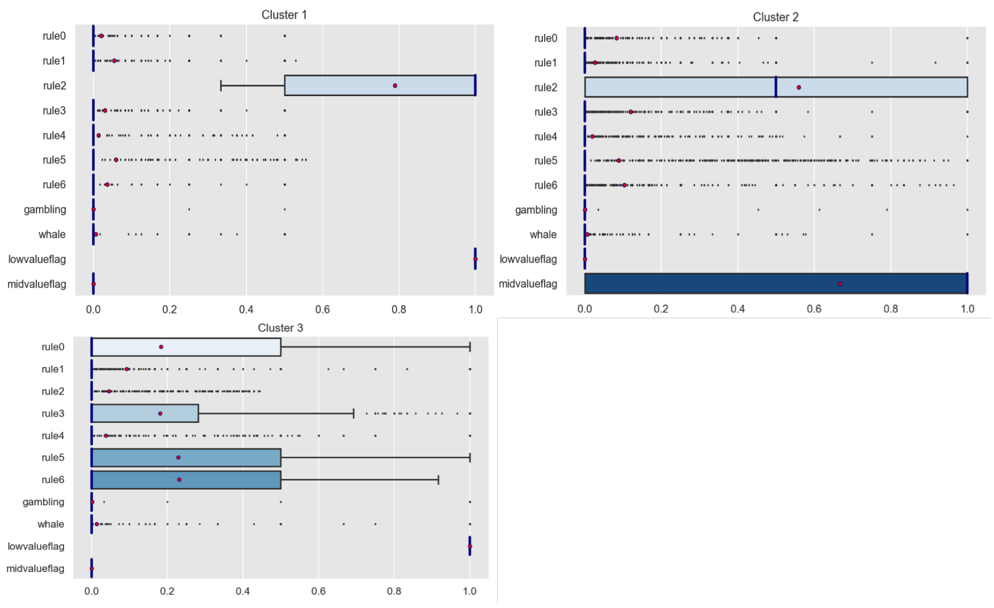


Figure 13: K-means with k = 3, boxplot over 3 clusters

- Cluster two has low and high values in dimension rule 2. Next to that, the dimension midvalueflag shows a median of 1 in this dimension and the mean is above 60%. Also the high values are included in cluster two. So cluster two contains almost all values over 1 BTC.
- The third cluster shows higher means in rule 0 (coinbase transactions), rule 3 (1:N transactions), rule 5 (N:2 transactions) and dimension rule 6 (N:N transactions). Additionally, cluster three gathers mostly lowvalue wallets as the mean and median is at 1.

This cluster structure is important to know in order to find out more about the nature of the wallet.

For further analysis, the plotting of the data points supports the interpretation. However, plotting for a multidimensional data set as in this thesis, the functions are limited. There is one plot functionality from scikit-learn called parallel_coordinates which shows how the clusters are formatted over several dimensions.

Figure 14 shows the first cluster of k-means over the different dimensions.

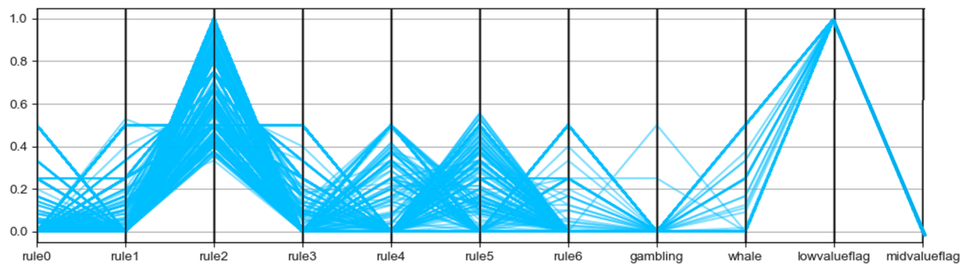


Figure 14: Cluster 1 k-means parallel plot based on Llewelyn (2019)

Visible are triangle shapes which show that some values rise until the maximal value of one and some stay at 0. This plot should give an understanding about how the different features are connected in each cluster. Similar to the boxplot from before (figure 13), in cluster one (blue lines in figure 14) the dimension rule 2 is clearly higher for all data points and dimension lowvalueflag confirms that this cluster contains small values.

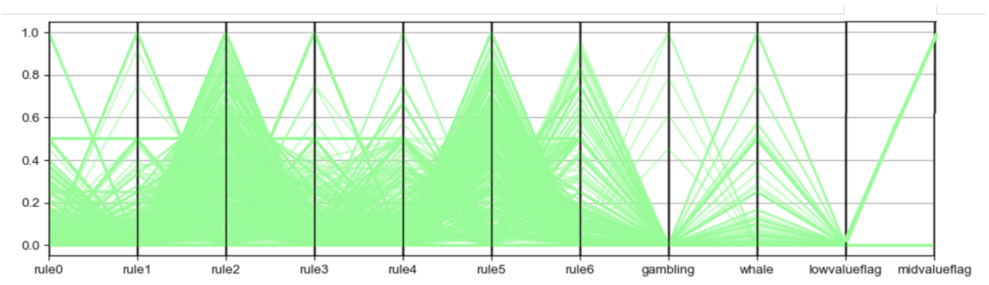


Figure 15: Cluster 2 k-means parallel plot based on Llewelyn (2019)

The figure 15 displays cluster two in light green over all the dimensions. Rule 2 and rule 5 have stronger lines and the values for midvalueflag dimension are higher.

In figure 16 cluster three with dark green lines shows very strong attributes in rule 0, rule 3, rule 5 and rule 6 as well as lowvalueflags. In all three parallel_plots the gambling dimension is negligible.

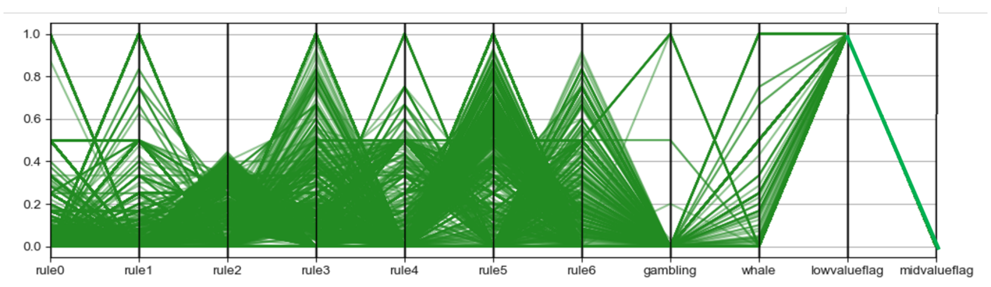


Figure 16: Cluster 3 k-means parallel plot based on Llewelyn (2019)

5.1.3 Validation

In unsupervised learning, it can be difficult to understand the meaning of each cluster. To avoid this problem, US blacklisted addresses have been grouped into the corresponding wallets as well. This should give a better comparison of the received k-means clusters.

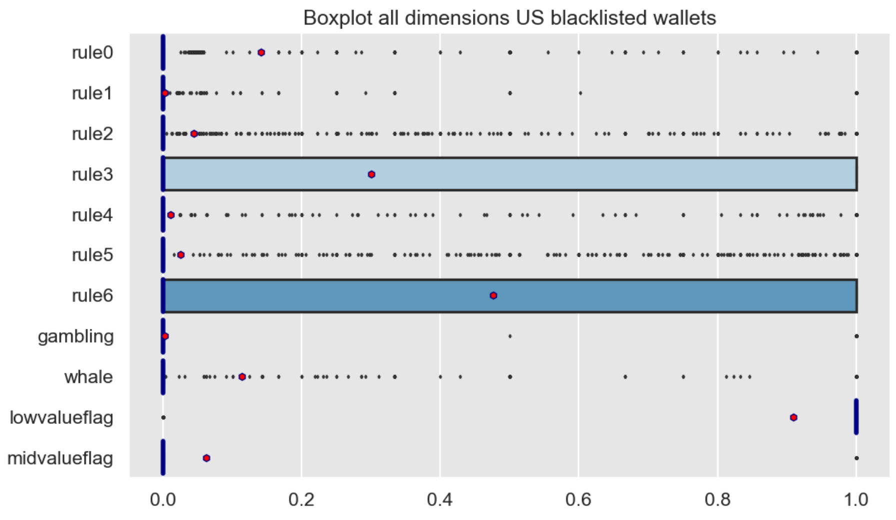


Figure 17: Boxplot US blacklisted values

In total 28’581 wallets were included in transactions of such blacklisted wallets. This data has been plotted in figure 17 to get an overview of the distribution. It is interesting to note that rule 3 and rule 6 are occurring very often, as well as the lowvalueflag. In the boxplot figure 17 the median values are marked with a blue line and the means are represented by the red dots for each dimension. Observing this boxplot with the higher mean from dimension rule 3 and rule 6 can lead to the assumption that such blacklisted wallets behave in a way, that rule 3 (1 to N transactions) and rule 6 (N to N transactions) are often involved together with a low value below 1 BTC. This result gives a better insight in the k-means clustering from before figure 13, where cluster three was highlighted specially to have a larger proportion of the three dimensions rule 3, rule 6 and lowvalueflag. The k-means algorithm can give a pre-selection on interesting wallets, which are likely to contain blacklisted wallets.

Connection with mining pools

Cluster three from boxplot figure 13 has not only higher values in rule 3 and rule 6, but also in the dimension rule 0 and rule 5. These two dimensions were not significantly present in the distribution from the US government black listed wallets in figure 17. This means that the k-means algorithm most likely detected similar patterns between miners who are involved in rule 0 and the black wallets. The researchers Wang and Liu (2015) explained how the miners transformed their behavior over time. In the beginning all the miners where solo-miners, meaning they mined the bitcoin blocks by themselves. However, over time due to efficiency reasons and bundling of computation power, miners startexrm pools, where the earned bitcoins where splitted in shares. This reward model is the most occurring one and is called pay-per-share. This pool mining constellation, brings the same expected payout for a miner doing solo mining, but the variance of the payout is largely reduced. This behavior of pool mining brings a connection of rule 0 (coinbase transactions), rule 3, rule 5 and rule 6 where multiple inputs and multiple output addresses are involved. The mining pool needs to distribute shares to a lot of addresses involving multi input, multi output transactions. A more recent paper focused on the distribution system from mining pools as well (Romiti et al. (2019)). They analyzed three of the four largest mining pools and documented that the pay-out system. The earnings distribution can either be a tree-like structure, or by randomly choose a number of miners to pay in one transaction then shift to a new ID and distribute payouts to the next group of miners. Other variations are possible as well, but for this thesis it is beneficial to see the connections between rule 0 and rule 3, rule 5 and rule 6. An extension of the payout schema structure of the paper Romiti et al. (2019) is provided in appendix A.

Even though cluster three contains miner wallets as well, with 32% of all the sample data (table 15) this cluster is selected for follow up research outside of this thesis. Authorities with interest in knowing which wallets are potentially black wallet owners, should analyse this cluster three especially well.

Recall calculation
In Python scikit-learn library (Scikit-learn (2020)) there is a function from k-means to predict new data on the clustering made from previous data. During this thesis, the black wallets have been analyzed with the k-means predict function to control how many data points flow into each of the three clusters. The expected result would be, that most of the data points from the black wallets should be categorized in k-means cluster three. Since cluster three had much higher values in rule 3, rule 6 and the lowvalueflag dimension. The appropriate code line for this is:

```
model      = sklearn.cluster.KMeans(n_clusters = 3,
init='k-means++', randome_state=0).fit(sample_data)

blacklabel = model.predict(blackwallets)
```

Table 16 shows the result after predicting the black wallet labels from the k-means algorithm initially trained on the sample data. With 88% of all data points from the black wallets belonging to cluster three, this is evidence, that the initial k-means algorithm was successful in clustering the black wallets into a cluster.

Summarizing on this k-means chapter 5.1, it can be said, that k-means detected a cluster which had similar attributes to the US government blacklisted wallets. These attributes contain a significant number of data points in the three dimensions rule 3, rule 6 together with lowvalueflag.

Cluster	Number of datapoints	Percentage of total
Cluster 1	856	3 %
Cluster 2	2'570	9 %
Cluster 3	25'155	88 %
Total	28'581	100 %

Table 16: K-means predict on black wallets

A pattern emerges from black wallet owners, who like to transact in multi input / multi output transactions, in order to blur tracks and distribute small amounts to multiple addresses. With the verification technique of predicting the black wallets on the trained k-means algorithm, 88% of the black wallets landed in the same cluster three. Since the k-means cluster of the potentially black wallets (cluster three in figure 13) also have higher means in dimension rule 0 and rule 5, there are probably other wallet types as well included like the mining pools. They do not necessarily need to be black wallets. One group of owners are the miners, which increasingly collaborate in pools in order to have steady earnings over time. This mining pools distribute the earned new Bitcoins to its members and therefore transact in multi input and or multi output transactions as well.

5.2 PCA

In the previous subchapter, it was shown that k-means could have a problem with the curse of dimensionality involving too many dimensions. The principal component analysis (PCA) can help with this matter.

PCA is a technique for reducing the number of dimensions in a data set while retaining as much information as possible. It is using the correlation between some dimensions and tries to provide a minimum number of variables that keep the maximum amount of variation on the original distributed data set. While PCA calculates which dimensions contribute the most to the clusters, it is also a technique to separate the data from the noise. For PCA it is crucial, to pre-process the data first which has been done in chapter 4.3.

As PCA can be applied to any data matrix, if properly transformed and scaled, this is a very good first step for a multivariate analysis as for example with k-means. PCA helps to see the structure of the data, with outliers or delineate classes. Mathematically PCA is calculating with eigenvalues and eigenvectors of the data-matrix. Eigenvector stands for a vector whose direction stays unchanged when applying a linear transformation. Eigenvalues are scalars if they are high. It is considered to contain more information on our data distribution than vice versa. There are some basic steps to follow (Derksen (2016)):

1. Standardizing the data.
2. Calculating the covariance matrix of the whole data set.
3. Computing eigenvectors $(e_1, e_2, ..., e_d)$ and corresponding eigenvalues $(\lambda_1, \lambda_2, ..., \lambda_d)$,
4. Sorting the eigenvalues in descending order.
5. Forming principal components (PC) by selecting the k eigenvectors with the largest eigenvalues, where k is the number of dimensions used in the new feature space (k$\leq$d).
6. Projection into the new feature space.

Always when performing a principal component analysis, it is important to plot a cumulative explained variance plot in order to see how much variance is explained by the principal components. In figure 18, the first principal component explains almost 40% of the data sets variance, the second principal component explains around 30% and the third principal component explains 10% which leads together to almost 80% explained variance within the first three principal components. It is desirable to reach a high percentage. Otherwise it can be assumed that there are too many features which do not contribute to explaining the model in addition to existing features.

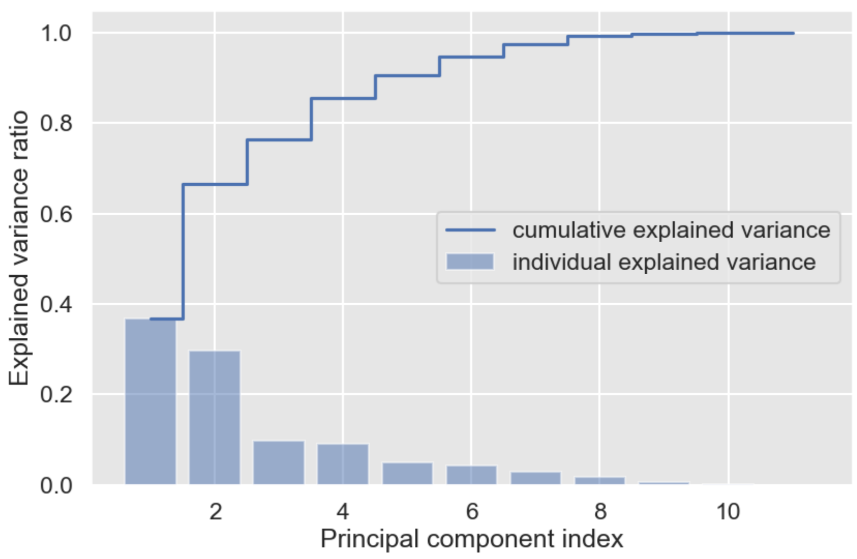


Figure 18: PCA explained variance

Components after PC 3 should be ignored, as they only contribute a little increase in the total explained variation. Another term for this is called the law of diminishing marginal returns. Including PC dimensions after the third PC does not give much additional value to the model. It makes sense to look at PCA's which explain together a significant percentage of 90% or more from the variances (Holland (2008)). To interpret the principal component result, it is useful to analyze the correlation between the original data variables and the principal components. The correlation of variable X_i and principal component Y_j is

$$r_{ij} = \sqrt{a_{ij}^2 * var(Y_j)/s_{ii}} \tag{6}$$

a_{ij} is the i-th variable principal component weight on a principal component j and $var(Y_j)$ is the variance of the jth principal component score divided through s which represents the diagonal matrix of eigenvalues.

Table 17 shows the dimension composition of PCA one to three. They are called loading vectors (James et al. (2013)). The most influential values have been marked bold. As the values have been standardized, all the values lie between -1 and 1. A value of 1 is perfectly correlated

Dimensions	PC 1	PC 2	PC 3
rule0	-0.0424	0.1955	**-0.3041**
rule1	-0.0466	0.0476	0.0320
rule2	**0.2377**	**-0.8739**	-0.0477
rule3	-0.0148	**0.2030**	-0.1825
rule4	-0.0064	0.0193	0.0111
rule5	-0.0681	0.1573	**0.8564**
rule6	-0.0593	**0.2512**	**-0.3652**
gambling	-0.0004	0.0018	-0.0024
Whale	-0.0037	0.0124	-0.0186
lowvalueflag	**-0.7513**	-0.1828	0.0035
midvalueflag	**0.6055**	0.1812	0.0604

Table 17: Principal component loading vectors

with the principal component, and a value of -1 has the exact opposite behavior as the principal component. Therefore, significant values lie near the value of -1 and 1. Low correlation values lie around 0. The main variance of the observation comes from the dimension low- and midvalueflag, as these have the most significant impact on PC 1. PC 2 mainly comprises rule 2, rule 3 and rule 6. This composition signifies that the observations in the data set vary significantly in these dimensions, as PC 2 explains approximately 30% of the total variance. With only 10% of the explained variance, the third principal component (PC 3) is less important and is mostly composed from rule 0, rule 5 and rule 6.

Another possibility to plot the PCA data is with biplot. It shows how strongly each feature influences a principal component. All vectors start at the same origin and their projected directions explain how much weight each of them has on a certain principal component. The angles between the individual vectors give indication about the correlation between them. The blue dots in the figure 19 represent the data points. The length of the arrows represent the influence on the principal component. A small/large angle to the PCA shows a positive/negative correlation, where a 90° angle shows no correlation.

The biplot figure 19 shows a substantial impact on dimension lowvalueflag as well as midvalueflag for the first PC, as they are parallel to the x-axis.

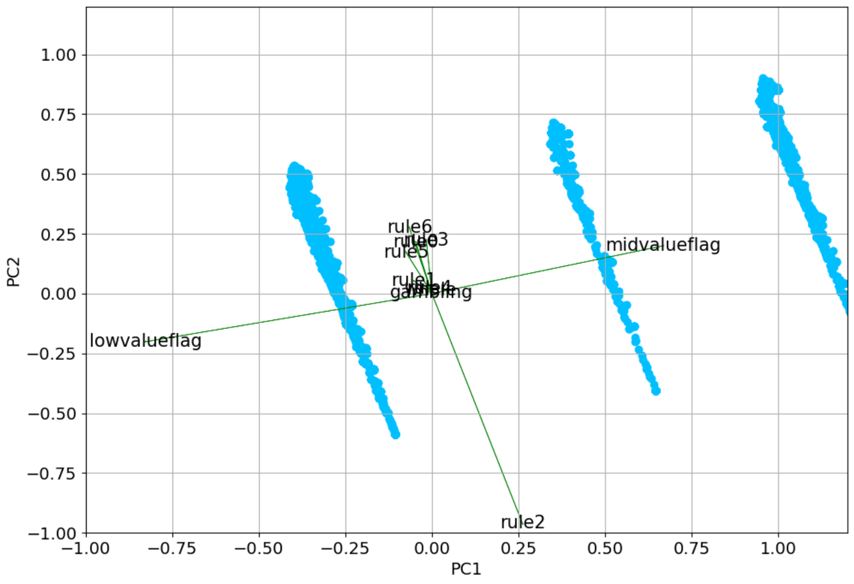


Figure 19: PCA biplot

For the second principal component, dimension rule 2 has a strong influence. Almost unimportant in the variance of the principal components 1 and 2 are the values in the middle of the biplot, such as dimensions gambling, rule 1, rule 4 and Whale. The biplot shows identically what the numbers from table 17 tell. The fact that the dimensions gambling, rule 1, rule 4 and Whale have shallow impact on the first three principal components is consistent with their overall impact on the bitcoin network. Table 14, which explained all the features already, indicated that rule 1 has 8%, rule 4 has 4%, gambling has 1.9% and Whale has 0.5% covered in all transactions. Nevertheless, rule 0 which has a coverage of 3.4% or rule 6 with coverage of 1.2% are lower but have more impact on the first three principal components. This is important to notice, as the choices of the features themselves for the machine learning part is key. Getting this information out of the data, that certain dimensions only contribute little to the variance, can give a better understanding in weighting the dimensions.

5.2.1 PCA with k-means

There are two different ways how k-means labels can be analyzed. Either run the k-means algorithm before the PCA transformation or after PCA transformation. First the process of labeling on the sample data and then performing PCA is explained below:

PCA before k-means

In the earlier subchapter, k-means labeled the data points into three clusters. These labels from k-means have been saved, and the sample data was performed on PCA. The appropriate code from the scikit-learn library is below.

K-means labeling

```
model    = sklearn.cluster.KMeans(n_clusters=3,
init='k-means++', random_state=0).fit(sample_data)
clusters = model.labels_
```

PCA data transformation

```
pca      = sklearn.decomposition.PCA(n_components =
3).fit(sample_data)
```

N_components = 3 has been selected because the first three components explain 80% of the variance. After transforming the data with PCA it was plotted in figure 20. The data points where highlighted according to their k-means labels, to show how the clusters are organized. This procedure gives the observer an overview of how the k-means clustered the data in a visual perspective. Earlier in the k-means chapter only a visualization with parallel coordinate plot was possible, because too many dimensions were involved.

The three clusters are marked with different colors in figure 20, whereby cluster three is already noted as possible black wallets. The cluster three

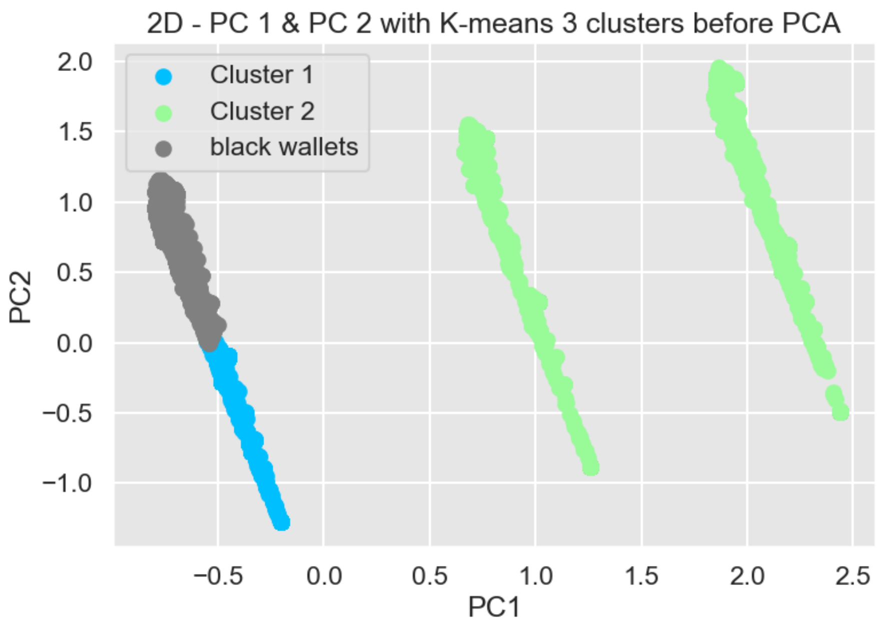


Figure 20: PC 1 and PC 2 with k-means coloring

with the possible black wallets was proven to be an effective cluster, after linking the analysis to the black listed wallets separately, documented in chapter 5.1.3.

PCA after k-means

Now the PCA analysis with k-means clustering can be done in a different perspective. Instead of clustering the data with k-means before the PCA analysis, the other way around was performed. First the data was transformed with PCA analysis. Then the reduced PCA data ran through the k-means algorithm in order to show how the clusters would change. To get the right amount of k clusters, the four internal measurement scores explained in chapter 5.1 were performed on the PCA transformed sample data.

The calculation from the internal validation measures on the PCA transformed data is displayed in figure 21.

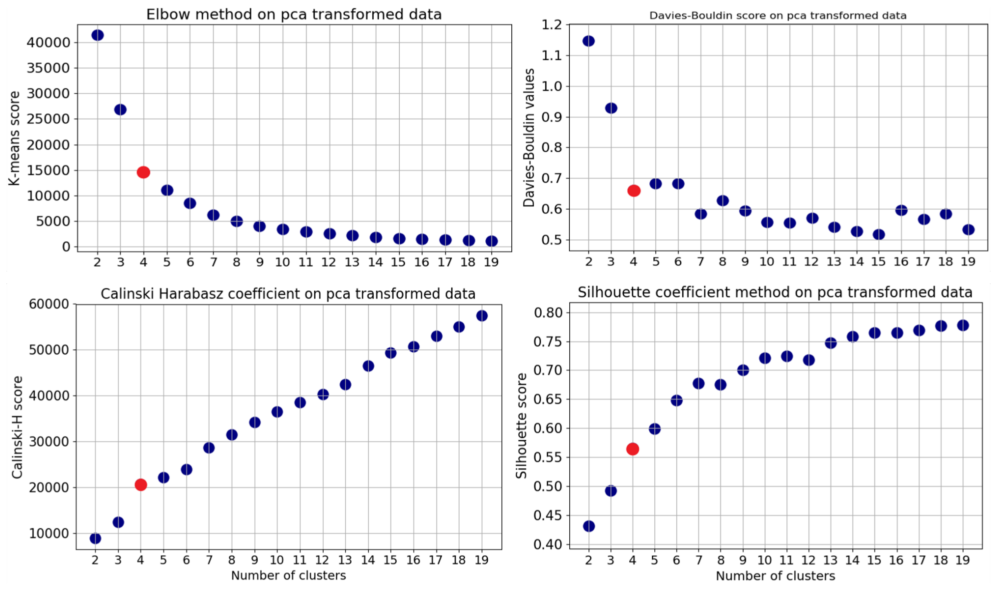


Figure 21: Internal validation methods to choose the correct k cluster size

For the two grids above (Elbow and Davies-Bouldin scores) the best number of k should be minimalized. The grids below (Silhouette and Calinski-Harabasz) scores should be maximized. Considering all the measures, one of the optimal number of k is four (k = 4). The elbow grid (upper left) shows a sharp fall and at k = 4 starts to be flatter again, which is an indication of a good cluster size. Davies-Bouldin score (upper right grid) falls till k = 4 before it slightly increases and then gets lower again. This local minimum at k = 4 is supporting the conclusion of four clusters. The left lower grid (Calinski-Harabasz) with a steadily increasing score, makes a jump between the cluster size three and four, which could lead to the conclusion that k = 4 could be a good measurement. The lower right grid (Silhouette coefficient) is steadily increasing. Between the cluster size three and four the increasing gap is larger than between other data points. For this analysis k = 4 has been selected.

To calculate the values for the boxplot, each data point from the sample was sorted in one of the new four cluster labels from the PCA - k-means analysis.

Figure 22 shows the PCA plot with the four clusters based on k-means on the PCA transformed data. The plot is visualized in 3D in order to better detect the four clusters. The three dimensions are PC 1, PC 2 and PC 3.

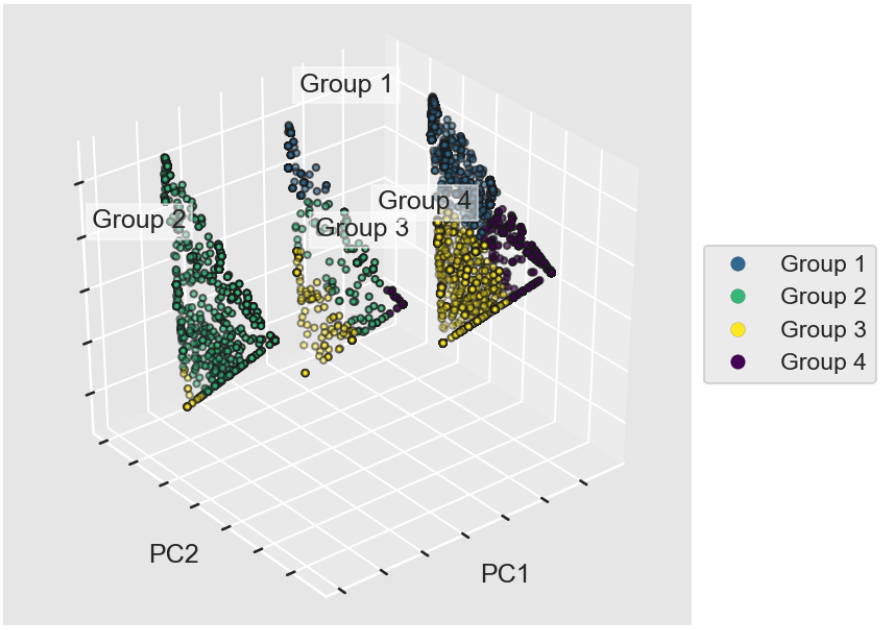


Figure 22: PC 1 and PC 2 with k-means clusters after PCA transformed data

In the figure 22 it is not visible at first sight, which cluster contains the black wallets. To have this kind of insight, the four clusters have to be analysed individually seen in figure 23 with the features distributed per cluster.

The first cluster contains 6'004 data points, which contributes 30% to the overall sample size. The second cluster contains 2'996 data points, representing 15% of the whole data set. The third cluster has 3'501 (18%) data points in it and cluster four 7'499 points (37%).

- The boxplot of cluster one (upper left plot) from figure 23 shows a slightly higher value of rule 0, rule 2, rule 3, rule 6 and a significant high value of lowvalueflag. This cluster has similar attributes as the black wallet sample which was plotted in figure 17. There rule

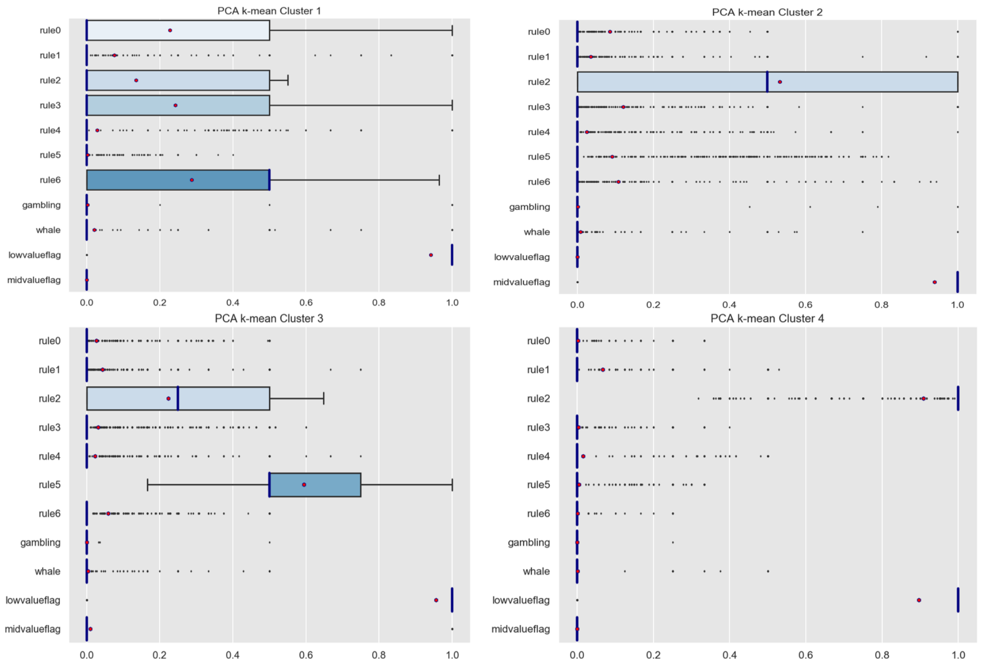


Figure 23: Boxplot PCA transformed data with k-means clustering k = 4

3 and rule 6 had higher values compared to the average observation of the data set.

- In cluster two rule 2 is increasingly present including midvalueflag=1. It seems that this cluster comprises rather normal transactions.
- Cluster three from the left lower half from figure 23 shows that rule 2 and rule 5 is significantly higher than the regular observation. Especially for rule 5, this means this kind of transaction (N:2 transaction) are quite unique and separated from the rest of the dimensions.
- Cluster four is again concentrating on rule 2, with a significant value of 1, together with a mean of 1 in the dimension lowvalueflag. With these characteristics, it can be expected that the fourth cluster comprises harmless wallets.

- All 4 boxplots show lowvalueflag and midvalueflag splitted by either 0 or 1. This can be explained because in PC 1 the two dimensions low- and midvalueflag explain a big part of the variance (figure 17). And as they explain a major part of the variance, this leads to bigger influence on the distance measures between this two dimensions. That in turn influences k-means, as k-means is distance based driven.

5.2.2 Validation

As explained in the k-means chapter 5.1, when using the k-means algorithm on a data set, it is possible with the predict function to use the k-means on a new data set. This has again been performed in the same way as documented before. So, the US government blacklisted wallet data set was transformed with a principal component analysis and after that run through the k-means algorithm with the predict function. On the boxplots figure 23 from our original sample the conclusion was that cluster one should contain the most black wallets. Now calculating each cluster with the respective data points gives a different conclusion, see table 18.

US blacklisted wallets		
	Number of data points	Percentage of total
Cluster 1	5’251	18%
Cluster 2	5’179	18%
Cluster 3	2’568	9%
Cluster 4	15’583	55%
Total	28’581	100%

Table 18: K-means predict on black wallets

It was expected that the US blacklisted wallets will show the largest amount of data points in cluster one. However, as displayed in table 18, the cluster one only got 18% of the black wallets in it. Most of the points landed in cluster four. Due to this observation, a PCA clustering with k-means is not helpful in clustering the black wallets in one group.

5.3 DBSCAN

The algorithm in section 5.2 (PCA) tries to find hidden linear correlations between variables. If the features are not linear but have another shape, like a spiral, PCA will not identify the best groups. From the PCA figure 20, long structures are visible in the data set, which might indicates that DBSCAN would be helpful in such a scenario. For these kinds of shapes, density-based spatial clustering of applications with noise (DBSCAN) might be a better choice.

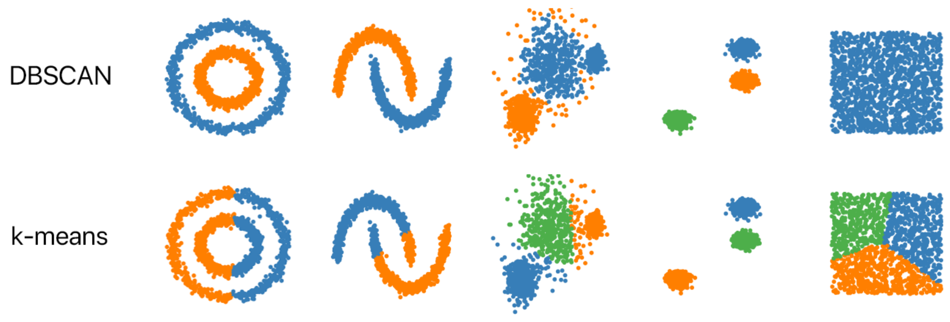


Figure 24: DBSCAN and k-means algorithm compared, graph by Mattt (2020)

For a better understanding how k-means and DBSCAN algorithm differ from each other, figure 24 shows different data structures and the clustering performance. When the data structure represents donut circles, DBSCAN is able to identify this groups separately, where else k-means due to the distance measure calculations group this circles in half. If the data set consists of convex/concave strings near to each other, then DBSCAN can again detect the forms, but k-means gets confused by the distances of the two objects so near from each other. In case the data points are blobs of different densities, k-means can lead in some cases to a better performance, since DBSCAN has difficulties with varying densities. Concentrated blobs of data points which are far from each other, are easily detectable by both clustering algorithms. If all points lie near to each other without a shape or density difference, DBSCAN recognize this as one group, as the density is the same. K-means group this cluster into k numbers of groups, which has been predefined in the beginning. Overall concluding, DBSCAN works with densities, and splits clusters between dense regions. The shape of the data is not relevant (Mattt (2020)).

DBSCAN is scalable on huge samples and medium cluster size, and is very favorable when the underlying data has a special shape, where the standard Euclidean distance is not the right metric. One of its best properties is that the number of clusters does not have to be defined beforehand, as was the case with k-means. Instead, the algorithm needs to be pre-filled with two parameters. These parameters are called epsilon (ϵ or *eps*) and the minimum sample size per cluster (*min_points*) (Scikit-learn (2020)).

The *eps* parameter defines the radius of the neighborhood around a point x. This is called the eps-neighborhood of x. X is randomly defined when starting the algorithm.

Min_points parameter defines how many data points need to be next to each other, to form an own cluster. When there are fewer data points than min_points in the neighborhood, then they are considered as noise or get attached to another cluster. The neighborhood distance is defined by the ϵ distance.

DBSCAN scores to be deterministic. Given the same data set and the same density parameters, DBSCAN returns the same clusters. This is contrary to k-means, which can result in different clusters each run.

The algorithm works as follows:

- First, the data set is divided into n dimensions.
- Secondly, DBSCAN builds n dimensional shape around each datapoint and checks how many data points fall within that shape.

- In the third step, DBSCAN expands iteratively each of these shapes by counting the other nearby data points. Figure 25 displays an overview how the points are assigned to one cluster. The red dots are considered core points, as they include the minimum number of points required for a cluster (min_points) within the defined distance (ϵ). The yellow points display border points and they do not count as core points, because within the ϵ distance, not enough points are available. However, as these border points still count to the core point cluster, they belong to the cluster. The blue point illustrates a noise point, as within the reachable ϵ distance no other point can be reached. In scikit-learn library these noise points get labeled with -1, all other points belonging to a cluster get a label of 0 and above. If two core points are close (within the eps distance), then they combine to a cluster (Lutins (2017)).

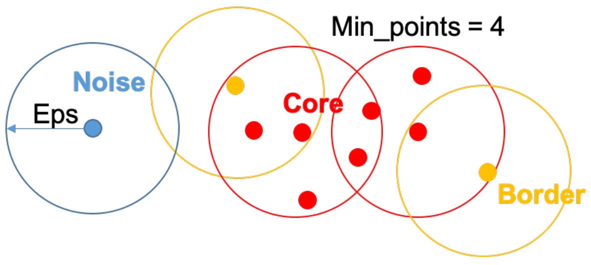


Figure 25: DBSCAN explained, graph based on Lutins (2017)

To start with the DBSCAN on the sample data the ϵ parameter and min_sample parameter needed to be defined. For DBSCAN in Python the default parameters set are 0.5 for eps and 50 for min_points. The whole command from sklearn.cluster is as follows (Pedregosa et al. (2011)):

```
DBSCAN(eps = 0.5, min_samples = 50, metric = 'euclidean')
```

With eps = 0.5 and min_points = 50, there are five clusters and 67 noise points from the sample. The 67 noise points are outliers and are not clustered in any of the clusters. To measure the cluster quality again some scores can be calculated. The Silhouette score is 0.28, the

Calinski-Harabasz score is 3'007 and the Davies-Bouldin score is 1.33. Compared to the k-means score (Silhouette = 0.43, Calinski-Harabasz = 9'656, Davies-Bouldin = 1.18) all scores are worse in DBSCAN. Silhouette and Calinski-Harabasz should be maximized and Davies-Bouldin minimized.

The parameters eps = 0.5 and min_points = 50 are a compromise of the three different scores. When starting to lower the min_sample parameter by keeping the eps parameter of 0.5, the Silhouette score starts to increase as well as the Calinski-Harabasz score. The Davies-Bouldin score increases as well, which is a not favorable. The number of noise points decreases, as data points have a lower barrier to count to a cluster. With increasing the min_points parameter, the Davies-Bouldin scores lower, however the Sillhouette and Calinski-Harabasz scores lower as well.

On the other hand when starting to modify the eps parameter by keeping the min_points on the level of 50, then the Silhouette coefficient increases when lowering eps score. By lowering the eps score, the Davies-Bouldin score is lowering as well. As a result, the number of clusters increase and so do the noise points. Increasing the eps parameter will increase the Silhouette coefficient and the Calinski-Harabasz score. The number of clusters is decreasing and the noise points reduce as well. An analysis about this behavior is attached in the appendix table 20.

The rest of the chapter continues with eps = 0.5 and min_samples = 50 to have an acceptable score for all of the three internal measurement scores.

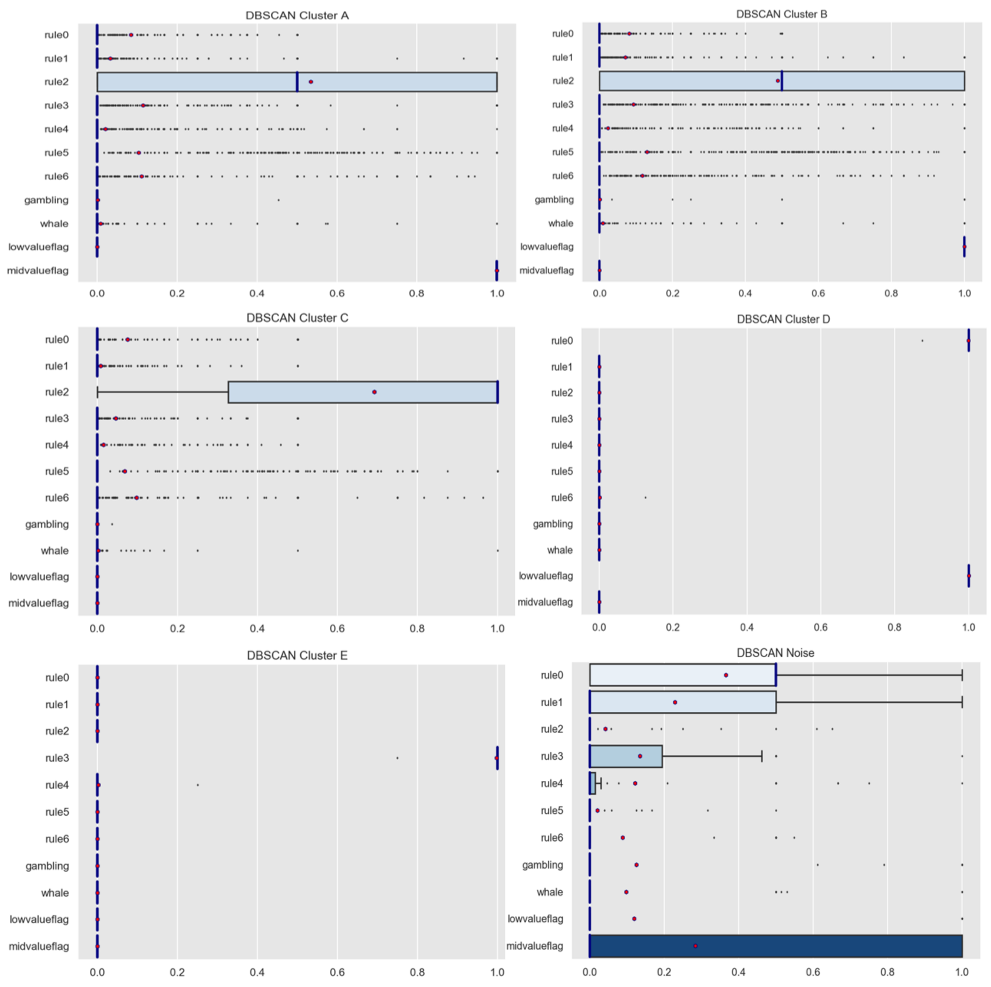


Figure 26: DBSCAN 5 clusters with eps = 0.5 and min_points = 50

Figure 26 displays the distribution overview of all five clusters with the DBSCAN. The distribution of the noise points is shown in the lowest right grid.

- Cluster A has an increased value in rule 2 with a large variance. In addition, the value of midvalueflag is 1.
- Cluster B has also a slightly higher value of rule 2 as well, and the value of lowvalueflag is 1.
- Cluster C's property also shows significant high values in rule 2, and includes only wallets with high values, as neither dimension

lowvalueflag nor midvalueflag is visible in the boxplot.

- Cluster D contains all miners with a high value of rule 0 and low-valueflag.
- Cluster E has significant high values in rule 3 and only includes wallets with high values above 10 BTC.
- The noise dimension has in most dimensions significant values. In rule 0, rule 1, rule 3 and midvalueflag the values are greater than zero, which indicates, that these observation are more likely to include miner and 1:1 transactions with values of more than 1 Bitcoin.

This overview leads to three conclusions.

1. Rule 2 is very often occurring in the whole bitcoin data set. DBSCAN separates the rule with the values (cluster A: rule 2 and midvalues; cluster B: rule 2 and low values; cluster C: rule 2 and highvalues).
2. DBSCAN is unable to detect in this data set a cluster similar to k-means, where rule 3 and rule 6 are higher in order to get a lead for possible black wallets.
3. Rule 0 and rule 3 have a very own data structure, different from the rest. They were separated by DBSCAN in cluster D and E.

PCA before DBSCAN

To see a graphical overview how the DBSCAN clusters the points, a PCA analysis with the labels from the DBSCAN has been plotted in figure 27.

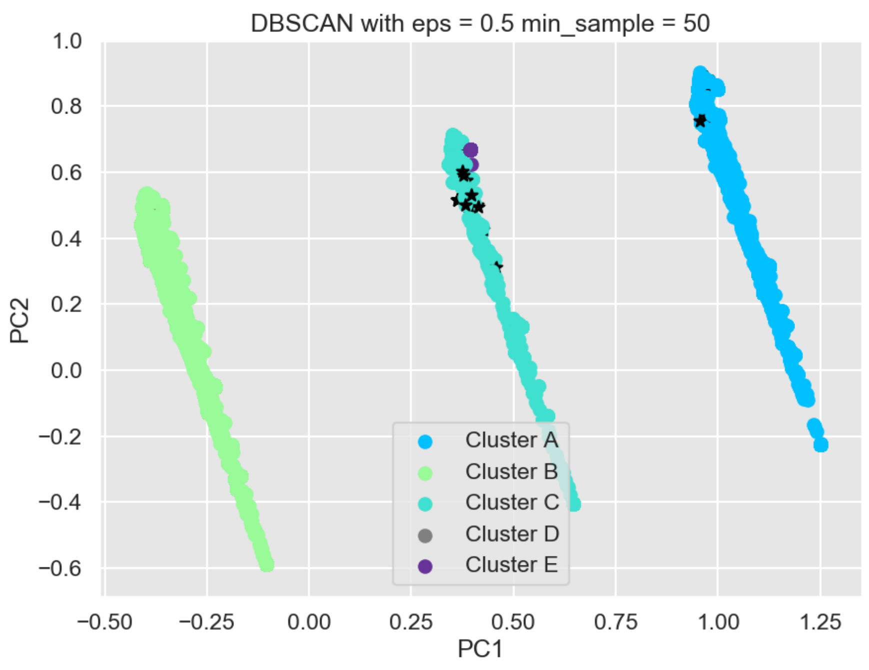


Figure 27: PC 1 and PC 2 with DBSCAN coloring

The first three clusters A, B and C seem for the observer larger than the clusters D and E, which are both barely visible in the plot. The largest cluster is the one to the left, cluster B with 15’609 points. This cluster B represents 78% of all the data points. Cluster A has 2’831 and cluster C 1’260 data points. Almost negligible is cluster D with 109 data points and cluster E with 124 data points. Cluster D was the group with significant values in rule 0 and lowvalueflag. Cluster E was the group concentrating on significant values from dimension rule 3 and high values. DBSCAN was able to detect both these two minor groups an own density groups, which was not the case in k-means. However, the largest group cluster B was separated in the k-means algorithm, where one part of this cluster B was the potential black wallets (figure 20).

PCA after DBSCAN

To compare the above results with a different approach the data was transformed with PCA first. After that, DBSCAN labeled the data with the same parameter eps = 0.5 and min_points = 50. One step later, the data with the new labels was plotted. Exactly three clusters are detected, with no noise points. This is displayed in figure 28.

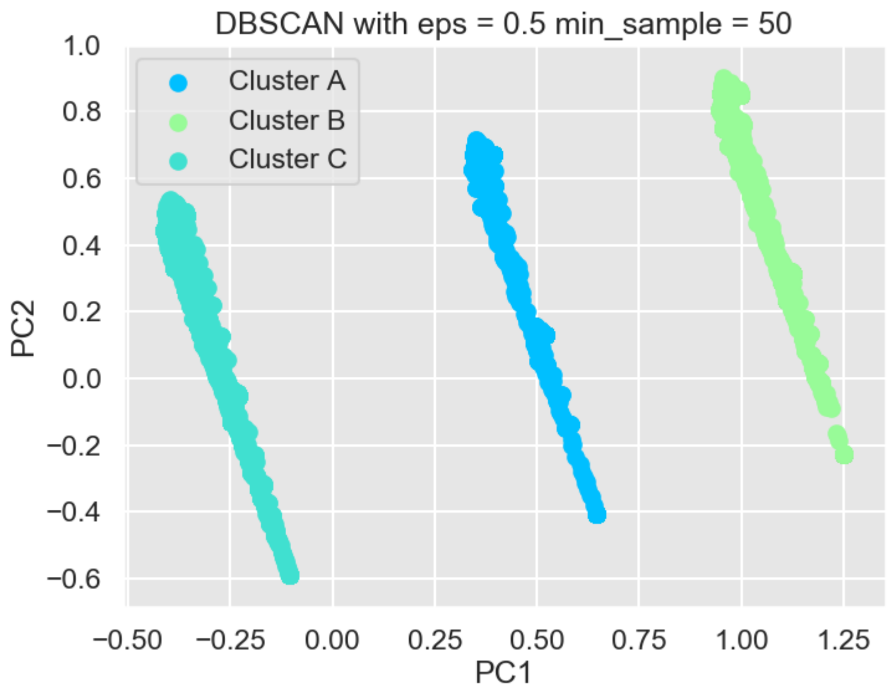


Figure 28: PC 1 and PC 2 with DBSCAN clustering after PCA transformed data (eps = 0.5, min_points = 50)

Where k-means before was unable to locate the three long structured shapes, DBSCAN did a much better job in detecting these density strings. Cluster A contains 1'424 (7%) data points, cluster B 2'850 (14% data points) and cluster C 15'726 (79%) data points. In the internal measurement scores, Silhouette coefficient scored with 0.396, Calinski-Harabasz with 6916.3 points and DB with 1.2 score. All three scores are performing better than the DBSCAN scores before PCA clustering.

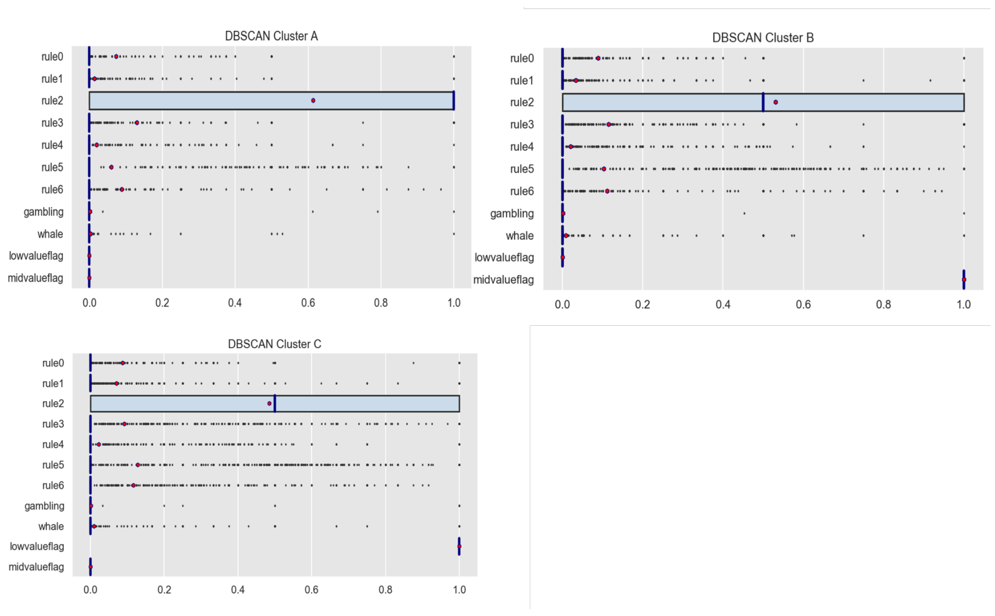


Figure 29: DBSCAN 3 clusters with eps = 0.5 and min_points = 50

Figure 29 documents the three clusters, where rule 2 is again significant in all of them together with a value.

From the eye of the observer, figure 28 looks like a perfect clustering. But this needs to be treated carefully. The goal of the algorithm was to find black wallets. However, only viewing the clustering graphically will not tell, which algorithm between k-means and DBSCAN is better. Both clustered the data differently and it is up to the observer, to choose the best algorithm to use. The boxplot overview from DBSCAN in figure 26 led to the conclusion, that it it did not separate the potentially black wallets from the rest of the population. Nevertheless, DBSCAN plotted the three strings accurately in the PCA plot.

In the other two chapters k-means 5.1 and PCA 5.2, a predict function on the US blacklisted wallets was used for an external evaluation. In the scikit-learn library the DBSCAN function does not provide a predict function for the US blacklisted wallets. Therefore, the conclusion needs to be based on the overall picture from given informations. Informations are retrieved from internal validation scores from DBSCAN clustering, the boxplot overview about all clusters and the size of the clusters itself. The validation scores before the PCA transformation were better with k-means than with DBSCAN. Even though they performed better after PCA transformation, the k-means internal measurement scores were still higher. The boxplot figure 26 did not show similarities to the black wallet properties and also the cluster sizes did not match the cluster size for k-means. All in all, DBSCAN does not seem to perform well in this specific task of locating potential black wallets during clustering. What should be taken into account, is that in both analysis from k-means and DBSCAN the dimension gambling and Whale wallets did not raise significant impact in any of the boxplots studies above. Also in the PCA loadings table 17, both dimensions did not contribute to a high explained variance of the model. It can therefore be said, that the connection to gambling or Whale wallets was not significant for black wallets. On the other hand the rule dimensions and the value dimensions had strong influence on the clustering results.

6 Conclusion

In this thesis explained is, how to uncovering patterns in the bitcoin block-chain by using unsupervised machine learning. Applying machine learning to such a vast data set of 1.1 TB, a logical and exact pre-processing of the data needs to be carried out. The central part of the pre-processing was linking different addresses to a wallet that is owned by the same person. Based on these wallets, machine learning was performed. Machine learning itself is very experimental and the algorithms need to be carefully chosen for the underlying data set.
One of the most famous benchmarking algorithms is k-means, which performs well on a large data set. With a final sample data set after pre-processing 20'000 wallets, this algorithm has been applied. K-means was able to cluster three groups, where one of them had similar attributes to selected wallets which were blacklisted by the US government. With the predict function, 88% of the US blacklisted wallets landed in one of these clusters. This is an indication that k-means is able to detect patterns in the bitcoin blockchain and to split black wallets from the rest of the population. It is interesting to see, that the black wallets in the k-means clusters were in the same group as miners. This is an indication that transactions are carried out by the miners shows a similar pattern to the owners of the black wallets.
To confirm the k-means result, a principal component analysis was performed to reduce the 11 features down to three principal components. Applying k-means on the PCA data set did not enable the desired clustering between the black wallets and the rest of the wallets. So PCA was not supporting the hypothesis in detecting the nature of the black wallets.
Another algorithm DBSCAN was performed on the data set. DBSCAN found different regions of densities from the feature with the most variance which is rule 2 (1 to N transaction) in combination with the value amount of the transaction (low, mid and high value). Disregarding this clustering of rule 2 and value, the DBSCAN was unable to locate the region of potential black wallets. This leads to the conclusion that DB-

SCAN is not the right choice for such a task and neither is PCA. Whereas k-means was able to locate potentially blacklisted wallets and a tinier clustering within this blacklisted wallet cluster leads with a high percentage to actual shady wallet owners.
Therefore, this thesis concludes that patterns of potential black wallets may be detected with the k-means algorithm.

7 Further research

The task of determining the quality of the clustering algorithm results is not easy. To measure the output of the clusters more accurately, supervised machine learning is recommended. This thesis includes only a limited amount of off-chain data like the gambling addresses, the Whale addresses and the US government blacklisted addresses for the machine learning analysis. But chapter 3 referenced to papers, which used web-scraping in order to locate as many wallets as possible with real world identities. This approach enables finer granularity in detecting the nature of the wallets and can certainly give the machine learning algorithm new information for clustering. Running a supervised machine learning approach can also work with labeled data sets from companies like Chainalysis (chapter 2.4). In unsupervised machine learning theory, hierarchical clustering is less effective in large data sets. However, there exist modifications like BIRCH algorithm (balanced iterative reducing and clustering using hierarchies) which performs exceptionally well on large data sets (Zhang et al. (1997)). This kind of algorithm can also be included for a pattern recognition analysis on the bitcoin network. Besides, this thesis only included a limited amount of features. This can be expanded with features which could give interesting contributions to the model next to the one analyzed in this thesis.

References

Androulaki, E., Karame, G. O., Roeschlin, M., Scherer, T. and Capkun, S. (2013), Evaluating user privacy in bitcoin, *in* 'International Conference on Financial Cryptography and Data Security', Springer, pp. 34–51.

Arthur, D. and Vassilvitskii, S. (2006), k-means++: The advantages of careful seeding, Technical report, Stanford.

Baarsch, J. and Celebi, M. E. (2012), Investigation of internal validity measures for k-means clustering, *in* 'Proceedings of the international multiconference of engineers and computer scientists', Vol. 1, sn, pp. 14–16.

Bitcoin.Wiki (2019), 'Segregated witness'. https://en.bitcoin.it/wiki/Segregated$_{\text{W}}$itness, accessed on 08/11/2020.

Bitcoin.Wiki (2020), 'Common vulnerabilities and exposures'. https://en.bitcoin.it/wiki/Common$_{\text{V}}$ulnerabilities$_{\text{a}}$nd$_{\text{E}}$xposures, accessed on 08/11/2020.

Bitfury (2020*a*), 'Bitfury'. https://www.linkedin.com/company/bitfury/, accessed on 08/11/2020.

Bitfury (2020*b*), 'International bitcoin flows analytics report 2013 — q1 2020'. https://crystalblockchain.com/assets/reports/International$_{\text{B}}$itcoin$_{\text{F}}$lows$_{\text{A}}$nalytics$_{\text{R}}$eport$_{2}$013$_{\%}$E2%80%94$_{\text{Q}}$1$_{2}$020.pdf, accessed 08/11/2020.

Bitinfocharts (2020), 'Bitcoin rich list'. https://bitinfocharts.com/top-100-richest-bitcoin-addresses-1.html, accessed on 07/02/2020.

Blockchain.com (2020), 'Blockchain charts'. https://www.blockchain.com/charts/blocks-size, accessed on 08/11/2020.

BTC.com (2020), ‘Blockchain pools’. `https://raw.githubusercontent.com/btccom/Blockchain-Known-Pools/650a92227bf65b06ff0a5b58bb57c13856a3babf/pools.json` and `https://raw.githubusercontent.com/blockchain/Blockchain-Known-Pools/29ab27c844ebdb63110f8783f73b9decd4abc221/pools.json`, accessed on 08/11/2020.

Burks, L. S., Cox, A. E., Lakkaraju, K., Boyd, M. J. and Chan, E. (2017), Bitcoin address classification., Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States).

Cdecker (2017), ‘List of addresses for gambling in bitcoin’. `https://stackoverflow.com/questions/42049617/list-of-addresses-for-gambling-in-bitcoin?rq=1`, accessed on 08/11/2020.

Chainalysis (2020), ‘Chainalysis inc.’. `https://www.linkedin.com/company/chainalysis/`, accessed on 08/11/2020.

Chawathe, S. S. (2019), Clustering blockchain data, *in* ‘Clustering Methods for Big Data Analytics’, Springer, pp. 43–72.

Cipher Trace, . (2020), ‘Ciphertrace’. `https://www.linkedin.com/company/ciphertrace/`, accessed on 08/11/2020.

Computerworld (2019), ‘Elliptic traces bitcoin to hunt dark web criminals for the fbi’. `https://www.computerworld.com/article/3558441/elliptic-traces-bitcoin-to-hunt-dark-web-criminals-for-the-fbi.html`, accessed on 06/23/2020.

Conti, M., Sandeep Kumar, E., Lal, C. and Ruj, S. (2018), ‘A survey on security and privacy issues of bitcoin’, *IEEE Communications Surveys Tutorials* **20**(4), 3416–3452.

Crystal Blockchain, . (2020), ‘Crystalblockchain’. https://crystalblockchain.com/, accessed on 08/11/2020.

Dabbura, I. (2018), ‘K-means clustering - algorithm, applications, evaluation methods, and drawbacks’. https://imaddabbura.github.io/post/kmeans-clustering/, accessed on 08/11/2020.

Daniel, W. W. and Cross, C. L. (2018), *Biostatistics: a foundation for analysis in the health sciences*, Wiley.

Derksen, L. (2016), ‘Visualising high-dimensional datasets using pca and t-sne in python’. https://towardsdatascience.com/visualising-high-dimensional-datasets-using-pca-and-t-sne-in-python-8ef87e7915b, accessed on 08/11/2020.

Douceur, J. R. (2002), The sybil attack, *in* ‘International workshop on peer-to-peer systems’, Springer, pp. 251–260.

Drake, J. D. and Worsley, J. C. (2002), *Practical PostgreSQL*, " O’Reilly Media, Inc.".

Elliptic (2020), ‘Elliptic’. https://www.elliptic.co/, accessed on 08/11/2020.

Ermilov, D., Panov, M. and Yanovich, Y. (2017), Automatic bitcoin address clustering, Vol. 2017-December, Institute of Electrical and Electronics Engineers Inc., pp. 461–466.

Fleder, M., Kester, M. S. and Pillai, S. (2015), ‘Bitcoin transaction graph analysis’, *arXiv preprint arXiv:1502.01657* .

Goldstein, M. and Uchida, S. (2016), ‘A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data’, *PLoS ONE* **11**.

Harlev, M. A., Yin, H. S., Langenheldt, K. C., Mukkamala, R. and Vatrapu, R. (2018), Breaking bad: De-anonymising entity types on the bitcoin blockchain using supervised machine learning, Hawaii International Conference on System Sciences.

Harrigan, M. and Fretter, C. (2016), The unreasonable effectiveness of address clustering, *in* '2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)', IEEE, pp. 368–373.

Harris, B. and Zero, P. (2019), *Bitcoin and Lightning Network on Raspberry Pi*, Apress.

Hirshman, J., Huang, Y. and Macke, S. (2013), 'Unsupervised approaches to detecting anomalous behavior in the bitcoin transaction network', *3rd ed. Technical report, Stanford University* .

Holland, S. M. (2008), 'Principal components analysis (pca)', *Department of Geology, University of Georgia, Athens, GA* pp. 30602–2501.

Huang, B., Liu, Z., Chen, J., Liu, A., Liu, Q. and He, Q. (2017), 'Behavior pattern clustering in blockchain networks', *Multimedia Tools and Applications* **76**, 20099–20110.

James, G., Witten, D., Hastie, T. and Tibshirani, R. (2013), *An introduction to statistical learning*, Vol. 112, Springer.

Janda, A. (2020), 'Bitcoin block explorer with address grouping and wallet labeling'. `https://www.walletexplorer.com/`, accessed on 08/11/2020.

Jardine, E. (2015), 'The dark web dilemma: Tor, anonymity and online policing', *Global Commission on Internet Governance Paper Series* (21).

Johnson, R. A., Wichern, D. W. et al. (2002), *Applied multivariate statistical analysis*, Vol. 5, Prentice hall Upper Saddle River, NJ.

Kalodner, H., Goldfeder, S., Chator, A., Möser, M. and Narayanan, A. (2017), 'Blocksci: Design and applications of a blockchain analysis platform', *arXiv preprint arXiv:1709.02489* .

Koblitz, N., Menezes, A. and Vanstone, S. (2000), 'The state of elliptic curve cryptography', *Designs, codes and cryptography* **19**, 173–193.

Kondor, D., Pósfai, M., Csabai, I. and Vattay, G. (2014), 'Do the rich get richer? an empirical analysis of the bitcoin transaction network', *PLoS ONE* **9**.

Lemieux, P. (2013), 'Who is satoshi nakamoto?', *Regulation* **36**(3), 14.

Llewelyn, F. (2019), 'Perform an exploratory data analysis'. `https://github.com/OpenClassrooms-Student-Center/Multivariate-Exploratory-Analysis/blob/master/functions.py`, accessed on 08/11/2020.

Lutins, E. (2017), 'Dbscan: What is it? when to use it? how to use it.'. `https://medium.com/@elutins/dbscan-what-is-it-when-to-use-it-how-to-use-it-8bd506293818`, accessed on 08/11/2020.

Mackay, B. (2019), 'Evaluation of security in hardware and software cryptocurrency wallets'.

Maesa, D. D. F., Marino, A. and Ricci, L. (2016), Uncovering the bitcoin blockchain: An analysis of the full users graph, Institute of Electrical and Electronics Engineers Inc., pp. 537–546.

Maple Tech. International LLC., i. (2020), 'Sample size calculator'. `https://www.calculator.net/sample-size-calculator.html`, accessed on 08/11/2020.

Mattt (2020), 'Dbscan'. `https://github.com/NSHipster/DBSCAN`, accessed on 08/11/2020.

Matzutt, R., Kalde, B., Pennekamp, J., Drichel, A., Henze, M. and Wehrle, K. (2020), 'How to securely prune bitcoin's blockchain'.

Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G. M. and Savage, S. (2013), A fistful of bitcoins: Characterizing payments among men with no names, pp. 127–139.

Nakamoto, S. et al. (2008), ‘Bitcoin: A peer-to-peer electronic cash system.(2008)’. `http://www.bitcoin.org/bitcoin.pdf`, accessed on 08/11/2020.

Nie, J.-Y., of Electrical, I., Engineers, E. and Society, I. C. (2017), ‘A first estimation of the proportion of cybercriminal entities in the bitcoin ecosystem using supervised machine learning’, *IEEE International Conference on Big Data : proceedings : Dec 11- 14, 2017, Boston, MA, USA* .

pandas development team, T. (2020), ‘pandas-dev/pandas: Pandas’. `https://doi.org/10.5281/zenodo.3509134`, accessed on 08/11/2020.

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

Pinker, S. (1997), ‘How the mind works. 1997’, *NY: Norton* .

QuantaBytes (2020), ‘A survey of bitcoin transaction types’. `https://www.quantabytes.com/articles/a-survey-of-bitcoin-transaction-types`, accessed on 08/11/2020.

Reid, F. and Harrigan, M. (2011), ‘An analysis of anonymity in the bitcoin system’, *IEEE International Conference on Privacy, Security, Risk, and Trust* .

Romiti, M., Judmayer, A., Zamyatin, A. and Haslhofer, B. (2019), ‘A deep dive into bitcoin mining pools: An empirical analysis of mining shares’, *arXiv preprint arXiv:1905.05999* .

Ron, D. and Shamir, A. (2013), Quantitative analysis of the full bitcoin transaction graph, *in* 'International Conference on Financial Cryptography and Data Security', Springer, pp. 6–24.

SatoshiDice (2020), 'Satoshidice'. `https://www.satoshidice.com/rules`, accessed on 08/11/2020.

Scikit-learn, . (2020), '2.3. clustering'. `https://scikit-learn.org/stable/modules/clustering.html`, accessed on 08/11/2020.

ShenTu, Q. and Yu, J. (2015), 'Research on anonymization and de-anonymization in the bitcoin system', *arXiv preprint arXiv:1510.07782* .

Sommer, D. (2019), 'Processing bitcoin blockchain data using a big data-specific framework'.

Spagnuolo, M., Maggi, F. and Zanero, S. (2014), Bitiodine: Extracting intelligence from the bitcoin network, *in* 'International Conference on Financial Cryptography and Data Security', Springer, pp. 457–468.

U.S.Government (2020*a*), 'Specially designated nationals and blocked persons list (sdn) human readable lists'. `https://www.treasury.gov/resource-center/sanctions/SDN-List/Pages/default.aspx`, accessed on 08/11/2020.

U.S.Government (2020*b*), 'United states district court for the district of columbia'. `https://www.justice.gov/opa/press-release/file/1253491/download`, accessed on 08/11/2020.

Wang, L. and Liu, Y. (2015), Exploring miner evolution in bitcoin network, *in* J. Mirkovic and Y. Liu, eds, 'Passive and Active Measurement', Springer International Publishing, Cham, pp. 290–302.

Weber, M., Domeniconi, G., Chen, J., Weidele, D. K. I., Bellei, C., Robinson, T. and Leiserson, C. E. (2019), 'Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics', *arXiv preprint arXiv:1908.02591* .

YahooFinance (2020), 'Bitcoin usd ccc coinmarketcap'. `https://finance.yahoo.com/quote/BTC-USD?p=BTC-USD`, accessed on 08/11/2020.

Yin, H. H. S., Langenheldt, K., Harlev, M., Mukkamala, R. R. and Vatrapu, R. (2019), 'Regulating cryptocurrencies: A supervised machine learning approach to de-anonymizing the bitcoin blockchain', *Journal of Management Information Systems* **36**, 37–73.

Zhang, T., Ramakrishnan, R. and Livny, M. (1997), 'Birch: A new data clustering algorithm and its applications', *Data Mining and Knowledge Discovery* **1**(2), 141–182.

Zhang, Y. (2010), *New advances in machine learning*, BoD–Books on Demand.

Zola, F., Eguimendia, M., Bruse, J. L. and Urrutia, R. O. (2019), Cascading machine learning to attack bitcoin anonymity, Institute of Electrical and Electronics Engineers Inc., pp. 10–17.

List of Figures

List of Tables

List of Abbreviations

ASCII	American Standard Code for Information Interchange
BCSM	Between cluster scatter matrix
BTC	Bitcoin currency
CH	Calinski-Harabasz
CIA	Central Intelligence Agency
CSV	Comma-separated values
DB	Davies-Bouldin
DBSCAN	Density-based spatial clustering of applications with noise
DEA	Drug Enforcement Administration
DOS	Denial-of-service
ϵ, eps	Epsilon (parameter of DBSCAN)
FBI	Federal Bureau of Investigation
GB	Giga Byte
IP	Internet protocol
IRS	Internal Revenue Service
JSON	JavaScript Object Notation
KB	Kilo Byte
P2P	Peer-to-peer
P2PKH	Pay-to-Public-Key-Hash
P2SH	Pay-to-Script-Hash
P2WPKH	Pay to Witness Public Key Hash
P2WSH	Pay to Witness Script Hash
PCA, PC	Principal Component Analysis, Principal Component
PHP	Hypertext preprocessor
RAM	Random-access memory
SQL	Structured Query Language
SSE	Sum of squared errors
TB	Terra Byte
TOR	The Onion Router
Txid	Transaction
UTXO	Unspent transaction output
WCSM	Within cluster scatter matrix

A Appendix

Genesis bitcoin block JSON style

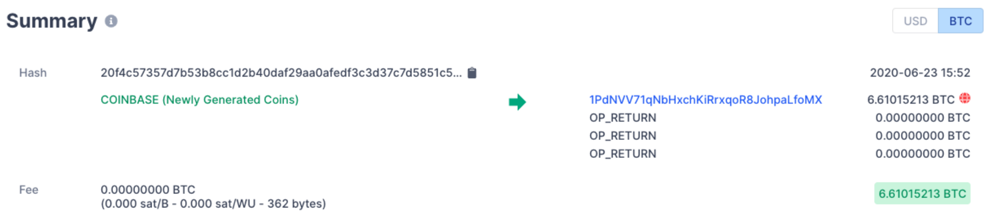
```
{
 "ver":1,
 "next_block":[
 ],
 "time":1231006505,
 "bits":486604799,
 "fee":0,
 "nonce":2083236893,
 "n_tx":1,
 "size":285,
 "block_index":0,
 "main_chain":true,
 "height":0,
 "weight":1140,
 "tx":[
    {
      "hash":"4a5e1e4baab89f3a32518a88c31bc87f618f76673e2c c77ab2127b7afdeda33b",
      "ver":1,
      "vin_sz":1,
      "vout_sz":1,
      "size":204,
      "weight":816,
      "fee":0,
      "relayed_by":"127.0.0.1",
      "lock_time":0,
      "tx_index":0,
      "double_spend":false,
      "result":0,
      "balance":0,
      "time":1231006505,
      "block_index":0,
      "block_height":0,
      "inputs":[
           {
              "sequence":4294967295,
              "witness":"",
              "script":"04ffff001d0104455468652054696d65732030332
                   f4a616e2f32303039204368616e63656c6c6f72206
                   f6e206272696e6b206f66207365636f6e64206261696c6
                   f757420666f722062616e6b73",
              "index":0
           }
        ],
       "out":[
            {
               "type":0,
               "spent":false,
               "value":5000000000,
               "spending_outpoints":[

               ],
               "n":0,
               "addr":"1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa",
               "tx_index":0,
               "script":"4104678afdb0fe5548271967f1a67130b7105cd6
                    a828e03909a67962e0ea1f61deb649f6bc3f4cef38c4f35504e51
                    ec112de5c384df7ba0b8 d578a4c702b6bf11d5fac"
            }
         ]
      }
   ],
   "hash":"000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f",
   "prev_block":"0000000000000000000000000000000000000000000000000000000000000000",
   "mrkl_root":"4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b"
}
```

Heuristics example

All the examples have been selected from bitcoin block 636'000 and copied from the website blockchain.com.

Rule 0, coinbase transaction

20f4c57357d7b53b8cc1d2b40daf29aa0afedf3c3d37c7d5851c533f9478f884



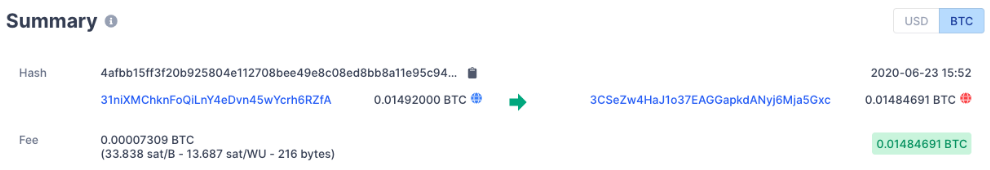
Summary

USD BTC

Hash	20f4c57357d7b53b8cc1d2b40daf29aa0afedf3c3d37c7d5851c5...			2020-06-23 15:52
	COINBASE (Newly Generated Coins)	➡	1PdNVV71qNbHxchKiRrxqoR8JohpaLfoMX	6.61015213 BTC
			OP_RETURN	0.00000000 BTC
			OP_RETURN	0.00000000 BTC
			OP_RETURN	0.00000000 BTC
Fee	0.00000000 BTC (0.000 sat/B - 0.000 sat/WU - 362 bytes)			6.61015213 BTC

Rule 1, 1:1 transaction

4afbb15ff3f20b925804e112708bee49e8c08ed8bb8a11e95c944da25a192e5f



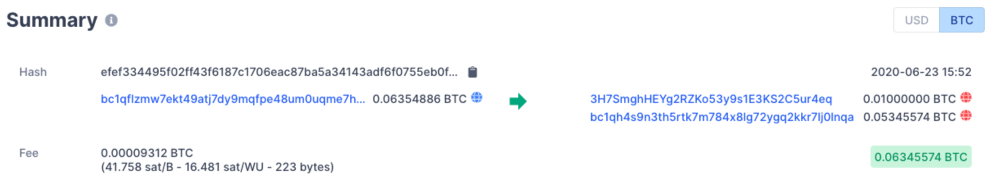
Summary

USD BTC

Hash	4afbb15ff3f20b925804e112708bee49e8c08ed8bb8a11e95c94...				2020-06-23 15:52
	31niXMChknFoQiLnY4eDvn45wYcrh6RZfA	0.01492000 BTC	➡	3CSeZw4HaJ1o37EAGGapkdANyj6Mja5Gxc	0.01484691 BTC
Fee	0.00007309 BTC (33.838 sat/B - 13.687 sat/WU - 216 bytes)				0.01484691 BTC

Rule 2, 1:2 transaction

efef334495f02ff43f6187c1706eac87ba5a34143adf6f0755eb0fcd975246f3



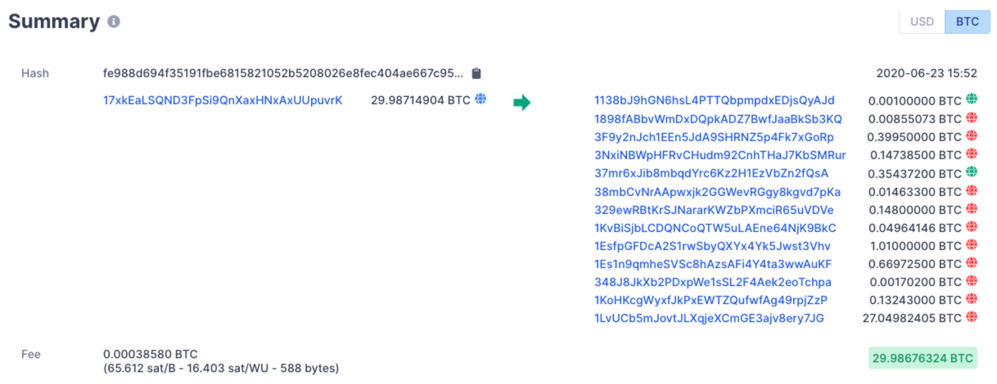
Summary

USD BTC

Hash	efef334495f02ff43f6187c1706eac87ba5a34143adf6f0755eb0f...				2020-06-23 15:52
	bc1qflzmw7ekt49atj7dy9mqfpe48um0uqme7h...	0.06354886 BTC	➡	3H7SmghHEYg2RZKo53y9s1E3KS2C5ur4eq	0.01000000 BTC
				bc1qh4s9n3th5rtk7m784x8lg72ygq2kkr7lj0lnqa	0.05345574 BTC
Fee	0.00009312 BTC (41.758 sat/B - 16.481 sat/WU - 223 bytes)				0.06345574 BTC

Rule 3, 1:N transaction

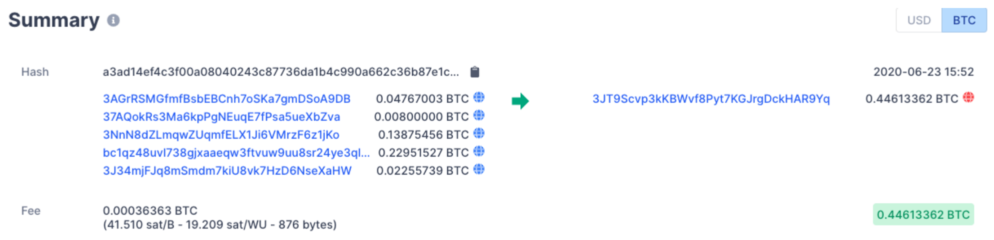
fe988d694f35191fbe6815821052b5208026e8fec404ae667c9519a77d0ca807



```
Summary                                                                                         USD  BTC

Hash   fe988d694f35191fbe6815821052b5208026e8fec404ae667c95...                                  2020-06-23 15:52
       17xkEaLSQND3FpSi9QnXaxHNxAxUUpuvrK    29.98714904 BTC     1138bJ9hGN6hsL4PTTQbpmpdxEDjsQyAJd   0.00100000 BTC
                                                                 1898fABbvWmDxDQpkADZ7BwfJaaBkSb3KQ   0.00855073 BTC
                                                                 3F9y2nJch1EEn5JdA9SHRNZ5p4Fk7xGoRp   0.39950000 BTC
                                                                 3NxiNBWpHFRvCHudm92CnhTHaJ7KbSMRur   0.14738500 BTC
                                                                 37mr6xJib8mbqdYrc6Kz2H1EzVbZn2fQsA   0.35437200 BTC
                                                                 38mbCvNrAApwxjk2GGWevRGgy8kgvd7pKa   0.01463300 BTC
                                                                 329ewRBtKrSJNararKWZbPXmciR65uVDVe   0.14800000 BTC
                                                                 1KvBiSjbLCDQNCoQTW5uLAEne64NjK9BkC   0.04964146 BTC
                                                                 1EsfpGFDcA2S1rwSbyQXYx4Yk5Jwst3Vhv   1.01000000 BTC
                                                                 1Es1n9qmheSVSc8hAzsAFi4Y4ta3wwAuKF   0.66972500 BTC
                                                                 348J8JkXb2PDxpWe1sSL2F4Aek2eoTchpa   0.00170200 BTC
                                                                 1KoHKcgWyxfJkPxEWTZQufwfAg49rpjZzP   0.13243000 BTC
                                                                 1LvUCb5mJovtJLXqjeXCmGE3ajv8ery7JG  27.04982405 BTC

Fee    0.00038580 BTC                                                                           29.98676324 BTC
       (65.612 sat/B - 16.403 sat/WU - 588 bytes)
```

Rule 4, N:1 transaction

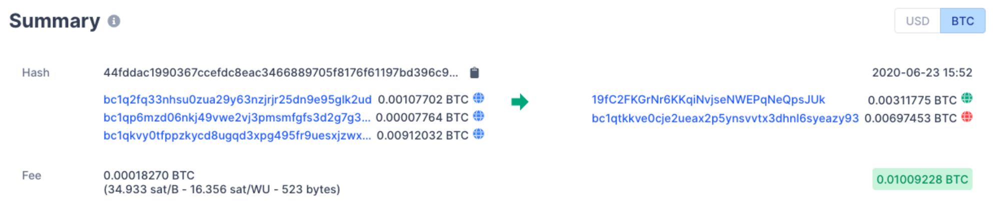
a3ad14ef4c3f00a08040243c87736da1b4c990a662c36b87e1ca6a5a08ce757b



```
Summary                                                                                         USD  BTC

Hash   a3ad14ef4c3f00a08040243c87736da1b4c990a662c36b87e1c...                                   2020-06-23 15:52
       3AGrRSMGfmfBsbEBCnh7oSKa7gmDSoA9DB      0.04767003 BTC     3JT9Scvp3kKBWvf8Pyt7KGJrgDckHAR9Yq  0.44613362 BTC
       37AQokRs3Ma6kpPgNEuqE7fPsa5ueXbZva      0.00800000 BTC
       3NnN8dZLmqwZUqmfELX1Ji6VMrzF6z1jKo      0.13875456 BTC
       bc1qz48uvl738gjxaaeqw3ftvuw9uu8sr24ye3ql... 0.22951527 BTC
       3J34mjFJq8mSmdm7kiU8vk7HzD6NseXaHW      0.02255739 BTC

Fee    0.00036363 BTC                                                                           0.44613362 BTC
       (41.510 sat/B - 19.209 sat/WU - 876 bytes)
```

Rule 5, N:2 transaction

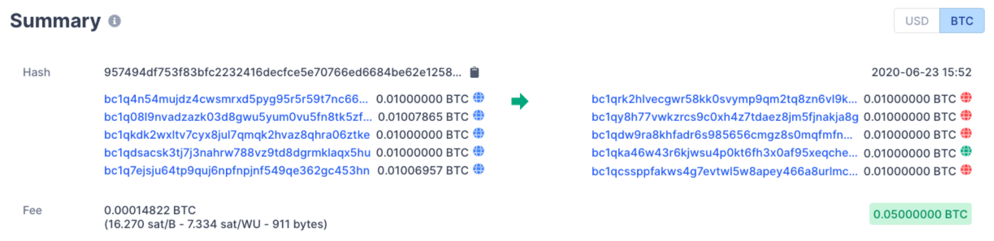
44fddac1990367ccefdc8eac3466889705f8176f61197bd396c95d36b4f950af



```
Summary                                                                                         USD  BTC

Hash   44fddac1990367ccefdc8eac3466889705f8176f61197bd396c9...                                  2020-06-23 15:52
       bc1q2fq33nhsu0zua29y63nzjrjr25dn9e95glk2ud 0.00107702 BTC     19fC2FKGrNr6KKqiNvjseNWEPqNeQpsJUk        0.00311775 BTC
       bc1qp6mzd06nkj49vwe2vj3pmsmfgfs3d2g7g3... 0.00007764 BTC     bc1qtkkve0cje2ueax2p5ynsvvtx3dhnl6syeazy93 0.00697453 BTC
       bc1qkvy0tfppzkycd8ugqd3xpg495fr9uesxjzwx... 0.00912032 BTC

Fee    0.00018270 BTC                                                                           0.01009228 BTC
       (34.933 sat/B - 16.356 sat/WU - 523 bytes)
```

Rule 6, N:N transaction

957494df753f83bfc2232416decfce5e70766ed6684be62e1258da9b765379cb



```
Summary                                                                                         USD  BTC

Hash   957494df753f83bfc2232416decfce5e70766ed6684be62e1258...                                  2020-06-23 15:52
       bc1q4n54mujdz4cwsmrxd5pyg95r5r59t7nc66... 0.01000000 BTC     bc1qrk2hlvecgwr58kk0svymp9qm2tq8zn6vl9k... 0.01000000 BTC
       bc1q08l9nvadzazk03d8gwu5yum0vu5fn8tk5zf... 0.01007865 BTC     bc1qy8h77vwkzrcs9c0xh4z7tdaez8jm5fjnakja8g 0.01000000 BTC
       bc1qkdk2wxltv7cyx8jul7qmqk2hvaz8qhra06ztke 0.01000000 BTC     bc1qdw9ra8khfadr6s985656cmgz8s0mqfmfn... 0.01000000 BTC
       bc1qdsacsk3tj7j3nahrw788vz9td8dgrmklaqx5hu 0.01000000 BTC     bc1qka46w43r6kjwsu4p0kt6fh3x0af95xeqche... 0.01000000 BTC
       bc1q7ejsju64tp9quj6npfnpjnf549qe362gc453hn 0.01006957 BTC     bc1qcssppfakws4g7evtwl5w8apey466a8urlmc... 0.01000000 BTC

Fee    0.00014822 BTC                                                                           0.05000000 BTC
       (16.270 sat/B - 7.334 sat/WU - 911 bytes)
```

Mining Pools

The three figures below show the payout pattern of the three large mining pools researched by Romiti et al. (2019). The gray circles are the reward addresses, the red circles are the addresses performing payout transactions. Blue circles represent the pool members and green circles are the change addresses. The orange rounded squares are coinbases of blocks mined by the pool. Furthermore, the size of the circles (nodes) display the differences of received BTC per transaction per address.

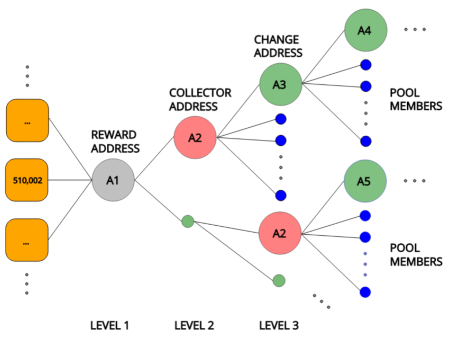


Figure 30: AntPool payout schema, graphic from Romiti et al. (2019)

AntPool (Figure 30) has a payout chain that originates from the same address (A1). After it distributes with different change addresses as inputs for the payout transactions.

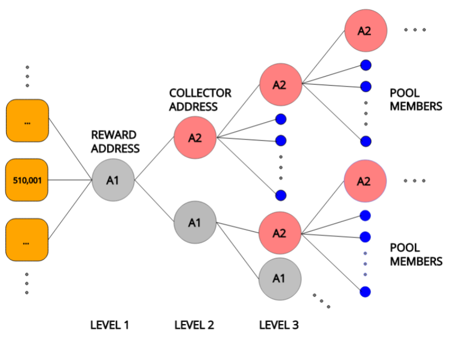


Figure 31: BTC.com payout schema, graphic from Romiti et al. (2019)

In BTC.com mining pool (Figure 31) payout transactions is done with one single collector address.

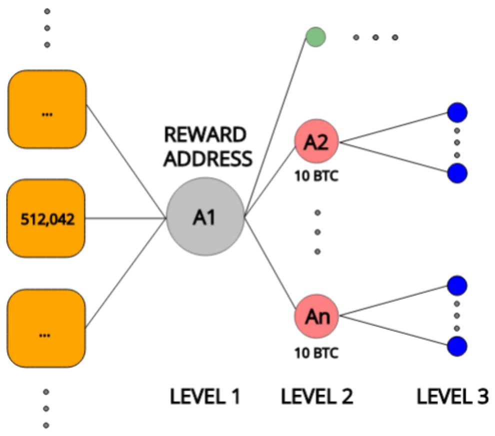


Figure 32: ViaBTC payout schema, graphic from Romiti et al. (2019)

ViaBTC mining pool (Figure 32) is structured in a way that they payout the reward with multiple addresses which are always changing. This changing addresses receive 10 BTC each from this addresses.

The paper (Romiti et al. (2019)) indicates as well that mining pools payout distribution change over time and each pool has a different strategy.

DBSCAN parameters and scores

Modification of Min_points:						
Eps	Min_points	Clusters	Noise points	Silhouette coefficient	Calinski-Harabasz	Davies-Bouldin
0.5	10	4	50	0.30	3’589.45	1.47
0.5	20	5	52	0.29	3’003.87	1.33
0.5	30	5	53	0.29	3’004.08	1.33
0.5	40	5	54	0.29	3’003.91	1.33
0.5	50	5	67	0.28	3’006.78	1.33
0.5	60	5	67	0.29	3’009.76	1.34
0.5	70	6	69	0.29	2’675.16	1.26
0.5	80	6	85	0.29	2’678.57	1.25
0.5	90	6	87	0.29	2’678.28	1.26
0.5	100	6	114	0.29	2’677.02	1.29
Modification of eps:						
Eps	Min_points	Clusters	Noise points	Silhouette coefficient	Calinski-Harabasz	Davies-Bouldin
0.1	50	35	2’737	0.73	3’071.37	1.04
0.2	50	27	1’514	0.49	2’133.78	1.04
0.3	50	14	890	0.21	1’424.11	1.16
0.4	50	7	217	0.28	2’393.08	1.31
0.5	50	5	67	0.28	3’006.78	1.33
0.6	50	3	36	0.40	4’617.58	1.65
0.7	50	3	20	0.40	4’618.73	1.49
0.8	50	3	6	0.39	4’620.26	1.15
0.9	50	3	1	0.40	4’613.47	1.01
1	50	1	0	NA	NA	NA

Table 20: DBSCAN on sample data with varying ϵ and min_points parameter