Identifying blockchain transactions using unsupervised machine learning

Oliver Brandt, Martin Johnsen & Andreas Motzfeldt Jensen

Kongens Lyngby
June 2018

Supervisor: Morten Mørup

Technical University of Denmark
Department of Applied Mathematics and Computer Science
Richard Petersens Plads, building 324,
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

Oliver Brandt - s154131
Martin Johnsen - s144731
Andreas Motzfeldt Jensen - s154097

Abstract

The anonymity properties of the bitcoin blockchain have recently been a popular topic of discussion, especially in relation to money laundering. Given these properties, a question can be posed about whether it is possible to extract meaningful insights from raw transaction data. Using Bayesian machine learning techniques, a framework is proposed in order to analyse the flow of Bitcoins, cluster the users of Bitcoin, and detect abnormal behaviour of these users.
The flow of Bitcoins is analysed by predicting future transactions based solely on the current flow. It is regarded as a network problem, and modelled with a Latent Space Model and a Stochastic Block Model. Both models proves to capture the network structures well, though the Stochastic Block Model turned out to be the most correct model, since it captured group structures assumed to exist between users. The Stochastic Block Model was, with high precision, able to predict missing or future transactions between the most active users.
A Gaussian Mixture Model was utilised on features extracted from Bitcoin transactions. It was applied to cluster users on the bitcoin blockchain, and detect users with abnormal behaviour. With held out data, abnormal users are successfully identified and described.
Given this framework, it was possible to extract meaningful information from a subset of the bitcoin blockchain. Using the Edward library, models was implemented in the scalable architecture of TensorFlow, though a sufficient level of computational power was required to scale-up the entire bitcoin blockchain.

Contents

Chapter 1

Introduction

Bitcoin is a cryptocurrency based on blockchain technology, which was first introduced by Satoshi Nakamoto (most certainly a pseudonym)[1] in 2008. The Bitcoin technology has some features that are fundamentally different from the conventional ways to exchange cash and currencies. It differs from fiat money in two aspects: First, it is decentralised, and therefore no authority can regulate and generate new Bitcoins. Instead, Bitcoins are issued at a pre-determined rate, and new Bitcoins will cease to be issued when the total number has reached 21 million. Due to the peer-to-peer network of Bitcoin, the usual key players in the financial market (the authorities and the banking industry) are eliminated, which lowers transaction costs, and makes it easier for transactions to cross borders and hide the involved parties[2]. Second, every transaction is globally visible through the Bitcoin blockchain, which logs every single transaction that remains irreversible. However, since a user is only identified by a pseudonym (address) and can have an arbitrary number of pseudonyms, the bitcoin blockchain can provide a channel for money laundering and other illegal services[3].

The first Bitcoin (BTC) transaction was made in 2009, and by early June 2018, there are over 16,9 million Bitcoins in circulation. The value of Bitcoin skyrocketed with a record high of $19.850 per BTC in December 2017, but it has tumbled down to $7.540 per BTC[4] by early June 2018. The volatile behaviour, and the nature of Bitcoin is getting a lot of attention both financially and politically.

For instance, it has been featured on the front-page of both The Economist and Forbes Magazine[5] [6]. Some governments and other political institutions are concerned about the untraceability of Bitcoin transactions and its risk to society through money laundering, illegal services, and tax evasion[7]. The absence of a central authority to monitor and regulate Bitcoin is what often concerns or fascinates people[8].

Most studies which focus on the bitcoin blockchain (see appendix A for preliminary literature retrieval), deal with a variety of clustering and anomaly detection methods, but due to the recent developments of Bitcoin, only a limited number of transactions have been classified as fraud. For that reason, the use of supervised learning methods is inconvenient.
Data from the bitcoin blockchain can be regarded from two points of view: (1) a *user record* dataset and (2) a *transaction network* of nodes and edges. This project will apply machine learning techniques regarding both views, in order to examine transaction profiles by: (1) the behaviour of the users and (2) the link pattern of the users.

The only study that regards the bitcoin blockchain as a *user record* dataset is the work of Poikonen[9]. It builds on the work of Reid and Harrigan[8], by compiling a record of metrics for every unique user on the bitcoin blockchain, based on two general heuristics (described in section 2.1). The two heuristics are applied to connect Bitcoin transactions that are transferred by the same unique user. This process of connecting several transactions to a unique user applies a feature extraction used to cluster Bitcoin users. By compiling and formatting a transaction matrix where each row contained a source ID, destination ID, timestamp, and transaction value, they were able to extract several features, using Principal Component Analysis, the five most important features for explaining the dataset were determined. Furthermore, K-Nearest Neighbour was applied, and the effectiveness was validated. However, no significant results have been reached at this stage.

The majority of the previous related works have regarded the bitcoin blockchain as a *user network* dataset. The work of Pham and Lee[10] investigated the anonymity perspective by applying K-means clustering, Mahalanobis distance, and Unsupervised Support Vector Machine (SVM). The goal was to detect anomalies using ground truth data defined as cases of theft, and other cases of illegal activity, linked to either users and/or transactions. The study used the three unsupervised machine learning methods on two graphs generated by transactions from the bitcoin blockchain, where several features were specified for each node. The unsupervised learning algorithms provided a success rate of 3 out of 30, and detected one case of loss and two cases of theft. The work of Monamo et al.[11] had a similar approach, and obtained approximately the same results by using a trimmed K-means clustering. The goal of the study

was to detect fraudulent activity on the bitcoin blockchain based on pattern recognition on Bitcoin transactions. The detection was slightly better than the results from[10], with a success rate of 5 out of 30. The work of Hirshman et al.[12] tried to cluster a dataset of transactions using K-means, but also to detect attempts at money laundering. Furthermore, the study tried to trace the outputs of a mixing service[1] back to its input. The study were able to identify abnormal behaviour using RolX, but failed to identify cases of mixing services.

As described above, K-means clustering is often used in the studies, but the methodology lacks the ability to estimate densities, which is why this paper proposes a Gaussian Mixture Model in order to cluster users and detect abnormal users. In general, this project stands out from the above mentioned works, since it takes a Bayesian approach to model the data. Furthermore, the ecosystem of the bitcoin blockchain has, at the time of writing, not been mapped. Therefore, this project builds on the work of Liben-Nowell and Kleinburg[13], and the work of Hoff et al.[14], by proposing random graph models in terms of interpreting a structure of the ecosystem, identifying a flow of Bitcoins between users, and predicting future flows based on the present structure. We analyse the structure by applying a Stochastic Block Model, and compare it to the alternative method, the Latent Space Model (or Latent Position Model), presented in[14]. The work of[13] considers the link prediction problem, and seeks to uncover how accurately the evolution of a social network can be modelled with features intrinsic to the network itself, i.e. ability to predict the interactions (edges) that will be added to the network. We will build on this method by predicting links between users (thereby the future structure of the bitcoin blockchain) with the Stochastic Block Model and Latent Space Model. By predicting solely on the link pattern for each user rather than specific metrics for each user, the approach differs from the related works[10][11] presented above.

For this project, we are interested in using unsupervised machine learning methods to explore the bitcoin blockchain, given that it was never intended to guarantee anonymity[1]. The project has three aims:

1. To predict the future flow of Bitcoins.
2. To extract meaningful clusters of Bitcoin users.
3. To detect abnormal behaviour.

Our goal is to shed light on the complexity of the bitcoin blockchain, and whether it is possible to gain new insights. For instance what can be interpreted from the hidden structures of the bitcoin blockchain, and what can be said about the flow of Bitcoins. Thus, this project will not predict the market value of

[1] A service that mix one's bitcoins with others, in order to make the funds unlinkable

Bitcoins, and the value is therefore not taken into account. Neither will it attempt to de-anonymize the users of the bitcoin blockchain. It will identify Bitcoin transactions using unsupervised learning, which can help increase the transparency of the users performing transactions, and shed light on important questions such as their degree of anomaly, their link pattern, and their position in the ecosystem. Maybe it will even be possible to eliminate some of the scepticism associated with Bitcoins. Furthermore, the bitcoin blockchain can be seen as one of the largest publicly available transaction datasets in the world. Therefore, our contribution builds on a large-scale architecture of unsupervised machine learning models.

Our framework including the models and prepossessing is all available on our GitHub at: https://github.com/brandtoliver/Blockchain-Transaction-Classification

The paper is structured as follows: In chapter 2, our main focus is on understanding the anonymity properties of the bitcoin blockchain, including a description of the collected data and the associated preparation. The methods applied in order to interpret hidden structures of the bitcoin blockchain are introduced in chapter 3. In chapter 4, the three different unsupervised machine learning models are implemented and used for clustering, anomaly detection, and link prediction, including an analysis and discussion of the results. At last, we present the conclusion of the project, and reflect on future work in chapter 5.

Chapter 2

Data preperation

In this chapter, we will give a thorough introduction to the system of Bitcoin, and the data we extract from the bitcoin blockchain. This is done by, in the first section, explaining the general principles, in order to describe them in detail. The second section touch on how the data was collected, and transformed into applicable data structures. Thus, it was possible to extract a *user record* and *user network* dataset, which is elaborated on, in section 3 and 4.

2.1 The bitcoin blockchain

In order to proceed with our analysis, it is essential to provide an introduction to the bitcoin blockchain. From now on, the term 'bitcoin network' will be applied, and it refers to the Bitcoin peer-to-peer network. The term 'bitcoin protocol' will also be applied, and refers to the rules and nature of the bitcoin blockchain. Finally, the term 'Bitcoin (BTC)' refers to the currency itself.

The above terms together define the three essential parts that make up the bitcoin blockchain (see figure 2.1): one, the bitcoin protocol that defines the rules on how to transact; two, the bitcoin network that facilitates the peer-to-peer interaction; and three, the Bitcoin cryptocurrency, which enable the transfer of value in the system.

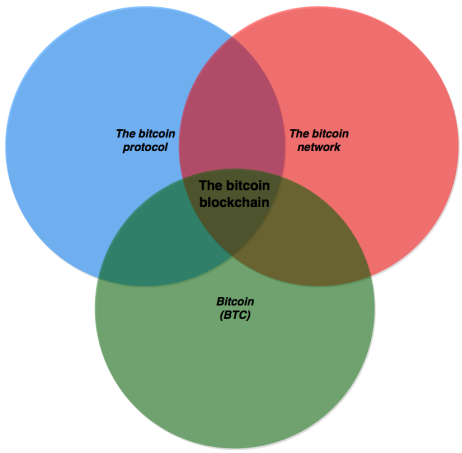


Figure 2.1: Illustration of the system

The bitcoin blockchain is also referred to as a global transaction ledger[3], meaning that anyone can access transaction information, which is unlike a banking ledger. This transparency enables the bitcoin network to verify new transactions[2].

The transfer of money is built upon trust[15]. This also applies to cryptocurrencies. Within the bitcoin blockchain, trust is provided through a combination of cryptographic digital signatures, and a decentralised network.

The digital signature is made by combining two linked keys: the private key and the public key. The private key is used to create the signature, and the public key to verify the signature [8]. In this way, it is possible to prove the rightful owner without revealing the private key. In- and outgoing transactions are connected by a public key, which refer to a user, who can have an arbitrary amount of public keys[1]. In other words, the public key is the address you send Bitcoins to. In order to spend Bitcoins, you must prove that you are the rightful owner of the public key through a digital signature, which is the private key. Because the digital signature is linked to the transaction, it will be unique for every transaction, and therefore it cannot be either reused or manipulated, since it will invalidate the transaction[2]. To validate, transactions are collected into blocks, and accepted to the chain of blocks (hence the name blockchain) when consensus is accomplished. Like in a democracy, the one with the most votes wins. On the bitcoin blockchain, consensus is found at the block that was validated by most computational power. The trust is maintained through the fact that it is infeasible to own the majority of the computational power in the network[1].

Example: If Bob has a certain amount of Bitcoins stored at one of his public keys and wants to send Bitcoins to Alice, then he must first refer to the transactions that provided him with the Bitcoins. Therefore, Bob creates a message to a receiver, identified by Alice's public key, and a transaction value. Bob has to validate the transaction through his private key, which is the digital signature. This digital signature is what indicates where the Bitcoins originated from, and what validates that Bob is the rightful owner of the specified value. Combining the message and the digital signature makes the transaction. Then, Bob's transaction is broadcasted to the bitcoin blockchain. However, it is important to mention that a transaction can have an arbitrary number of inputs and outputs. In this example, the transaction is simplified to contain only one input and one output equal on amount. As mentioned, in order to send Bitcoins, Bob must refer to previous transactions. This means, that balances and ownership of Bitcoins are defined by previous transactions. If Bob wants to transfer 10 BTC, then he must refer to other transactions where he received 10 BTC or more. In this manner, ownership of Bitcoins flows in a chain, where the validity of each transaction relies on previous transactions.

Broadcasted transactions are, as previously mentioned collected into blocks, where peers on the bitcoin blockchain are competing against each other to close the block. This process is called mining, and is a computational resource-intensive cryptographic puzzle, also referred to as Proof of Work. The first miner to solve the puzzle and establish consensus earns newly issued Bitcoins and a transaction fee in return[3]. The main purpose of mining is to log transactions to a certain block with a timestamp, verifying each transaction, and thereby each block.

In this study, the following three properties of the bitcoin blockchain are of significant importance.

- **Property 1:** The entire transaction history is publicly available and each transaction must refer to a previous transaction. Therefore, it is possible to link transaction inputs and outputs, thus tracing the whereabouts of a certain Bitcoin.

- **Property 2:** A transaction can have multiple inputs. If a user does not hold enough funds on one address to pay a given amount, the sender must include an additional input from another address to pay the sufficient amount.

- **Property 3:** The sender *must* spend all Bitcoins stored at the public key when involved in a transaction. Therefore, a transaction often has two (or more) outputs: a payment output and a change output.

These three properties make it possible to create two heuristics, which can be applied to the transaction data in order to map addresses to individual users. *The first heuristic* is based on the second property, and states that all input addresses in a transaction must belong to the same user. This heuristic is quite intuitive, since the reason for several input addresses stems from an insufficient amount of Bitcoins on one address. As mentioned, the same user must control all input addresses by knowing the private keys, to perform a transaction containing multiple inputs. *The second heuristic* exploits the third property, and assigns the change output and the associated address to the same user who initiated the transaction. This second heuristic holds a bit more uncertainty, since it requires correct identification of the change address, and it seems reasonable to assign ownership of the change address to the same user who controls the input addresses. However, this project never managed to implement *the second heuristic*, thus the data will only account for *the first heuristic*.

2.2 Data collection, parsing and preprocessing

In this section, we rely on transaction data obtained from the bitcoin blockchain. The data is publicly available through the Bitcoin Core[16]. The Bitcoin Core is a software that captures all valid blocks with valid transactions throughout the history of the bitcoin blockchain. As of right now, June 2018, the transaction history is around 160GB[4]. The obtained dataset used in this project, contains transactions from the creation of bitcoin blockchain until December 28th, 2013 (later filtered to a smaller dataset).

To make data usable for the scope of this project, a data structure matching the modelling purposes is obtained, and the first heuristic is applied. The first heuristic is applied, while the scope is to examine independent user transactions on the bitcoin blockchain. Analysing independent addresses would not serve this purpose, as different users often hold multiple addresses.

Few frameworks for applying heuristics to data from Bitcoin Core is publicly available, while modelling blockchain data is still relatively unexplored. However, two open source projects have been of benefit for this project. Michael Egger and the "rusty-blockparser" GitHub repository provides a framework for parsing BLK.dat files into a flat .csv format. Daniel Kondor and the ELTE Bitcoin project has published a method for applying the first and most important of the two heuristics.

The data obtained from Bitcoin Core is structured in .dat files. The structure of .dat files requires special parsing before programming languages like Python,

R or similar are able to process it.

Figure 2.2 is a visualisation of how to get data from Bitcoin Core, and transform this into a flat .csv format, by applying the first heuristic.

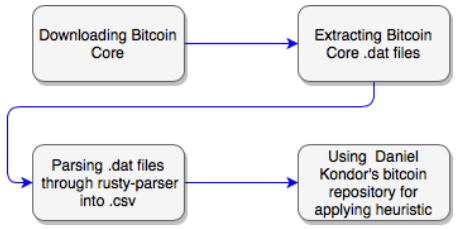


Figure 2.2: Parsing data from the Bitcoin Core

When the transformation of .dat files into a usable .csv format is succeeded, preprocessing takes place. This is done to obtain two datasets to be applied for two independent modelling purposes. One dataset should contain records of individual users and information about their transaction behaviour. This dataset represents the bitcoin blockchain as a user record.

Another dataset should contain an adjacency matrix, thus looking at the blockchain from a *user network* perspective (See section 2.4).

Note that due to scalability limitations, both the user record dataset and the adjacency matrix is filtered to only contain the 9968 most active users, based on the number of performed transactions pr. user. These 9968 users are used throughout the rest of the project.

2.3 User record

In this section, we give a description of the approach used to obtain the *user record* dataset. First, we introduce the structure of the dataset downloaded from the bitcoin blockchain and its relation to the *user record* dataset. Second, we motivate the feature extraction and the domain relevance for these features. Lastly, the raw user record data, used throughout the rest of this project, is examined briefly. The user record dataset contains N users and D features in a $N \times D$ matrix.

Ultimately, the transformation and parsing (see appendix A) leaves five tables, which all hold various attributes to be joined. To provide a simple overview,

figure 2.3 shows an entity relationship diagram of how the *user record* dataset has been created. The record dataset contains information of every distinct user on the bitcoin blockchain, given the first heuristic (where all input addresses in a transaction are regarded as belonging to the same user).

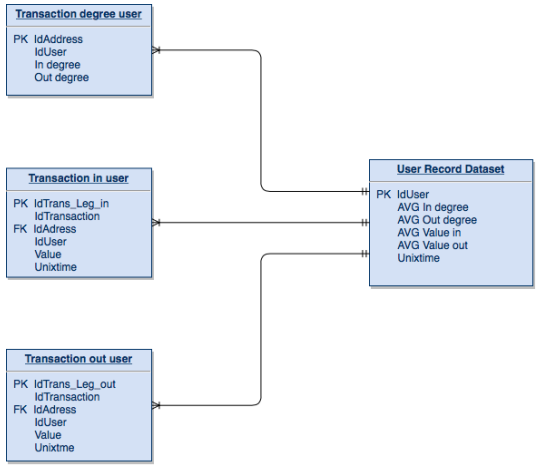


Figure 2.3: ER-Diagram of the user record dataset

By combining the transformed tables, it is possible to extract the information into Transaction degree, Transaction in, and Transaction out. All rows of these tables hold an *IdUser*, which is the distinct user. The tables are then aggregated into the User Record Dataset, holding inherited information about every distinct user. Derived information is added through feature extraction.

2.3.1 Feature extraction

Based on the obtained variables from the bitcoin blockchain and an extensive study on previous practices (see appendix A), meaningful features are extracted to feed a learning algorithm with domain-specific knowledge. Relevant features result in better models, and improve answers to the proposed research questions. Our aim is to discover features that can segregate abnormal and normal users. In this section the features are defined and described, and the motivation behind them are given. In total, 8 features are derived from the bitcoin blockchain.

1. *Number of pseudonyms*: Represents the number of public keys belonging to each user.

With this feature, the aim is to divide users by their level of unlinkability.

2. *Number of transactions*: Represents the number of transactions where this user appears.

   With this feature, the aim is to divide users by their activity level.

3. *Average in degree*: Represents the average number of addresses which appear as input in transactions, where this address appears as output.

   With this feature, the aim is to divide users by how many different addresses the user interacts with on average, when receiving Bitcoins.

4. *Average out degree*: Represents the average number of addresses, which appear as output in transactions where this address appears as input.

   With this feature, the aim is to divide users by how many different addresses the user interact with in average, when sending Bitcoins.

5. *Average amount received*: Represents the average amount of Bitcoins the address has received in a transaction.

   With this feature, the aim is to divide users by the average amount of Bitcoins they receive.

6. *Average amount send*: Represents the average amount of Bitcoins the address has send in a transaction.

   With this feature, the aim is to divide users by the average amount of Bitcoins sent.

7. *Active duration per user*: Represents the time interval between the first transactions and most recent transaction.

   With this feature, the aim is to divide users by the time they have been active on the bitcoin blockchain.

8. *Average active duration per pseudonym*: Represent the average time between first and most recent transaction for each public key per user.

   With this feature, the aim is to divide users by how long they keep their public key active on average.

2.3.2 Summary statistics and visualisation

A summary of the user record dataset is presented in table 2.1, to discover insights before modelling the dataset.

It is important to observe how the mean and median of the first four features

Table 2.1: Summary statistics for user record dataset with amounts in Satoshi[1] and duration in seconds

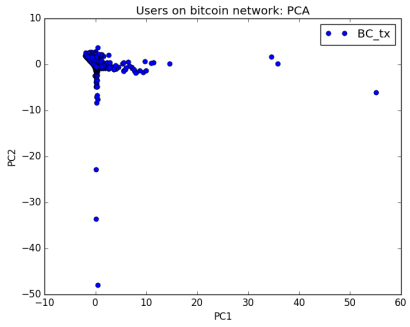
	Mean	Median	Min	Max
1. #Pseudonyms	639.58	36.00	1.00	544754.00
2. #Transactions	3974.31	777.00	404.00	12315496.00
3. Avg. in degree	66.86	4.40	1.00	119079.00
4. Avg. out degree	88.72	16.62	0.00	12296.00
5. Avg. amount received	4.49e+08	3.13e+07	2.54e+03	6.41e+11
6. Avg. amount sent	4.53e+08	3.16e+07	2.71e+03	6.43e+11
7. Active duration pr. user	2.58e+07	1.85e+07	1.78e+03	1.54e+08
8. Avg. active duration pr. pseudonym	1.59e+07	1,02e+07	1786.00	9.50e+07

are significantly different. This implies that a minority of users are using a lot of pseudonyms, making a lot of transactions and have a lot of in- and out degrees, while the majority of users are less active and hold fewer pseudonyms. The suggested minority might be interpreted as 'super-users' within the 9968 observed users.

The 9968 observed users have an active duration (feature 7.) mean around 298 days. The minimum observation for this feature is less than a day, and the maximum observation is almost 5 years. 5-year-old users indicate activity since the beginning of the bitcoin blockchain in January 2009.
Features five and six suggest that some users sent and receive significantly more than the mean and median user. This might imply that some users have more Bitcoins, or spend more Bitcoins than the rest of the users.

Figure 2.4a presents a Principal Component Analysis (PCA) of the user record data. Visualisation of users implies three users having significantly high values for principal component one, and three other users having significantly low values for principal component two.
Holding the PCA in perspective to summary statistics, table 2.1, the six outlying users could represent the high and low values from the table. Significantly different users suggest outliers.

Features and their correlation can suggest linear relationships between data points. The correlation matrix, figure 2.4b, indicates a strong linear correlation between the number of pseudonyms, and the average active duration of a pseudonym. This is intuitive, in relation to bitcoin blockchain, since users need a new pseudonym when an old one is not used anymore. Thus, if the average period of holding a pseudonym is low, a user is creating new pseudonyms more often. The rest of the feature does not show a high level of correlation.



(a) Principal component analysis for user record dataset

	nr_pseudonyms	avg_pos_period	mean_value_in	nr_transactions	mean_value_out	avg_out_degree	avg_in_degree	active_duration
nr_pseudonyms	1,00	0,70	0,00	0,15	0,00	0,00	0,00	0,00
avg_pos_period	0,70	1,00	0,01	0,37	0,08	0,06	0,06	0,00
mean_value_in	0,00	0,01	1,00	0,01	-0,01	0,01	0,07	0,20
nr_transactions	0,15	0,37	0,01	1,00	0,00	0,03	0,06	0,00
mean_value_out	0,00	0,08	-0,01	0,00	1,00	0,00	0,00	-0,02
avg_out_degree	0,00	0,06	0,01	0,03	0,00	1,00	0,03	-0,01
avg_in_degree	0,00	0,06	0,07	0,06	0,00	0,03	1,00	0,02
active_duration	0,00	0,00	0,20	0,00	-0,02	-0,01	0,02	1,00

(b) Correlation matrix for user record features

Figure 2.4

2.4 User network and Link prediction

A *user network* dataset is extracted to describe the user from a network approach. The extraction of the *user network* dataset is explained by first commenting on the relevance of this approach. Second, we visualise the transactions used throughout the project, and argue for a reasonable filtering of transactions. Lastly, an ER-diagram of the transaction network is shown and the link prediction approach is introduced.

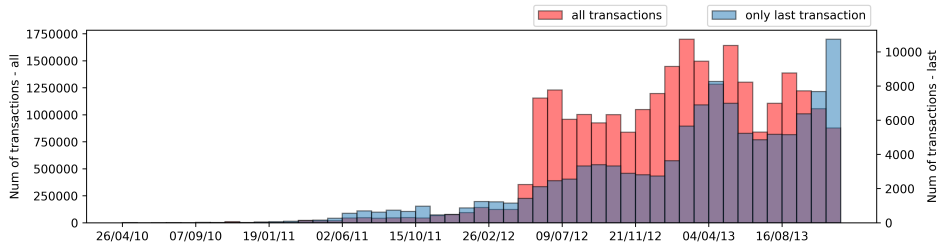


Figure 2.5: Dual axis histogram of transactions between the 9968 top users, where red bars represent the total number of transactions, and blue bars represent only the last transaction between pair of users.

In figure 2.5 it is seen that the transaction volume is increasing exponentially. This growth in number of transactions in the subset (red), and the smaller subset of only the last transaction (blue), closely resembles the growth of the total number of transactions[2](not shown in the figure).
Lastly, the edge-list is transformed to a 9968×9968 adjacency matrix, with a sparsity of 0.6%.

Figure A.7 in appendix B gives a more detailed overview of how the adjacency matrix is obtained.

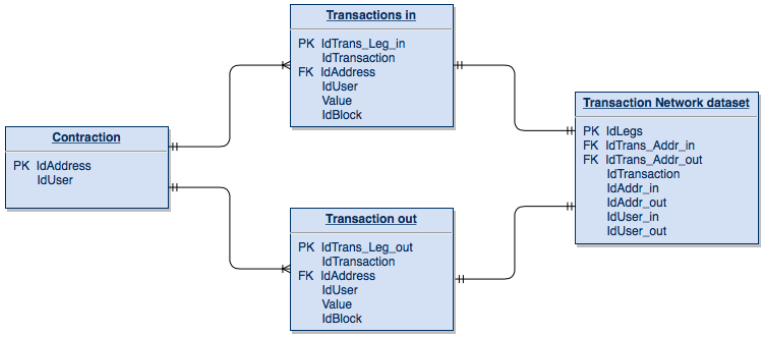


Figure 2.6: ER-Diagram of the network dataset

Predicting transactions between users is highly related to the *link prediction problem* described in *The link Prediction Problem for Social Networks* by *Kleinberg et al.*[13]. Given a network of users transacting with each other at time t, is it possible to predict which transactions will be performed in the time-interval t to t'? To answer this, we define the time-interval t to t', to be the last 1% of transactions in the network. In practice, we roll back the network by removing (zeroing) the last 1% of transactions from the adjacency matrix. This corresponds to rolling back the network of transaction 22.2 hours. From figure 2.5, and the fact that we only use the last transaction between two users, it is clear why 1% of the transactions only represent 22.2 hours. Later, we use the last 1% of the transactions in the network as ground-truth, when evaluating the predictive performance of our models.

[2]https://blockchain.info/charts/n-transactions-total?timespan=all

Chapter 3

Methods

In this chapter, we will introduce the methodologies central to this project. First, we discuss the field of machine learning and probability theory. The second section introduces probabilistic models, a Bayesian approach to clarify the relations between users in the bitcoin network. In the third section, we define the applied inference algorithms before defining the applied models in the fourth section: a Gaussian Mixture Model (GMM), a Latent Space Model (LSM) and a Stochastic Block Model (SBM). At last, we introduce the scalable machine learning framework TensorFlow and the probabilistic library Edward.

3.1 Notation

Table 3.1 defines some general notations used to describe the introduced methods. Any notation not defined, is explained explicitly when relevant.

X	True dataset
$\tilde{X}$	Generated dataset
θ	Model parameters
N	Number of observations
n	Observation
K	Number of mixtures, latent dimensions & communities
q	Approximated posterior
p	Probability
A	Adjacency matrix
i, j	Particular matrix elements
z	Latent variables

Table 3.1: Notation table

3.2 Machine learning and Bayes theorem

Machine learning is typically simplified into two classes: *supervised learning* and *unsupervised learning*. In *supervised learning*, the intention is to learn outcome y, based on input x, given a dataset X, consisting of labeled (x,y) pairs: $X = (x_i, y_i)_{i=1}^N$.
In *unsupervised learning*, you are not given any desired output, therefore the intention is to find patterns and relations only given an input: $X = (x_i)_{i=1}^N$, and build a representation with the purpose of decision making, predicting etc.[17].

In this project, probabilistic theory is considered to be the most efficient way of learning from data, since we assume the intention of bitcoin blockchain users are uncertain. The bitcoin blockchain is full of uncertainty, due to the anonymity properties, as mentioned in chapter 1. Probability theory gives us a framework to model these uncertainties, by training a model to find both observable and latent patterns in the data. We define the probability of an event as the likelihood of that event to occur[18]. For example, the probability of event A to occur, or the event of both event A and B to occur, can be written as

$$\mathrm{p}(A) \text{ and } \mathrm{p}(AB)$$

And if we instead wanted to find the probability of event A occurring, or both A and B occurring, given a third event C, we write this as the conditional probability

$$\mathrm{p}(A \mid C)\ \mathrm{p}(C) \text{ and } \mathrm{p}(AB \mid C)\ \mathrm{p}(C)$$

A probability can take all values between 0 and 1, where $\mathrm{p}(A) = 0$ denotes that event A will most certainly not occur, and $\mathrm{p}(A) = 1$ denotes that event A will most certainly occur. In probability theory, there are two crucial rules:

$$\text{The sum rule: } \mathrm{p}(A \mid C) + \mathrm{p}(\overline{A} \mid C) = 1$$

$$\text{The product rule: } \mathrm{p}(AB \mid C) = \mathrm{p}(B \mid AC)\mathrm{p}(A \mid C)$$

Bayes' theorem builds on the axioms of the conditional probability, and follows from the two rules stated above, where it shows the relation between its inverse. It states that for two events A and B, that if we know the conditional probability of event B given event A and the probability of A, we can compute the conditional probability of A given B[19].

$$\text{Bayes' theorem: } \mathrm{p}(A \mid B) = \frac{\mathrm{p}(B \mid A)\ \mathrm{p}(A)}{\int\ \mathrm{p}(B \mid A)\ \mathrm{p}(A)} \tag{3.1}$$

Bayes' theorem was introduced by the statistician Thomas Bayes, who showed how to more accurately re-estimate the same probability based on new knowledge[17]. In other words, the posterior probability $\mathrm{p}(A \mid B)$ can be calculated by multiplying the likelihood $\mathrm{p}(B \mid A)$ by the prior probability $\mathrm{p}(A)$, and dividing their product by the marginal likelihood $\int\ \mathrm{p}(B \mid A)\ \mathrm{p}(A)$. The prior probability is the probability initialised by the present knowledge, and the posterior probability is what we believe is the probability, given present belief and some likelihood estimates.

3.3 Probabilistic models

A model is generally formed by three types of variables. First, the observed data represented by N data points $x = \{x_1, \ldots, x_N\}$. Second, the model holds expected latent variables encoded as hidden quantities. These quantities follow a given prior distribution defined accordingly. Third, a model can include hyperparameters, a non-random fixed number[20].

Probabilistic models are built upon probability theory, i.e. it requires randomness, in order to predict future outcomes. Probability theory describes that for two jointly distributed continuous random variables X and Y, the conditional probability of Y given a certain value $X = x$ is:

$$\mathrm{p}(y \mid x) = \frac{\mathrm{p}(x, y)}{\mathrm{p}(x)} \tag{3.2}$$

where $\mathrm{p}(x, y)$ is the joint probability density for X and Y. From the conditional distributions of X and Y, the joint probability density $\mathrm{p}(x, y)$ is now given by:

$$\mathrm{p}(x, y) = \mathrm{p}(x)\ \mathrm{p}(y|x) \tag{3.3}$$

where $\mathrm{p}(y \mid x)$ is the conditional distribution of Y, given $X = x$, as shown in equation (3.2). Modelling the joint distributions of the conditionals, represents the relationship of the latent and observed random variables.

Probabilistic models, are models build with embedded uncertainties. Model parameters are encoded as prior distributions, which anticipate certain structures on the subject of matter. The goal of probabilistic modelling is to take some data X, and say there is some latent structure in it z, for then to pose a model of how X and z are related $\mathrm{p}(X, z)$. For example, for the data obtained in this project, the transaction record is our data X, where links between users could underlie some latent structure z. Then in a probabilistic model, we can denote probabilities p of the links occurring based on their latent representation z. Exactly this latent representation is shaping model parameters to provide the posterior distribution, which procure a predictive model. Therefore, the posterior distribution, given by the inferred model parameters, holds the anticipated and encoded latent structures.

In probabilistic models, the posterior distribution is found through an inference algorithm, which finds the posterior distribution of the latent variables, given some data. The definition of a probabilistic modelling is ambiguous. In this project, the Variational inference is viewed as what makes up probabilistic modelling. The idea is to write a model $\mathrm{p}(X, z)$, provide it with some data X, and then specify an inference algorithm to find the true underlying structure of the data. These features can be embedded into a generative model described in section 3.5.

3.4 Inference algorithms

The learning process of probabilistic modelling happens through an inference algorithm. An inference algorithm is used to derive parameters given some data, X, possessing hidden structures. The parameters, so-called hidden parameters, seeks to explain the hidden structures that created X. This project will utilise three different inference algorithms: Maximum a Posteriori, Variational Inference and Gibbs Sampling. These all imply a probabilistic approach, one of them using variational inference. This section will briefly touch upon inference conceptually, before explaining the inference algorithms in more detail.

In exact inference, given a directed graphical model of nodes $S = \{S_1..S_n\}$, each node S_i depends on its parent node $S_{\Pi(i)}$, $\Pi(i)$ being the set of parent nodes of any given node S_i. Expressed formally, node S_i is conditional of its parents $\mathrm{p}(S_i \mid S_{\Pi(i)})$ [21]. To obtain a probability distribution of all nodes in the graph, the joint probability distribution is calculated:

$$P(S) = \prod_{i=1}^{N} P(S_i \mid S_{\Pi(i)}) \tag{3.4}$$

3.4.1 Maximum a Posteriori

In the field of probability theory, Maximum a Posteriori (MAP) is the procedure of sampling data, in order to estimate the mode of a posterior distribution, which can be seen as the best estimate. Meaning that MAP can be utilised as a point estimation of some unobserved parameters, based on the prior distribution. Hence MAP involves calculating a maximum likelihood incorporating the prior distribution[17].

Say that we are given data $X = (x_1, \ldots, x_N)$, and we assume a probabilistic model, in particular a joint distribution $\mathrm{p}(X|\theta)$, where θ is a random variable governing the distribution on the data. By applying MAP, our goal is to choose a quantity that maximises the probability of θ, given the data, i.e. MAP maximises the posterior distribution[22].

$$\theta_{MAP} = \underset{\theta}{argmax}\ \mathrm{p}(\theta \mid X) \tag{3.5}$$

MAP follows from Bayes Theorem given by equation (3.1), which allows us to assume θ to be a random variable. Therefore, MAP will estimate the model parameters as the posterior distribution incorporating θ as this random variable[22].

$$\theta_{MAP} = \underset{\theta}{argmax}\ \mathrm{p}(\theta \mid X) \tag{3.6}$$

$$= \underset{\theta}{argmax} = \frac{\mathrm{p}(X \mid \theta)\ \mathrm{p}(\theta)}{\int\ \mathrm{p}(X \mid \theta)\ \mathrm{p}(\theta) d\theta} \tag{3.7}$$

$$= \underset{\theta}{argmax}\ \mathrm{p}(X \mid \theta)\ \mathrm{p}(\theta) \tag{3.8}$$

Note that the marginal likelihood $\int\ \mathrm{p}(X \mid \theta)\ \mathrm{p}(\theta) d\theta$ is not imposed on θ, and therefore irrelevant to the optimisation[23]. Often it is more convenient to use log:

$$\theta_{MAP} = \underset{\theta}{argmax}(\log(\ \mathrm{p}(\theta)) + \sum_{i=1}^{n} \log(\ \mathrm{p}(X_i \mid \theta))) \tag{3.9}$$

Where θ_{MAP} is the estimated parameter, argmax(θ) is value of theta that maximises, $log(\ \mathrm{p}(\theta))$ is the log prior, and $\sum_{i=1}^{n} \log(\ \mathrm{p}(X_i \mid \theta)$ is the sum of log likelihood.
Though MAP is often referred to the field of Bayesian statistics as a result of the assumption that θ is considered a random variable, it can be discussed if the classification of a Variational inference is correct. In contrast to MAP, Variational inference has to marginalise θ, in order estimate the model parameter, and thus MAP does not incorporate any uncertainty in the value θ[17].

3.4.2 Variational inference

Variational inference involves calculations of a conditional distribution $\mathrm{p}(z|x)$, although we do not know this distribution. Having equation $\mathrm{p}(z|x)$ as the desired distribution, it is convenient to introduce a family distribution, $\mathrm{q}(z)$, used to approximate the observed (true) distribution. Ideally, $\mathrm{q}(z)$ will approximate the conditional distribution, however, single $\mathrm{q}(x_i)$'s will indicate poor approximation to observed marginals, $\mathrm{p}(x_i)$[24].

Variational inference, as opposed to Monte Carlo inference, perceives the posterior approximation as a deterministic process, in which sampling is replaced with optimisation. It is relevant to note that variational inference tends to scale better than traditional sampling inference algorithms[25].

Some likelihood functions can measure similarities between the approximated posterior, and the true posterior distribution. This similarity (or dissimilarity) is used to shape model parameters, to gain a more complete approximation. The difference from an approximated posterior to a real posterior is evaluated using Kullback-Leibler (KL) divergence. Calculating KL-divergence, $KL(q||p)$, measures the distance between the approximated distribution $\mathrm{q}(z)$, and the true posterior $\mathrm{p}(z|x)$. The objective of KL-divergence is to minimize the function with respect to model parameters θ,

$$\theta* = \arg\min_{\theta} KL(q(z|\theta)||p(z|x)) \tag{3.10}$$

where KL-divergence is denoted as:

$$KL(q||p) = E_q\left[\log \frac{q(z)}{p(z|x)}\right] \tag{3.11}$$

E_q is the expectation taken with respect to the variational distribution[26][25].

Equation (3.11) is turned into an optimisation function by rewriting:

$$KL(q||p) = -E_q\left[\log \frac{p(z|x)}{q(z)}\right] \tag{3.12}$$

from (3.1)

$$= -E_q\left[\log \frac{\mathrm{p}(x,z)}{\mathrm{p}(x)} \frac{1}{q(z)}\right] \tag{3.13}$$

$$= -E_q\left[\log \frac{\mathrm{p}(x,z)}{q(z)} \frac{1}{\mathrm{p}(x)}\right] \tag{3.14}$$

$$= -E_q \log \left[\frac{\mathrm{p}(x,z)}{q(z)} \right] + \log(\mathrm{p}(x)) \tag{3.15}$$

$$KL(q||p) + E_q \log \left[\frac{\mathrm{p}(x,z)}{q(z)} \right] = \log(\mathrm{p}(x)) \tag{3.16}$$

$$KL(q||p) + \Delta = \log(\mathrm{p}(x)) \tag{3.17}$$

Thus, the Δ provides a lower bound to $\log(\mathrm{p}(x))$. KL-divergence holds the property of always being positive, thus maximising Δ is the same as minimising the KL-divergence. Thereby, minimising the KL-divergence results in reaching a desired best approximated posterior distribution.

3.4.3 Gibbs sampling

In contrast to variational inference methods, sampling methods draw directly from a posterior distribution, resulting in samples asymptotically following some desired distribution. Gibbs sampling is a part of Markov Chain Monte Carlo (MCMC) models[27]. Markov Chain models assume that X at some time t reveals information relevant for predicting future outcomes.

When drawing samples within the true distribution, it is desired to draw samples $q(z)$, where $p(z)$ is large[27]. This is due to the fact that the true distribution is best described around its most dense areas. Making it a Markov Chain Monte Carlo, stems from the definition that the next determined value of z only takes the current state into account, described formally:

$$P(z^{(t+1)}|z^{(0)}, ..., z^{(t)}) \tag{3.18}$$

Gibbs sampling defines that instead of determining all states of z at the same time, the probability is calculated separately for each dimension K[27]. Formally, this is expressed as:

$$P(z_i^{(t+1)}|z_1^{(t+1)}, ..., z_{i-1}^{(t+1)}, z_{i+1}^{(t)}, ..., z_K^{(t)}) \tag{3.19}$$

meaning that we draw a value of z_i, based on the current and updated states, of all the other z's.

Running Gibbs sampling a sufficient number of times, we eventually obtain values for the latent parameters that approximates the expectation of the desired

posterior, $E_{p(z)}[q(z)]$. Visually, the Gibbs 'walks around' (samples over) a desired distribution enough times to describe the posterior, shown in figure 3.1.

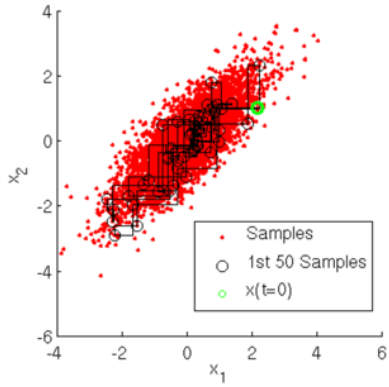


Figure 3.1: Gibbs sampling visualised. Source: [28]

3.5 The generative model

Generative models are a class of models for unsupervised learning, where our goal is to generate pseudo data that describes the true data. The objective is to improve an approximation of the true data from pseudo data at every iteration. Ideally, the generated pseudo data will resemble the true data. In this paper, the implemented probabilistic models are developed by telling a 'generative story', a story that explains how the true data came into existence originally. This is done by defining prior distributions associated with the parameters that explains how the real world was generated. As an example, the generative process for a Stochastic Block Model, explained in section 3.8, can be described as the following[29].

The process starts by generating z_n from a multinomial distribution, π from a beta distribution, and γ from a Dirichlet distribution[30].

1. Let $\boldsymbol{z_n} \sim \text{Multinomial}(\boldsymbol{\gamma})$ and $\boldsymbol{\pi} \sim \text{Beta}(\beta^+, \beta^-)$ and $\boldsymbol{\gamma} \sim \text{Dirichlet}(\alpha)$
2. Then, it is possible to generate the probability $p = \boldsymbol{z_n} \cdot \boldsymbol{\pi}_{\boldsymbol{z_n}} \cdot \boldsymbol{z_n}^\top$
3. and draw x_n from a Bernoulli distribution of $x \sim \text{Bernoulli}(p)$

Thus, the Stochastic Block Model knows how the true data was generated, therefore it is possible to generate pseudo data from the model. The pursued global variables underlie the latent structures of the dataset, and is shared for each observation. In the Stochastic Block Model, the model parameters γ, π, and α are the global variables, while the local variables z_i, affect the distribution of the $i'th$ iteration. Generative probabilistic models estimate the global variables by the joint distribution, in order to learn the latent variables and their true distribution[20].

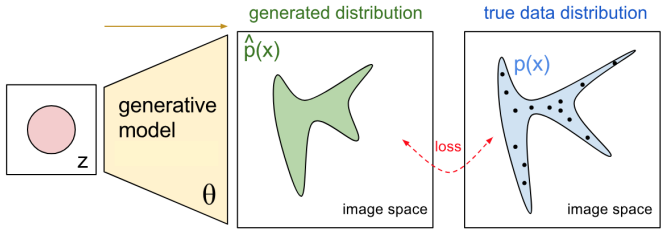


Figure 3.2: The generative process [31]

In the example visualised in figure 3.2, the blue region indicates the true data distribution, and the black dots represent data points. From section 3.3, it is known that a probabilistic model describes some distribution, and that the goal is to build a model that generates a new distribution as close to the true distribution as possible. Thus, an inference algorithm approximates the posterior distribution of the latent variables z_n from the joint distribution. Inputs to the joint distribution are the conditionals of the model parameters θ, and the generated distribution from the green region. Thereby, we compute the posterior distribution of the latent structure, to explore the assumed latent variables z_n, given a dataset. Upon sufficient iterations, the structure of the latent patterns is approximated, thus called the posterior distribution. For example, imagine the green region in figure 3.2 started out random, but then as the generative process proceeded, the model parameters θ iteratively changes, and shapes the green region to become a better representation of the true data distribution. In order to approximate the posterior distribution, some inference algorithm, for example KL-divergence, must be applied[32].

By summing it up, a generative process is a robust way to embrace specific anticipated latent structures from a probabilistic model. These features are preferable in this project, since it is possible explicitly present assumptions of how the world is designed instead of encoding them implicit. Furthermore, the use of likelihood scores, enables a comparison of probabilistic models, and derive probabilistic statements from given observations. Eventually, a generative probabilistic model makes it possible to predict future structures based on past

observations[18].

3.6 Gaussian Mixture Model

The Gaussian Mixture Model (GMM) is the most widely used mixture model, and is applied in this paper to cluster users on the bitcoin blockchain. Formally, GMM cluster observations into Gaussian distributed subpopulations by learning latent structures from the overall population[33]. In GMM, each subpopulation of the mixture is a multivariate Gaussian distribution, where μ_k is the cluster mean, and Σ_k is the covariance matrix [17].

$$p(x_n|\theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k) \tag{3.20}$$

As mentioned in section 3.3, in order to approximate the posterior distribution, the prior is multiplied with the anticipated structures of the data θ. The posterior distribution can also be formulated as Gaussian mixture.

$$q(x_n|\theta) = \sum_{k=1}^{K} \tilde{\pi_k} \mathcal{N}(x_n|\tilde{\mu_k}, \tilde{\Sigma_k}) \tag{3.21}$$

Where $\tilde{\pi_k}$, $\tilde{\mu_k}$ and $\tilde{\Sigma_k}$ are the updated model parameters using an inference algorithm in the generative process. In figure 3.3, an example of a mixture with four Gaussians illustrates the posterior inference. In figure 3.3 (a), a given dataset of observations are used to derive latent structures by approximating the posterior distribution of the mixture. In figure 3.3 (b) each observation has been assigned to the most likely cluster, visualised by colouring the data points, and the most likely cluster means μ_k are the centred grey points. The contours indicates the predictive distribution of which cluster a future data point will be assigned to[20].

Plot (b) in figure 3.3 is formed by the generative process for GMM, and are developed as following:

1. Starts by drawing mixture proportions

$$\boldsymbol{\pi} \sim \text{Dirichlet}(1) \tag{3.22}$$

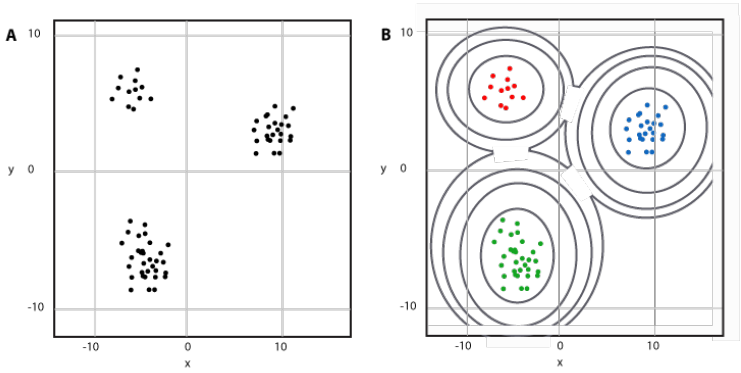


Figure 3.3: Example of a Gaussian Mixture Model

2. For each mixture component K

   $\boldsymbol{\mu_K} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ and $\boldsymbol{\Sigma}_K^2 \sim$ InverseGamma$(1, 1)$ is drawn.

3. For each observation n

   Mixture assignment $\boldsymbol{z_n}|\boldsymbol{\pi_n}$.

   Data point $\boldsymbol{x_n}|\boldsymbol{z_n}, \boldsymbol{\pi_n}, \boldsymbol{\mu_n}, \boldsymbol{\Sigma_n}$ is drawn.

π, μ and σ are all variational model parameters. Thus, a pseudo dataset is iteratively generated. Variational model parameters are the global variables, and z_i are the local variable to affect the distribution of the $i'th$ iteration.

3.7 Latent Space Model

Network data occurs everywhere. Typically it consists of actors and relational ties between these actors. In this case, actors represent users on the bitcoin blockchain, and the tie represent a transaction. An approach to analyse these networks is using the exponential family of random graph models. Exponential random graph models are a family of probability distributions on graphs. These models can be applied on various kind of graphs as: weighted, multiple-edged, or directed and are typically represented as an adjacency matrix A.
Within the category of exponential random graph models is a class of models, where it is assumed that the probability of a relational tie between two actors depends on their position in a latent space. In this section, we introduce Latent

Space Models (LSM), often called Latent Position Models. In the next section, we will present an alternative model: The Stochastic Block Model (SBM). LSM assumes that vertices are placed in a K-dimensional latent space, and the closer they are positioned, the more they tend to tie. The LSM's are widely used in many fields, as they provide the possibility of visualisation, given their geometry-based assumption, and are therefore an advantage when interpreting the network.
The LSM is applicable to data in the form of an $n \times n$ (n being nodes) adjacency matrix A, where each entry $a_{i,j}$ is denoting the value (strength) of the relational tie or whether the tie exists or not (dichotomous). When modelling, we assume that, whether or not a relational tie exists, is independent of all other ties, and the conditional probability of the adjacency matrix A is:

$$P(A \mid Z, X, \theta) = \prod_{i \neq j} P(a_{i,j} \mid z_i, z_j, x_{i,j}, \theta) \tag{3.23}$$

Where X and $x_{i,j}$ are the observed pair-specific characteristics, and θ and Z are the parameters and positions needed to be estimated. The work of Hoff et al.[14] employs a logistic function to model the distance, with the model parameters $\theta = \{\alpha, \beta\}$, intercept and coefficients respectively, giving the log odds of existing a tie between vertices i and j:

$$\eta_{i,j} = \text{logodds}(A_{i,j} = 1 \mid z_i, z_j, x_{i,j}, \alpha, \beta) \tag{3.24}$$

$$= \alpha + \beta' x_{i,j} - \|z_i - z_j\| \tag{3.25}$$

Meaning that whether a tie exists in A ($A_{i,j} = 1$), is determined by the attributes $x_{i,j}$, and the Euclidean distance ($\|z_i - z_j\|$) between vertices in the latent space [14].

In this project, we analyse a directed network of users transacting, as we seek to learn about the user transaction patterns, and predict who is likely to transact with who. Utilising the introduced LSM to model the bitcoin blockchain network, we will try to answer this by learning the latent embedding for each of the users in the network, to capture the similarities between them.

Presented below is the generative model, i.e. how the network of users transacting is believed to have been generated. The generative process is as follows:

1. Start by initialising a single

$$\sigma_\beta^2 \sim \text{InverseGamma}(10^{-3}, 10^{-3}) \tag{3.26}$$

$$\beta \sim \mathcal{N}(0, \sigma_\beta^2) \tag{3.27}$$

2. For each latent dimension k draw:

$$\boldsymbol{\sigma_k^2} \sim \text{InverseGamma}(10^{-3}, 10^{-3}) \tag{3.28}$$

3. For each vertex (user) $n = 1, ..., N$, draw:

$$\boldsymbol{z_i}, \boldsymbol{q_j} \sim \mathcal{N}(0, \boldsymbol{\sigma_k^2}) \tag{3.29}$$

4. For each tie generate:

$$\boldsymbol{A_{ij}} \sim \text{Bernoulli}(\sigma(-\|\boldsymbol{z_i} - \boldsymbol{q_j}\| + \beta)) \tag{3.30}$$

z_i, q_j and σ_z^2 are the local variables to affect the distribution of the $i'th$ iteration, and β, σ_β^2 are the global parameters shared for every observation in the latent space. All these latent variables are inferred given data in a way that characterises the network to the best of its abilites. To approximate the posterior, we define the random variables qz_i, qq_j, $q\sigma_z^2$, $q\beta$, and $q\sigma_\beta^2$, and use variational bayesian inference with the Kullback-Leibler divergence to infer these latent variables, given our data.

To account for the directed nature of the bitcoin network, a second latent variable q_j is introduced in LSM. Similar to z_i, q_j is also modelled with a normal distributed prior. This second latent variable is introduced, in order to explain the position of the user (embedding) in the latent space, both as the role of sender and receiver of a given transaction. If the embedding of sender and receiver is placed close together in the latent space, they transact. Contrary to this, if they are far apart, they do not transact. By modelling it this way, we are able to account for the fact that some users might often act as sender, but not as often as receiver. Example: If Bob is sending Bitcoins to Alice, and not receiving any from her, then Bob's latent embedding z_i would be close to Alice's latent embedding q_j, and therefore likely to generate a tie. On the other side, Bob's q_j would be far away from Alice's z_i in the latent space. By doing this, we able model whether the relation between users is reciprocate.

The role of β is to control the sparsity of the network, i.e. does the network contain many or few ties. This allow for the fact that vertices very close in the latent space might not be connected. If β is high, there are many ties in the network, and if it is low or negative, the network contains few ties. This bias is a noninformative prior distribution used to initialise the variance of β introduced by *Gelman 2006*, who recommends using $p(\sigma_\beta^2) \sim \text{InverseGamma}(10^{-3}, 10^{-3})$, as we wish to not constrain the posterior inference [34].

The sigmoid function maps the Euclidean distance and bias β to a probability $\pi_{i,j}$ of i and j forming a tie. We model the ties between users with a Bernoulli likelihood, given $\pi_{i,j} = \sigma(-\|\boldsymbol{z_i} - \boldsymbol{q_j}\| + \beta)$ (see equation 3.30).

3.8 Stochastic Block Models

Another exponential random graph model, is the Stochastic Block Model (SBM), where a graph implies a certain structure of objects, and objects that are tied, indicates a relation[35]. Given a graph network where vertices are either tied or not, (see figure 3.4a), SBM tries to identify a partition of the graph into subgroups that are alike, (see figure 3.4b). For example, all groups is more densely tied, with relative few links across the groups. The idea is that by partitioning vertices into some block structures, it is possible to derive blocks of communities from the network, where vertices are partitioned into the same block if a certain edge pattern is shared. At its core, SBM corresponds to a likelihood function that operates as a partition score function, where the probability of a link depends on which community the vertex belongs to. SBM is applied in different sorts of networks such as social networks, biological networks, communication networks etc. In this project, we seek to partition users on the bitcoin blockchain into communities by utilising the SBM[14].

(a) **(b)**

Figure 3.4: Left (a): A random graph network. Right (b): A partitioned graph network with $K = 5$ communities[36]

At its most essential form, the SBM is characterised by three parameters[37].

$$SBM(n, \pi, M)$$

- n: denotes the number of pair of vertices u, v in the graph.
- π: is a probability vector with the dimension $(1 \times n)$, $\pi = (\pi_1, ..., \pi_n)$ indicating a group index for each vertex.
- M: a $(K \times K)$ matrix indicating an edge probability between two vertices, where K is the number of communities.

Each observation n is assigned to a given community, K, thus if $u \in M_i$ and $v \in M_j$ are linking. Then, it is possible to estimate a matrix M of probabilities, where the element M_{π_i,π_j} denotes the probability of an edge between the vertices u and v. Note that K is chosen before π and M can be generated, and that vertices within the same community is identically linking to other vertices. The group index are given by a $n \times K$ matrix, where the highest probability for each vertex assigns the group index. Given these features, the SBM can both be applicable of juxtaposing the clusters found in GMM, and the link prediction in LSM, since SBM assigns a group index for every user, and a probability of forming a link for every pair u, v given π[38].

Given the generative process of SBM described in section 3.5, and K, π, and M, it is possible to generate a network iteratively, where each pair is associated with a probability M_{π_u,π_v} for that edge to exist. As mentioned, edges are iid, meaning that all edges are distributed in terms of their type of community, and depending on the edge distribution, SBM can generate different types of network structures. In table 3.2, some types are described[37]. In figure 3.5,

Table 3.2: Network types[37]

Type of structure	*Description*
Random	When $M_{i,j} = p$ is constant for all pairs i, j
Assortative	When $M_{i,i} > M_{i,j}$ for $i \neq j$
Disassortative	When $M_{i,i} < M_{i,j}$ for $i \neq j$
Core-periphery	When $M_{i,i} \approx M_{i,i+1} \approx M_{i,i-1}$
Ordered	When $M_{i,j}$ decreases as the community index increases

a given SBM is visualised by its input graph and a stochastic block matrix, which illustrate an instance of an assortative community. From the stochastic block matrix, it can be interpreted that vertices in this type of network tends to connect with vertices that are alike (i.e. in the same community), since the diagonal indicates a high probability. Furthermore, the graph illustrates more edges between vertices within a community than across. It could be a model of a social network, where a community denotes a high school, and it is seen that people within a given high school are connecting relatively more than across high schools. As mentioned, a graph can both be directed and undirected, where the bitcoin blockchain is by nature a directed network. The SBM can both handle directed and undirected graphs. This implies that in a directed graph, the edge probability $u \to v$ can differ from the edge $u \leftarrow v$.

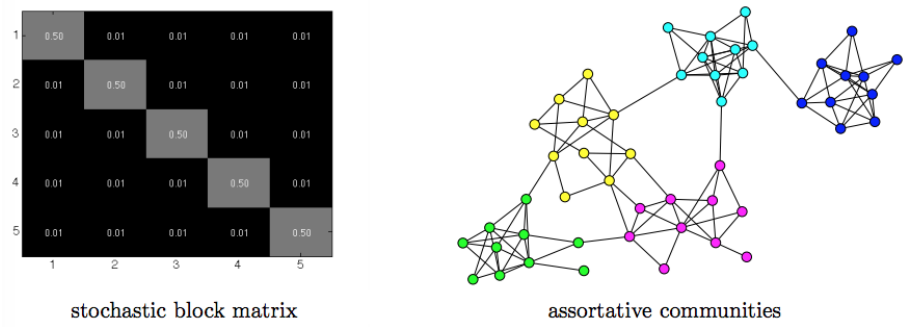


Figure 3.5: Stochastic block matrix and graph[37]

3.9 TensorFlow

TensorFlow is a framework for numerical computation using data flow graphs, meaning that it represents complex computations as graph vertices and edges being multidimensional data arrays called tensors. Using this flexible setup, TensorFlow can perform large-scale machine learning, by deploying and distributing the computations across hundreds of GPU's, or simply running it on your own laptop CPU. All this can be done without rewriting a single line of code. TensorFlow was developed by researchers of the Google Brain team in 2011, with the purpose of exploring and developing deep neural networks, but the framework is general enough to be applicable on other domains. TensorFlow allows users to instantiate these directed computational graphs, via a Python frontend. In figure 3.6, the TensorFlow computational graph can be seen as an empty shell of operations. Feeding data to the model is done by first defining a placeholder, like a promise to provide a value later with a specified shape. In order to propagate the data through the graph and evaluate the result, a TensorFlow session is instantiated and executed[39]. One of the main areas of focus of this project is scalability, thus our models have been implemented in the TensorFlow framework, since the properties of its architecture enables flexibility and scaling. TensorFlow is able to distribute tasks between devices, i.e. a cluster of computers, by distributing the computation graph among them. As the tasks are distributed, each task is assigned a parameter server where the master session is running, and a "worker" machine where the task is executed. The parameter server is responsible for holding all the model parameters and update them in response to incoming gradients, whereas the worker tasks are responsible for completing all the intensive parts of the computation.

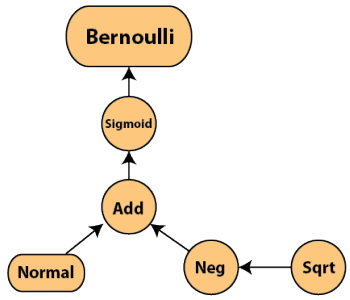


Figure 3.6: TensorFlow computation graph[39]

3.9.1 Edward

The python library Edward is a library for probabilistic modeling, infrence, and criticism. It is a Turing-complete probabilistic programming language, meaning that Edward is able to represent any computable probability distribution. Edward is led and actively developed by Dustin Tran, with guidance by David Blxei. Edward is built on TensorFlow, and draws on many of the features from Tensorflow, such as computational graphs, CPU/GPU integration, distributed training, and TensorBoard visualization. It allows for fast experimentation with probabilistic models, and is able to handle complex deep probabilistic models and large data sets at the same time. Edward combines the three fields, Bayesian statistics and machine learning, deep learning, and probabilistic programming[40].
The next version of the Edward library is to be merged into TensorFlow, as the probabilistic branch of TensorFlow.

Edward follows George Edward Pelham Box's philosophy of statistics and machine learning, and the Edwards library's functions enables one to simply cycle through Box's loop[20]. This means that Edward facilitates the iterative process

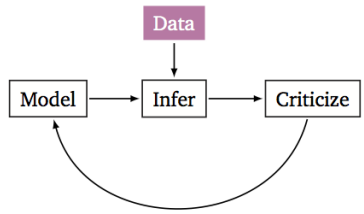


Figure 3.7: Box's loop; Build, Compute, Critique, Repeat [41]

of building and using latent variable models:

1. Formulate probabilistic model based on hidden structures believed to exist in data. Edward enables this step by providing a simple language of random variables to construct models.
2. Perform inference to approximate posterior. Edward provides a range of inference algorithms such as MAP and KL divergence, which uses existing optimizers inside TensorFlow.
3. Criticize the model by doing posterior predictive checks and predicting on unseen data. Revise model and repeat.

Our three models LSM, SBM, and GMM are implemented in the Edward and TensorFlow architecture. These implementations can be seen in appendix A.

Chapter 4

Results and Discussion

Having outlined our models along with the structures of a scalable framework, we now proceed to explain the findings. First, we will account for findings from the Link prediction using a Latent Space Model (LSM) and a Stochastic Block Model (SBM). The two models will represent the insights from the user network, thus how users transact with each other. Second, a Gaussian Mixture Model (GMM) is used to characterise the users on the bitcoin blockchain. With domain specific features, the user record dataset can reveal the behaviour of distinct users on the bitcoin blockchain. In addition, we will compare link prediction results from LSM and SBM, and cluster structures from SBM and GMM. Throughout the chapter, we will discuss results where relevant. Finally, we will discuss the impact of our data preparation on the results, and discuss the methodology relevance to the financial sector.

4.1 Link prediction

By utilising LSM presented in section 3.7 and SBM presented in 3.8, we now turn to the actual link prediction of bitcoin blockchain transactions described in section 2.4. Looking at Bitcoin transactions between the users who transact the most from January 2009 to December 2013, we will evaluate how well we are able to predict 1% of the most recent transactions, by treating them as missing.

To evaluate the predictions we use the area under the curve (AUC) of the receiver operator characteristic [42]. The AUC is evaluated on 1258 random zeros (non links), and the prediction of 1258 missing links. In our link prediction, it is important to note that the prediction is "crippled", in that we train the model to predict 0 (no link), where there actually is 1 (link) in the adjacency matrix, meaning that our prediction is rather a pseudo-prediction.

In this section, we will first look at LSM, then SBM, and lastly compare the two.

4.1.1 Latent Space Model

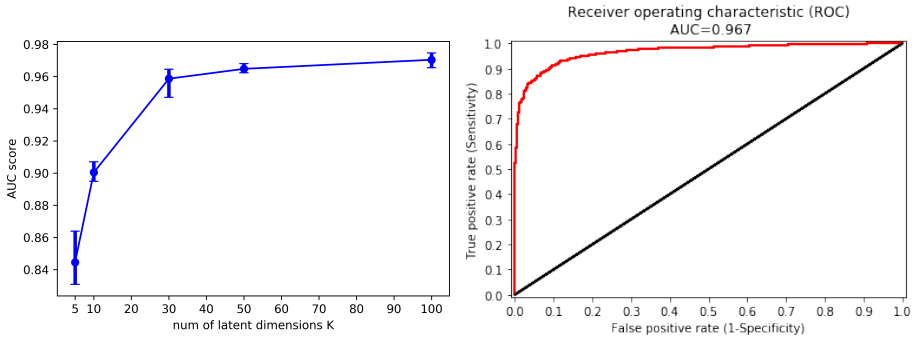
One of the goals of this project is to gain insights in the bitcoin blockchain. This goal can be obtained by applying an LSM (see appendix A) because it allows us, by link prediction, to tell how well it is capturing the structures of the network. Then, with a good prediction, we can visualise what the model captured about the network.

To select the best LSM, link prediction performance is evaluated. This is done with varying number of latent dimensions K and optimising with the variational Bayesian inference for 200 iterations, as all of the models converge well within 200 iterations (see figure 4.2). The results are averaged over 3 restarts, and presented in figure 4.1a, where the bars represent min, max, and average. As a baseline-model, we use the most simple network model, an Erdős–Rényi graph (see appendix A). This baseline-model predicts all links in the network based solely on the density of the network, i.e. not having a latent space to place the users in. The baseline-model results predict links with an AUC score of 0.5 (see appendix A.9). Achieving a significantly higher AUC score when using LSM will confirm that mapping the users in a latent space, does actually characterise the network.

We select LSM with $K = 30$ as the best model, since we see a relatively high increase in predictive performance (AUC), versus unnecessary increase in model complexity. In figure 4.1b the receiver operator characteristic curve for our selected model is shown.

Increasing the number of latent dimensions K further, we would expect to eventually see a decrease in the AUC score, as the model would overfit the data, i.e. when the LSM fit the links treated as missing links as actual missing links. Increasing K to equal the number users in the network N, we expect to see this overfitting, as the model is able to perfectly generate the network. However, this is not observed with parameters $K = 1000$ and $K = 2000$, and going

beyond this was deemed impractical, as running times exceeded 24 hours[1]. A reason why we are not observing this overfitting could be that the complexity of the network of Bitcoin users requires many latent dimensions, to allow sub-characterisation of the potential small subgroups of users. Another reason why we are not observing this overfitting could be because of Bayesian methods being very regularising and therefore generally less prone to overfitting.



(a) **(b)**

Figure 4.1: Left (a): LSM with AUC score as a function of number of latent dimensions K. Right (b): Receiver operator characteristic curve of LSM with $K = 30$

When optimising with the variational Bayesian inference, we look at the evidence lower bound (ELBO) to check for convergence. Figure 4.2 shows the selected LSM with latent dimensions $K = 30$, optimised with KL-divergence in 500 iterations. Here it is seen that it converges around 100 iterations. The LSM

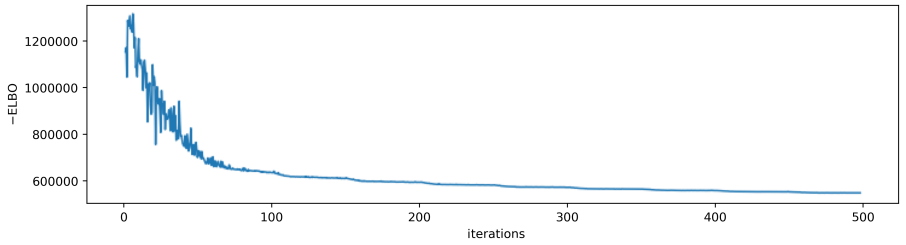


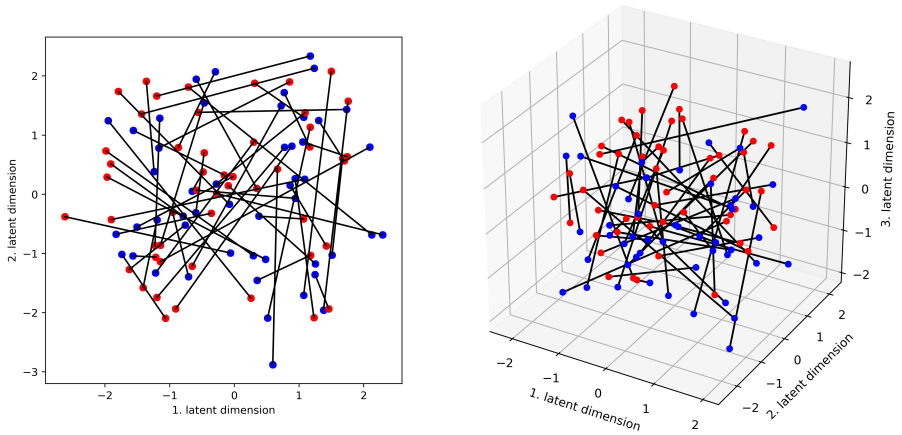
Figure 4.2: Latent space model, $K = 30$, 500 iterations (notice that loss is displayed in −ELBO).

predicts well, even with significantly lower dimensions than the best performing model. When predicting links with 2 and 3 latent dimensions, our results in

[1]On a Intel core i7-4720HQ 2.60Ghz CPU and 18GB ram

terms of the area under the curve (AUC) of the receiver operator characteristic are 0.76 and 0.80 respectively. Given our ability to make good predictions with few latent dimensions, it makes sense to visualise the locations of users in the latent space. Figure 4.3 shows these inferred locations in the latent space. From this, we are able to directly observe whether users are likely to transact, based on the relative distance between them (see appendix A.8). As described earlier, this is exactly one of the reasons, why LSM was used for this project.

In figure 4.3, pairs of qz_i (blue) and qq_j (red) are sender and receiver locations of 50 random users, visualised both in 2D and 3D. With both a sender and receiver location for each user, and a black line indicating that it is the same user, it is possible to evaluate whether the role of being a sender of a transaction is different to being a receiver. From this visualisation of locations, no apparent pattern is seen, indicating that the distances between sender and receiver of a single user are relatively shorter than distances between users. Therefore, the role of sending and receiving in this network is not the same, meaning that the probability of forming a link (doing a transaction), is in fact affected by whether the user is acting as sender or receiver, which is also expected in a directed graph. All 9968 users are visualised in appendix A.8. However, from the visualisation of users in the latent space, no apparent patterns or clusters of users as hoped are seen. As a sanity check, given the high AUC score, we test



(a) **(b)**

Figure 4.3: Representation of 50 users in the 2D and 3D latent space, where pairs of sender and receiver of the same user are represented by a red and blue point, and connected by a black line.

to see how well it performs on an increasing share of links treated as missing.

Predicting 5% missing links still score above 0.95 in AUC. In figure 4.4 it is shown how the AUC score decreases as the share of missing links are increased. As expected, when 99% of links are missing, LSM is not able to capture any structure in the network, resulting in an AUC of 0.52 The LSM has proven to

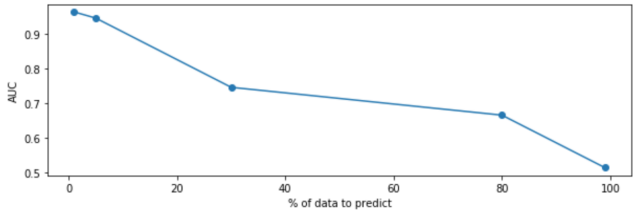


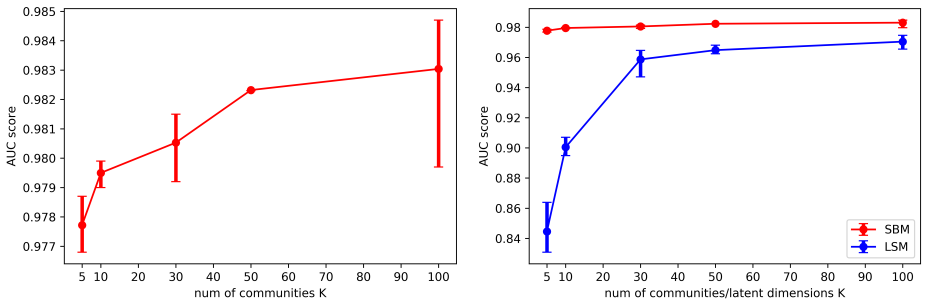
Figure 4.4: AUC score of LSM with an increasing share of missing links

model the users on the bitcoin blockchain well. With the link prediction, we are able to predict what transactions will occur between a subset of users on the bitcoin blockchain. The LSM provides a way to visualise a rather large and complex network, in a way, where the Euclidean distances between users is proportional to the probability of the users performing a transaction. Although the structure LSM is capturing is not directly visible in 2 and 3 dimensions, the SBM will perhaps do better capturing a structure on a group level, and present it in a visible manner.

4.1.2 Stochastic Block Model

Another way to model user transactions on the bitcoin blockchain, is with a SBM (see appendix A). Similar to LSM, the model performance is evaluated by its ability to perform link prediction, with MAP estimation. In figure 4.5a, the resulting AUC score of the SBM with varying number of communities K can be seen. Here it is seen that even with $K = 5$, the SBM is able to capture the structure of the network very well. Partition users with similar transaction profile into communities, turn out to be a better way to represent the users than LSM.

In order to select the best model, the same approach as with LSM is followed. Figure 4.5a shows the predictive performance (AUC) for SBM as a function of K, with 3 restarts. It is seen that the ability of SBM to predict links saturates quickly. Therefore, the choice of model is not obvious, however the AUC score indicates a relative large increase from $K = 5$ to $K = 10$. Therefore, we proceed using SBM with $K = 10$. As with $K > 10$, the model performance increases modestly in comparison to the cost of complexity. As in the LSM, it is eventually



(a) (b)

Figure 4.5: Left (a): SBM with AUC score as a function of number of communities, K. Right (b): LSM and SBM with AUC score as comparison of best performance.

expected to see a decrease in AUC as a result of overfitting, when increasing number of latent dimensions. Comparing the results of SBM with the results of LSM (see figure 4.5b), it is seen that SBM performs link predictions significantly better than LSM with $K < 30$. Furthermore, the AUC score of SBM varies less than LSM over 3 restarts. With SBM being the simpler model of the two, and clearly capturing the structure better, there is no reason to use the more complex LSM. A reason why SBM is better than LSM could be that it is the more correct model, meaning that users on the bitcoin blockchain actually do belong in groups defined by similar link pattern.

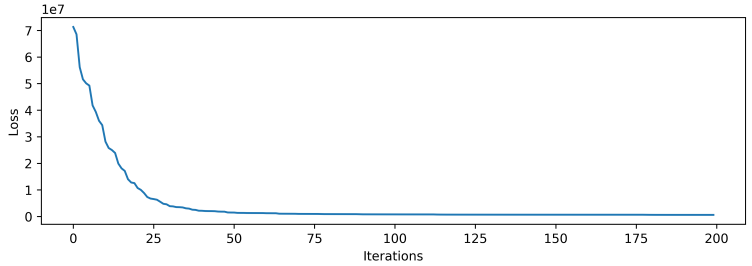


Figure 4.6: Stachastic block model, $K = 10$, 200 iterations (notice that loss is the negative log-likelihood)

To check for convergence, figure 4.6 shows the selected SBM with $k = 10$ estimated with MAP in 200 iterations. Here it is seen that it converges before 50 iterations. The plan was originally to utilise variational inference, but an

implementation was not supported in Edward. The chosen model with $k = 10$ partitions the users into communities, as indicated in table 4.1. The table shows the proportion of each community.

Table 4.1: Inferred γ denoting the proportion of users in each community

C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9
0.0002	0.0096	0.0002	0.0069	0.93	0.0004	0.05	0.0007	0.0008	0.0003

The SBM model (see appendix), makes it possible to compress the network of users into a network of communities. This network is expressed by the stochastic block matrix. The matrix gives an edge probability that a user of community i is linking to a user of community j. The predicted probabilities is visualised in figure 4.7. Note that M_i is the sending position, and M_j the receiving position.

	0	1	2	3	4	5	6	7	8	9
0	0,192	0,139	0,232	0,099	0,0948	0,23	0,077	0,148	0,215	0,229
1	0,097	0,02	0,122	0,047	0,0118	0,103	0,012	0,114	0,056	0,136
2	0,257	0,076	0,259	0,143	0,0699	0,222	0,106	0,199	0,231	0,243
3	0,089	0,035	0,091	0,032	0,0173	0,061	0,017	0,057	0,089	0,137
4	0,088	0,009	0,104	0,015	0,0003	0,106	0,002	0,07	0,043	0,202
5	0,165	0,064	0,11	0,078	0,0698	0,085	0,083	0,184	0,094	0,129
6	0,116	0,013	0,112	0,014	0,0017	0,103	0,003	0,091	0,091	0,119
7	0,17	0,055	0,107	0,095	0,061	0,13	0,06	0,079	0,114	0,22
8	0,151	0,058	0,105	0,09	0,1305	0,191	0,092	0,133	0,108	0,201
9	0,253	0,09	0,257	0,129	0,1903	0,123	0,081	0,179	0,14	0,259

Figure 4.7: The stochastic block matrix holding the probabilities M_{π_i,π_j}

Given the examples of network structures in section 3.8, it is difficult to get some intuition from figure 4.7. Except for three communities, the matrix shows a relatively weak diagonal, which indicates a diassortative network i.e. the users are connecting relatively more across communities. Nonetheless, the network also indicates assortative characteristics, i.e. the users are connecting relatively more within communities. Overall, it is difficult to capture one single network type describing the 9968 most active users on the bitcoin blockchain. This is

perhaps a result of analysing a subset of a large-scale network such as the bitcoin blockchain, in particular when the subset is filtered with no evidence of the filtered users being connected in any sense. No rule says that the most active users of the blockchain are linking relatively more with each other than with the less active users. However, it is possible to relate communities to each other by figure 4.7.

From table 4.1 it is observed that C_4 holds the largest proportion of users. Then, by looking at the matrix, it can be concluded that most users on the blockchain mainly transfer Bitcoins with C_9, since both M_{π_4,π_9} and M_{π_9,π_4} indicate a relatively high probability. This could indicate that C_9, which holds a small proportion, is some Bitcoin-cash exchange C_4 uses, and is supported by the relatively high probability of receiving Bitcoins from any community. C_4 on the other hand, tends not to link within the community, which could indicate a divers community of both investors, consumers etc. that all share the same Bitcoin-cash exchange. Since Bitcoins, at the time of writing, are not used for everyday tasks on a larger scale, it could make sense to see the largest proportion of users mostly interacting with exchanges. This is due to the suggestion that the majority of users today still primarily consists of enthusiasts and early adopters, whose transactions would mostly be with Bitcoin-cash exchanges.

In general, figure 4.7 tells us that we have a few communities, C_0, C_2, C_9, which tend to both link across communities and within their community. But it also tells us that a majority of communities do not tend to link within their communities, i.e. they share the same sending and receiving addresses outside the community. For instance, C_1, C_3, and C_4 all have in common that they tend to link with C_0, C_2, and C_9, but not with each other. Therefore, because SBM has partitioned the network based on a given transaction profile, it is possible to identify which users are prone to receive Bitcoins from who, given their community. Thus, it identifies a flow of Bitcoins between communities. In figure 4.7 a circulating flow between C_0, C_2, and C_9 can be sensed.

The SBM has proven to model the transactions patterns between users on the bitcoin blockchain better than LSM, performing a better and more consistent link prediction. The SBM provides a way to visualise a large and complex network in a way, where users are grouped together with other users who share the same transaction profile, thereby projecting complexity down to a visual structure. It is worth mentioning, that since LSM and SBM both captures the present network structure fairly well, it is possible to predict the network structure in the future. Therefore, it is possible to tell which users are most likely to link. The AUC score even indicates that deviations from the predicted should lead to suspicion. Furthermore, the SBM has shown that users partitioned in communities indicate how they act at group-level, and how they are transacting across these communities. It would be possible to characterise these commu-

nities, given features about their behaviour, to gain further insights. As the Gaussian mixture model assigns users into clusters, it will become relevant to compare partitions, as these could indicate similarities.

4.2 Anomaly detection

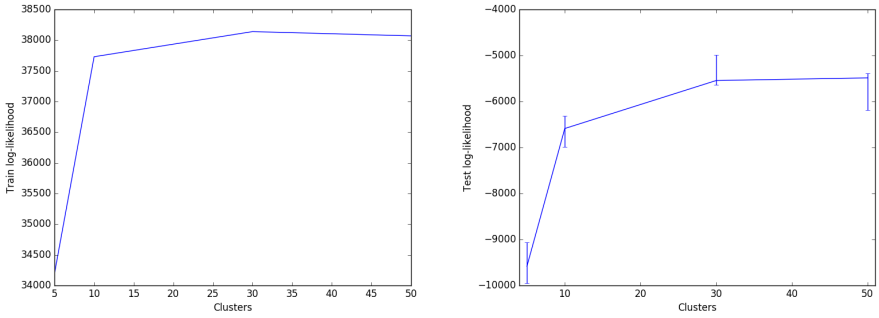
A GMM enables anomaly detection of newly generated data points. A part of the goal for this project, is to enable detection of abnormal users from so-called 'normal' users. This section will first identify the best model and reveal findings from this model. We assume that our training dataset only contain 'normal' user. Clusters of 'normal' users are characterised and described, thus making them meaningful. Second, we feed the trained model with held-out data, thus new data points are assigned to the clusters. This will show new data points' density and suggest anomalies. Lastly, the GMM cluster assignments are compared to the community assignment obtained from SBM, on the same users. The comparison uses the Rand Index to compare partitions.

4.2.1 Gaussian Mixture Model

The best performing GMM (see appendix A) is selected from its performance on test data. The user record dataset of 9968 observations is split into test- and train data using hold-out method. Using hold-out, a test and training set of 4984 observations are obtained. During training, 800 Gibbs samples are used, as the log-likelihood is seen to converge at around 600 iterations. Every model is trained three times in order to account for variation in test log-likelihood. Figure 4.8 gives an overview of model performance as a function of K, where K is the number of clusters the model aims to partition observations into.

Training results show an increasing performance up until $K = 30$. Thereafter, an almost flat curve showing no further improvement is observed. The test log-likelihood shows an increasing curve, with a maximum value where $K = 30$. This pattern gives a clear signal that $K = 30$ is the best model. To be clear, the best model is chosen by the best test log-likelihood. It is always desired to keep complexity low, while obtaining good results. Therefore, a balance between results and complexity has to be considered. In this case, we obtain the best result for the cost of complexity where the test log-likelihood has its peak.

Given the best model, it is possible to explore the cluster shapes from the



(a) Train log-likelihood **(b)** Test log-likelihood

Figure 4.8: Test and train log-likelihood as a function of number of clusters. Test log-likelihood shown with best, worst and median performance over three chains.

training process. The training set defined for this project is assumed to contain 'normal' users. Thus, obtaining clusters from the training, any new observations with predicted low densities are assumed to deviate from so-called 'normal behaviour', i.e. having abnormal behaviour. To be clear, the anomaly detection assumes that the training set provides a truth about what 'normal behaviour' is. In reality, this assumption would not hold, though it seems reasonable for the scope of this project. For the purpose of demonstrating a more correct approach, one should obtain a dataset with actual normal users, and then apply GMM as an unsupervised learning technique. Given this perspective, the project exploits unsupervised learning techniques in a very unsupervised manner.

Figure 4.9 visualises how the training and test data is assigned different clusters. The 4984 observations are clustered into $K = 30$ clusters.

On the training-set over 80% of all users are assigned to cluster 0. This suggests

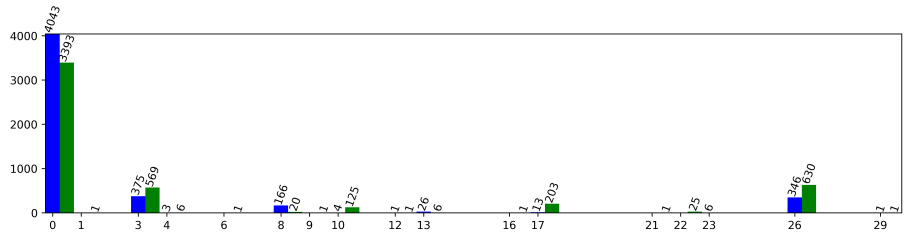


Figure 4.9: Cluster assignments of training and test data. Blue indicates train and green indicated test

no clear structure of the majority of users on the bitcoin blockchain. The remaining 20% of users from the training-set are assigned to 10 different clusters. Most striking are the two blue cluster-bars only containing one observation. In total, the training results in 19 of the 30 clusters not holding any users. Even though the model does not utilise these 19 of its 30 clusters, the best results are obtained with this model. This serves purely as an indicator, that the training and test data are relatively similar, as we see the most populated clusters are the same across both datasets. This gives an idea that some new users from the test-set might now hold the same behaviour as the users assumed to be normal in the training set.
Figure 4.9 gives an impression of the cluster proportions, however no further insights into which users these clusters contain. We now turn to the motivation of what 'normal users' on the bitcoin blockchain *could* look like, and later which user-characteristics the clusters hold.

Intuitively, users on the bitcoin blockchain *can* be assigned to the assumed following clusters:

- *Early adopters* should indicate a high active duration for both the pseudonym and user, and thus have performed a moderate number of transactions.
- *Working abroad* should perform many transactions, by transferring Bitcoins to their families abroad. Average amounts sent and received should be low, and average out degree should be higher than average in degree.
- *Long-term investors* should indicate a hold position by a high active duration per pseudonym, and the average amount received should be relatively high.
- *Day-traders* should have performed more transactions than the *Long-term investors*, and have a low active duration per pseudonym.
- *Mixing services* should have a significantly higher average out degree than the average in degree, and thus send smaller amounts of Bitcoins.
- *Consumers* should utilise Bitcoins in their everyday tasks, thus untraceability is irrelevant, and their number of pseudonyms should be lower or equal to the average. Furthermore, the average amount both sent and received should be moderate, and approximately at the same level.
- *Suspicious users* should try to increase their untraceability by controlling a myriad of pseudonyms. They should try to keep their Bitcoins in a constant flow between pseudonyms, and thus indicate a low active duration per pseudonym. Furthermore, the average amount both sent and received should be approximately at the same level.

The GMM partitions users into 11 different clusters. We have chosen six of these clusters summarised in table 4.2. These summary statistics are used in order to discuss cluster assignments, based on the intuitions mentioned above.

Table 4.2: Summary of users behaviour in the chosen six of 11 clusters

Cluster Labels	*0*	*4*	*10*	*13*	*23*	*26*
Avg. #txn.	6271	1092	1112	23849	142723	24144
Avg. #pseud.	765	162	168	7924	8439	6201
Avg. sent (BTC)	3,66	0,37	6,06	2,21	39,07	3,47
Avg. received (BTC)	3,62	0,41	6,05	2,20	39,06	3,46
Avg. in degree	81	7	5	12	2234	152
Avg. out degree	97	20	16	84	179	189
Avg. Active pseud. (days)	162	181	224	120	122	145
Avg. Active duration (days)	289	467	399	319	283	300

- *Cluster 0* is the largest cluster. Users appear as non-suspicious, since they have both a moderate number of pseudonyms, and performed transactions. The average active duration per pseudonym is low, and the users appear more as input to transactions than output, which could indicate that these users utilise Bitcoins in order to transfer money on a regular basis. It could be *consumers* transferring money to different merchants. This is reinforced by the relatively high number of out degrees. However, more than 80% of the users are placed in *cluster 0*, and it is plausible that the cluster contains smaller subgroups. The characteristics could also indicate different types of *investors* for instance.

- *Cluster 4* and *cluster 10* are two of the smallest clusters, with only a handful of users. These users have the same low number of pseudonyms and performed transactions, and interact with the same low amount of different pseudonyms. Their average active duration is high and roughly the same, which could indicate that the clusters contain *early adopters*. However, they differ by average active duration, and the average amount send and received, meaning our hypothesis on one type of an *early adopter* could be misleading. It could also indicate that *cluster 10* contains *Long-term investors*, since their pseudonyms are active in long time intervals, and that they perform relatively high value transactions.

- *Cluster 13* is performing many low value transactions. The average number of pseudonyms is high but active in short intervals, which could indicate that these users utilise Bitcoin in order to transfer money on a regular basis. It could be migrants and people *working abroad*, transferring money to their families back home. This is supported by a relatively low in degree, and a moderate out degree.

- *Cluster 23* has a high average number of pseudonyms. Compared to *cluster 0*, the number of performed transactions has been much more intense, since 142,723 transactions have been performed at the same time. Though, the average active duration for 283 days, the average active pseudonym is only active 122 days, i.e. the flow of Bitcoins between pseudonyms has been relatively high. Furthermore, the average amount sent and received is extremely high, and a high in degree could indicate a connection to a *mixing service*. These characteristics fits the profile of a *suspicious user* very well, but since the characteristics are purely based on assumptions, they have to be further investigated before any accusation can be justified.

- *Cluster 26* has a high average number of pseudonyms and low average active duration per pseudonym. In addition, the users are performing a high number of transactions. The average in and out degree indicates that the users interact with a large number of different distinct addresses. Taking the features into account, the cluster could contain a fair amount of *Day-traders*, since they perform several transaction in short time.

The above characterised clusters are the clusters assumed to contain so-called 'normal' users. When testing new data on our model, these new users will be assigned to one of the 30 clusters, some characterised above. Thereby, we are able to distinct new behaviour from past observed behaviour, and thus classify this as abnormal.

Given the assumption, that the training set holds normal users, we would like to enable anomaly detection from the new datapoints' density estimations. Being able to perform anomaly detection, could show whether some of the most abnormal users are being assigned new partitions (partitions not used by the classifier during training), or whether the abnormal users will be classified to the same partition same as the 'normal' users. From our results, we are able to rank users by their degree of abnormal behaviour they show on the bitcoin blockchain (c.f. their density), and see which clusters these users belong to. To explore this, we first summarise the top 10 most abnormal users, and compare them to the overall dataset, on the features where they are significantly different.

Table 4.3: The most significant average differences from the 10 most abnormal users and all 'normal' users

	Avg. Pseud.	Avg. Txn.	Avg. in degree	Avg. out degree
Full dataset	639.58	3974.31	66.86	88.72
Top 10	91	442	6.89	160.35

As seen from table 4.3 the top 10 users seems to be significantly less active than the average 'normal' user.
Looking further into every one of the 10 users (see appendix A.10) reveals two different patterns; little activity (few transactions) but no further important indications, and little activity with a significantly high number of pseudonyms per transactions relative to the mean.
This suggests two things. First, the model will handle users with very little activity as abnormal, since their characteristics does not reflect a behaviour. Behaviour is, from the models perspective, given from the features, and the more transactions a users performs, the more likely the user is to reveal a pattern. Thus, low volume of transactions means no clear behaviour, which results in abnormal behaviour. Second, two users of the top 10 most abnormal users have an almost one-to-one pseudonym to transaction ratio. This seems suspicious as it could indicate an attempt to hide any trails. Furthermore, given the fact that these two users have a very short lifespan, and low in and out degrees, it signals low interaction with other nodes in the network only doing one-time transfers, most probably with themselves. Given the observations on the two suspicious users, we would classify these as suspicious.

To compare partitions from our SBM, we use the Rand Index[19] and the Adjusted Rand Index[43][44]. The comparison is only possible, because of the two different datasets, user record and user network, have kept the "UserID" attribute throughout the data-processing and modelling. Given that results from the two datasets are comparable, enables an examination of whether the two approaches yields the same results in terms of clustering. If they do, a relationship between behaviour on the blockchain (user record) and who they are transacting with (user network) can be assumed. The user record approach is clustering users, who have the same attributes in terms of behaviour on the bitcoin blockchain, and the user network groups people who are transacting in the same way relative to other users.
If the Rand Index would indicate incomparable partitions, there would not be any evidence for the suggestion of dependencies between user-behaviour and transaction pattern.

The Rand Index for cluster and group assignments respectively yields a score of 0.66. However, to remain critical to obtained results, we measure the Rand Index for the two cluster assignments shuffled in random order. If this calculation yields a significantly lower score, the first result will be acceptable and interpreted as high similarity between partitions.
Shuffling the order of the clusters and calculating the Rand Index, yields a score of 0.65. This suggests a high amount of cluster and group assignments to different clusters, which in general yields results close to 1. Immediately, there is no real difference. We proceed the examination with an Adjusted Rand Index.

The Adjusted Rand Index yields a result of 0.039, suggesting low similarity between partitions from GMM and SBM. In comparison, random shuffling results in a score of -0.014. The evidence adjusted for biases suggests no similarity between cluster partitions.

Results from the Adjusted Rand Index questions the hypothesis that user behaviour, and who they transact with, are dependent. A discussion of whether this result seems natural or whether only applying the first heuristic has influence, seems appropriate.
If the preprocessing had accounted for change addresses as well (i.e. the second heuristic had been applied), less users would have been obtained, since some of the distinct users would be assigned to the same user. Furthermore, it would be more eloquent of the way users link. It is not possible to comment on whether the second heuristic could reveal better results.
Though, it is intuitive that the same users interacts with the same groups, this effect might be obscure on the bitcoin blockchain c.f. the anonymity it provides.

The findings are all based on the best model, and the stated definition of what 'normal' users are.

4.3 Discussion of data preprocessing

Given our data preprocessing, and the fact that we are only modelling a small subset of the full bitcoin blockchain, the value of our results are limited and considerations need to be made:

- First of all, Bitcoin transactions are regarded as belonging to the user, if they act as input addresses in a given transaction (first heuristic), which affects the result by obscuring transactions performed with centralised exchanges. These transactions will likely all be assigned to one user.
- Both the link prediction and anomaly detection are only considering transactions between the 9968 most active users (by number of transactions), our analysis will therefore by highly affected by the services described above.
- The transactions SBM and LSM are able to predict with high certainty, cannot be interpreted as small day to day Bitcoin transactions, but might rather be obvious links between exchanges or other services, representing

thousands of individual transactions. This is a result of how we subset the full data.

- By representing the network of user transactions as a graph, we are not able to model multiple transactions between the same users. The SBM and LSM are therefore not able to characterise whether a link between users represents thousands of transactions, or just a single one. To model this effect, a weighted graph could be considered.

4.4 Framework for the financial sector

As stated in chapter 1, our main motivation of this project is to gain new insights into the bitcoin blockchain, and maybe even eliminate with some of the scepticism associated with Bitcoins. The reasons why Bitcoin is associated with scepticism is i.a. because of the volatile value, and the supranational nature of the blockchain. However, the primary reason is the untraceability of its users. The anonymity properties prevent the bitcoin blockchain to abide by the Financial Regulations such as Anti Money Laundering (AML) and Know Your Customer (KYC)[7]. This means that financial institutions can not pursue business opportunities in the Bitcoin domain, before they can be sure to know their clients.

It is not unrealistic to imagine that the financial sector at some point wants to provide services within the blockchain/cryptocurrency sector. Either because it provides a potential lucrative business opportunity, or because customers demand these services. Given such circumstances, this project can be relevant, since it provides a tool that can assistance compliance to regulations.
By analysing blockchain transactions, we enable raising potential red flags at users, who indicates a suspicious behaviour. This behaviour can be both who they are transacting with, and how they act on the blockchain.

Chapter 5

Conclusion

Transaction data from the bitcoin blockchain is one of the largest publicly available transaction datasets, but it has few patterns or insights, since users are anonymised through unidentifiable cryptographic pseudonyms (addresses). The absence of central authorities to maintain common financial regulations, has made Bitcoins a disputed field. Given the high complexity of the domain, the purpose of this project was to extract meaningful insights from the bitcoin blockchain, which is currently a black box.

A part of the scope of this project was to obtain insights through meaningful clusters, anomaly detection, and predictions of future transactions.
We used a Gaussian Mixture Model with $K = 30$ and a Stochastic Block Model with $K = 10$, to extract meaningful clusters with two different approaches: a user record approach, and a user network approach. With both models, we were able to categorise users into clusters, and characterise these using domain specific knowledge. This is what we define as *meaningful* clusters. The fact that we are able to use domain specific features obtained from the blockchain, makes the interpretation less prone to misconceptions and thus supporting our contribution to information gain. Concretely, the six most noteworthy clusters from the Gaussian Mixture Model were characterised by summary statistics and interpretation.

For the link prediction problem, we used a Stochastic Block Model with $K = 10$,

and a Latent Space Model with $K = 30$. Link prediction is closely related to predict a future flow of Bitcoins. Therefore, we decided to apply the user network approach for this purpose. Our results show that we can systematically predict future transactions between current users on the bitcoin blockchain. This prediction is based entirely on the hidden structures of the user network consisting of transactions. The performance evaluation area under curve (AUC) of the receiver operating characteristics, yields scores of 0.967 and 0.98 for the Latent Space Model and the Stochastic Block Model, respectively. The results also reveals that the Stochastic Block Model is better at capturing hidden structures than the Latent Space Model. A Stochastic Block Model partition users into groups, whereas a Latent Space Model places users into a latent space. Therefore, we conclude that a Stochastic Block Model is the most correct model to absorb structures of the bitcoin blockchain, since it can capture the group structures assumed to exist between users.

Identification of bitcoin transactions and extraction of meaningful clusters enables anomaly detection on the bitcoin blockchain. All three models contributes to anomaly detection. The Gaussian Mixture Model enables detection of users deviating from the defined 'normal' behaviour. Thus, this anomaly detection is based on domain features that characterises behaviour on the bitcoin blockchain. Given held out data, we were able to detect abnormal users, and describe these exact users.
Furthermore, the link prediction of both the Latent Space Model and Stochastic Block Model can likewise serve to detect anomalies with high accuracy. Observing the transactions on the bitcoin blockchain, new links created between users with predicted low probability, can serve as a red flag about suspicious transactions, if it turns out to be a link.

Given the size of the data currently available on the blockchain, all models are implemented in the high scale architecture TensorFlowm using the Edward library as a probabilistic modelling tool. This fulfils the purpose of enabling high scale modelling of the bitcoin blockchain, potentially including all transactions available.

In conclusion, unsupervised learning and the proposed models enabled us to explore and provide hidden, yet meaningful insights about a very complex and relatively unexplored domain.

5.1 Future work

For future work, we would suggest initiatives that can result in improvements revealing more meaningful information. We suggest initiatives on three areas; data, architecture, and modelling.

Firstly for data improvements, we would apply the second heuristic (change addresses), which could potentially reveal new insights. Publicly available address tags are available, which would indicate the true person/company behind some addresses, e.g. addresses on specific exchanges and services. Furthermore, exploring an updated dataset with transactions not accounted for in this project (all transactions after December 28th 2013) could show new behaviours.

Second, we would implement the scalable models on machines having the ability to utilise the architecture. Enabling high performance computing is a prerequisite to run on larger datasets, thus it could open up for potentially undiscovered hidden structures.

Lastly, implementing variational Bayesian inference, $KL(q||p)$ on the Stochastic Block Model instead of MAP inference, would enable the model to deal with the assumed uncertainty of the users on the bitcoin blockchain. Using variational inference on the Stochastic Block Model would also make the comparison more meaningful. Obtaining a user record dataset containing users known to act 'normal', would further improve the interpretation and correctness of abnormal users.

Appendix A

Appendix

Information gathering

Preliminary academic information retrieval

Title	1-3	Data	Data Mining Methods	Problem statement	Year
An analysis of anonymity in the bitcoin system	3	Using the Bitcoin client and a modified version of Gavin Andresen's bitcointools.	Defines a topological structure of two networks derived from Bitcoin's public transaction history. - **not data mining but a passive analysis**.	The motivation for this analysis is not to de-anonymize individual users, but to demonstrate, using a passive analysis of a publicly available dataset, the inherent limits of anonymity when using Bitcoin.	2013
Bitcoin Transaction Graph Analysis	2	Scraping bitcoin addresses from public forums. ("real" name, public key) pairs. Full bitcoin core & "Armory" to parse through the blockchain.	• Graph analysis (80,030 unique vertices)	Explore the level of anonymity in the Bitcoin system. Two-fold approach: (i) annotate the public transaction graph by linking bitcoin public keys to real people - either definitively or statistically. (ii) run the annotated graph through our graph-analysis framework to find and summarize activity of both known and unknown users. **The goal is to associate numerous unrelated cryptographic IDs with an actual user**	
A fistful of Bitcoins	3	Unknown.	Unknown.	This article examines the limitations of Bitcoin anonymity and discover that the ability to cluster pseudonyms according to heuristics about shared ownership allows us to identify (i.e., associate with a real-world entity or user) a significant and active slice of the Bitcoin economy	2016
Unsupervised Learning of Bitcoin Transaction Data	3	• Raw data from the Blockchain • Tag data from Blockchain.info • Table of historic BTC/USD conversion rates (with time granularity of one day)	• Principal Component Analysis (PCA) • K-means clustering • C-means clustering with fuzzy logic • CURE Clustering Algorithm (utilizing a sampling method) • Hierarchical clustering	• Building on the works of Reid and Harrigan, and Brugere • Cluster each transaction within the Bitcoin network utilizing various unsupervised machine learning algorithms • Measure the effectiveness of these clusters in an objective manner. • Evaluate the computational and time requirements to form such clusters. • Compile a list of potentially anomalous transactions.	2014
Quantitative Analysis of the Full Bitcoin Transaction Graph	2	180,000 HTML files. (all the transactions ever carried out in the Bitcoin system)	• Variant of a Union-Find graph algorithm (addresses which are expected to belong to the same user).Introduction to Algorithms, Second Edition	In this paper we answer for the rst time a variety of interest- ing questions about the typical behavior of users, how they acquire and how they spend their bitcoins, the balance of bitcoins they keep in their accounts, and how they move bitcoins between their various accounts in order to better protect their privacy.	2012
Anomaly Detection in Bitcoin Network Using Unsupervised Learning Methods	3	Two graphs generated by the Bitcoin transaction network: one graph has users as nodes, and the other has transactions as nodes.	• K-means clustering • Mahalanobis distance • Unsupervised Support Vector Machine (SVM)	This paper consider anomaly detection particular to the Bitcoin transaction network. The goal is to detect which users and transactions are the most suspicious; in this case, anomalous behavior is a proxy for suspicious behavior.	2016
Unsupervised Approaches to Detecting Anomalous Behavior in the Bitcoin Transaction Network	2	The bitcoin dataset was obtained from http://compbio.cs.uic.edu/data/bitcoin/ , generously made available by Ivan Brugere.	• K-means clustering • Unsupervised learning algorithm, RolX	Using ML techniques to explore a dataset of bitcoin transactions; in particular, the anonymity guarantees. • Can we cluster the dataset in order to make exploration? • Can we detect attempts at money laundering? • Can we trace the outputs of a mixing service back to its inputs?	

Figure A.1: Preliminary academic information retrieval, part 1

How to de-anonymize Bitcoin (coursera)	**2**	No data, this is a 20 min video providing an overview of different methods.	No mining methods. He points out that there are two layers to base research in regards to anonymity. One approach is to use the blockchain data - data recorded directly on the blockchain. Another way is to use a network-layer approach, where one tries to catch an IP-adress of new nodes when they are connecting to the blockchain.	Overview of approaches. Blockchain: - A fistful of Bitcoins - An analysis of anonymity in the bitcoin system Network layer: - "The first node to inform you of a transaction is probably the source of it", Dan Kaminsky, 2011.	
Behavior pattern clustering in blockchain networks	**1**	They do not mention the origin of their data, neither do they give indications on the structure.	They use three methods to compare the different precisions. They use Density Based Method, Hierarchical Clustering Method, and Behavior Pattern Clustering (BPC). All three methods are run on some data not revealed, and precisions are compared. Conclusion is that they propose BPC to be a new way of clustering behacior.	The goal is to cluster the n sequences into k clusters such that sequences in the same cluster are similar to each other while sequences from different clusters are not similar.	2017
Analyzing the Bitcoin network: the first four years	**2**	Data used is public available data (not accessible anymore). The link is in reference [16]	There are no particular mining methods, the article is descriptive statistics from the first four years in the Bitcoin ecosystem. Is generally shows how different countries have been using Bitcoins, what kind of different services have had the most volume of transactions etc. They have identified hubs for Bitcoin flows and analysed where different transaction values most typically flow (e.g. small transactions on gambling, large transactions on exchanges).	We examine the economy and transaction network of the decentralized digital currency Bitcoin during the first four years of its existence. The objective is to develop insights into the evolution of the Bitcoin economy during this period.	2016
Bitcoin transactions: a digital discovery of illicit activity on the blockchain	**1**	The model they develop is never practically tested, but is more of a theoretical model that can be used in the future.	They use IP addresses etc. to tro to find the real persons behind a Bitcoin address.	A review of machine learning techniques and heuristics was carried out to learn how certain behaviours from the Bitcoin network could be augmented with social media technology and other data to identify illicit transactions.	2017
Unsupervised learning for robust Bitcoin fraud detection	**3**	Bitcoin dataset housed by the Laboratory for Computational Biology at the University of Illinois. All transactions from the genesis block to blockchain 230686 dated 7 April 2013.	• K-means clustering	1. The main objective of this paper is to find and classify anomalies on the Bitcoin network based on transaction patterns. This will serve as an aide to detect financial fraud and associated activities such as money laundering. 2. Secondary to that, the paper also seeks to assess performance of the anomaly detection algorithms using publicly available Bitcoin transaction data from blockchain. 3. The study will make an assessment results in relation to the impact of HDM.	2016
Blockchain Transaction Analysis using Domain Sets	**2**	A vector of 2048 blockchain transactions. They do not go into further details	Domain set approach. They are using this method to cluster the different transactions. They are using K-means clustering as well, but finds that domain sets gives a higher accuracy.	The main objective is to cluster transactions without mentioning the number of clusters in advance. They will use two different clustering approaches to determine the best suitable algorithm for this, measured by percentage accuracy. .	2017
Deanonymizing the bitcoin P2P network	**1**	No specific mentioning of any dataset.	"Eavesdropper adversary" enables linking from IP-adress to Bitcoin pseudonyms.	The main point of our paper is to show that Bitcoin's flooding protocols have poor anonymity properties, and the community's shift from trickle spreading (pre-2015) to diffusion spreading (post-2015) did not help the situation.	2017
Evaluating user privacy in Bitcoin	**2**	Data is taken from the first 140,000 blocks.			

Figure A.2: Preliminary academic information retrieval, part 2

Research Perspectives and Challenges for Bitcoin and Cryptocurrencies	**2**	Does not involve data	Does not involve machine learning	The paper includes 125 references and is a research on Bitcoin's properties and futures stability. It identifies three key components of Bitcoin's design that can be decoupled. It surveys anonymity issues in Bitcoin and provide an evaluation framework for analyzing a variety of privacy-enhancing proposals.	
User Categorization and Community Detection in Bitcoin Network	**2**	http://www.vo.elte.hu/bitcoin/downloads.htm Union-Find-algo to combine addresses.	K-Means, Node2Vector and Fiedler Vector	Bitcoin user categorization and community detection. Uses k_means to separate users into 5 different clusters: trading center, miner, investor, accumulators and merchants.	

Figure A.3: Preliminary academic information retrieval, part 3

Research of feature extraction best practices

Title	Data type	Data source	Features
Unsupervised Learning of Bitcoin Transaction Data	Transaction record	Raw data from the Blockchain, tag data from Blockchain.info and table of historic BTC/USD conversion rates (with time granularity of one day)	• Total USD sent • Total USD received • Highest USD value transaction • Total number of transactions as sender • Total number of transactions as recipient • Timestamp of first transaction • Timestamp of most recent transaction • Average timestamp of transactions weighted by USD • USD value of BTC still possessed by user • Ratio of average value in USD of incoming transaction/outgoing transaction • Number of transaction by source with gambling tagged site • Number of transactions by destination user with gambling tagged sites
Anomaly Detection in Bitcoin Network Using Unsupervised Learning Methods	Network based	Two graphs generated by the Bitcoin transaction network: one graph has users as nodes, and the other has transactions as nodes.	• In-degree • Out-degree • Unique in-degree • Unique out-degree • Clustering coefficient • Average in-transactions • Average out-transaction • Average time interval between in-transactions • Average time interval between out-transactions • Balance • Creation date • Active duration
Unsupervised learning for robust Bitcoin fraud detection	Network based	Bitcoin dataset housed by the Laboratory for Computational Biology at the University of Illinois. All transactions from the genesis block to blockchain 230686 dated 7 April 2013. In between the users are 37 450 461 edges (transactions) that link interactions among users.	Currency features: • Total amount sent • Total amount received • Average amount sent • Average amount received • Standard deviation received • Standard deviation sent Network features: • In degree • Out degree • Clustering coefficient • Number of triangles Average neighbourhood (source target) whereby with reference to each query node: source refers to origin on incoming transaction and target is the destination. The four features identified: in-in, in-out, out-out, out-in.
A Multifaceted Approach to Bitcoin Fraud Detection: Global and Local Outliers	Network based	Bitcoin dataset housed by the Laboratory for Computational Biology at the University of Illinois. All transactions from the genesis block to blockchain 230686 dated 7 April 2013. In between the users are 37 450 461 edges (transactions) that link interactions among users.	Currency features: • Total amount sent • Total amount received • Average amount sent • Average amount received • Standard deviation received • Standard deviation sent Network features: • In degree • Out degree • Clustering coefficient • Number of triangles Average neighbourhood (source target) whereby with reference to each query node: source refers to origin on incoming transaction and target is the destination. The four features identified: in-in, in-out, out-out, out-in.
Unsupervised Approaches to Detecting Anomalous Behavior in the Bitcoin Transaction Network	Network based	The bitcoin dataset was, generously made available by Ivan Brugere. It contains information on all transactions up through May of 2013, and provides	• The hub's degree (total # of transactions) • In-degree (total # of receiving transactions • Out-degree (total # of sending transactions • The means of the childens' total degree • The variance of the childrens' total degree • In-degree • Out-degree

Figure A.4: Research of feature extraction best practices, part 1

		a nice relational structure. The relational schema is detailed in figure 1.	• The mean of all the transactions amounts • The variance of all the transactions amounts • Incoming transactions amounts • Outgoing transactions amounts
Anomaly Detection in the Bitcoin System - A Network Perspective	Network based	We use the Bitcoin transaction data set obtained from Stanford Network Analysis Project. All Bitcoin transactions are documented in a public ledger and are in the currency unit called the Bitcoin (BTC).	User graph extraction: • 13 stk. tjek mendeley. Alle features står forklaret. Transaction graph extraction: • 13 stk. tjek mendeley. Alle features står forklaret.
Analysis of Bitcoin Network Dataset for Fraud	Network based	We have used publicly available data from [10]. This dataset consists of trans- action information of the Bitcoin Network until July 13, 2011. Table 1 shows brief statistics of the dataset that we have used. The dataset consists of two files: vertices and edges.	• User in degree: represents the number of incoming transactions to a user • User out degree: represents the number of outgoing transactions from a user • User unique in degree: represents the number of unique users from which the user has received money • User unique out degree: represents the number of unique users to which a user sent money to. • User mean in: represents the mean value of bitcoin received by a user • User mean out: represents the mean value of bitcoins sent by a user • User in standard deviation: represents the standard deviation of bitcoins received by a user • User out standard deviation: represents the standard deviation of bitcoins sent by a user • Number of in-transactions per unique in-transaction timestamp • Number of out- transactions per unique out-transaction timestamp • In-transaction rate • Out-transaction rate • Balance • Total known user keys • Average in-velocity • Average out-velocity • In-velocity standard deviation • Out-velocity standard deviation • Average in-acceleration • Average out-acceleration • Average neighbor out velocity

Figure A.5: esearch of feature extraction best practices, part 2

ER-Diagrams of data-processing

Processing of raw data

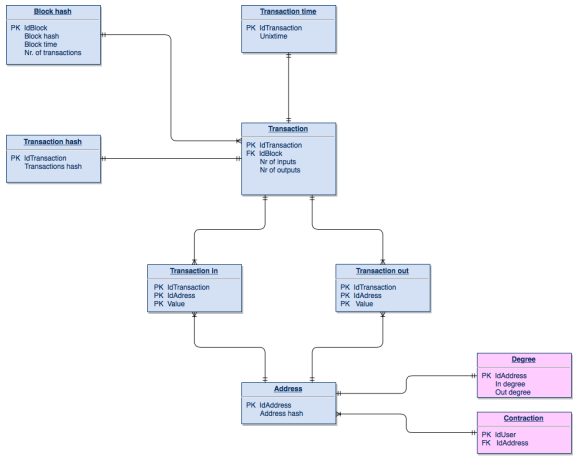


Figure A.6: ER-Diagram of the raw data

Data processing

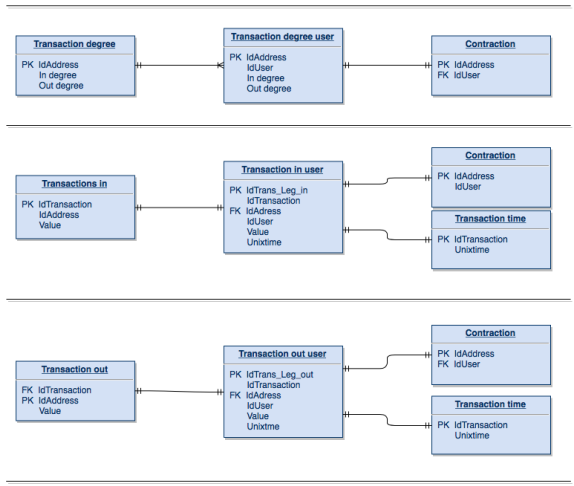


Figure A.7: Illustration of the data processing

Users latent positions in 3D

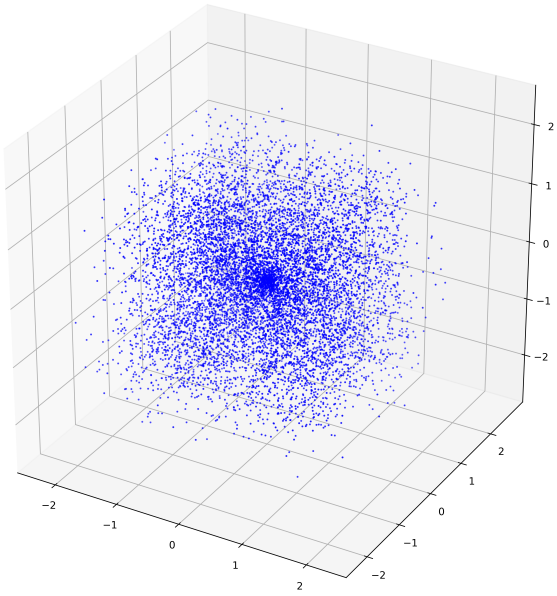


Figure A.8: Plot of all 9968 users latent positions

Baseline-model AUC plot

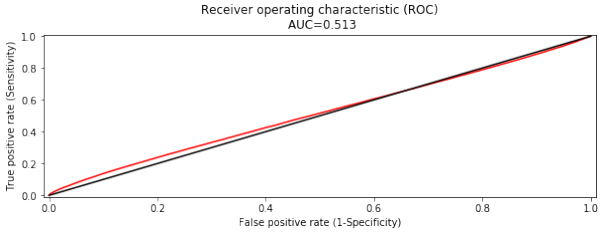


Figure A.9: Baseline ROC

Models in Edward

Baseline-model implementation

```
N = x_train.shape[0]  # number of data points

b = Normal(loc = tf.zeros(1), scale = tf.ones(1))

x = Bernoulli(logits = b*tf.ones([N,N]))
```

Listing A.1: Baseline model implemented in Edward

Latent space model

```
#Model
N = x_train.shape[0]  # number of data points
K = 30  # latent dimensionality

scale_z = InverseGamma(tf.ones([1,K])*1e-3, tf.ones([1,K])*1e-3)
scale_b = InverseGamma([1e-3], [1e-3]) #Gelman 2006

z1 = Normal(loc = tf.zeros([N,K]), scale=(tf.sqrt(scale_z)*tf.ones
    ([N, K])))
z2 = Normal(loc = tf.zeros([N,K]), scale=(tf.sqrt(scale_z)*tf.ones
    ([N, K])))
b = Normal(loc = tf.zeros(1), scale=(tf.sqrt(scale_b)*tf.ones(1)))

pi1 = tf.tile(tf.reduce_sum(tf.pow(z1, 2), 1, keepdims=True), [1,N
    ])
pi2 = tf.tile(tf.reduce_sum(tf.pow(z2, 2), 1, keepdims=True), [1,N
    ])
pi = pi1 + tf.transpose(pi2) - 2 * tf.matmul(z1, z2, transpose_b=
    True)

pi = -tf.sqrt(pi + tf.diag(tf.zeros(N) + 1e3))

x = Bernoulli(probs=tf.sigmoid(pi + b))face

#Inference
qz1 = Normal(loc=tf.get_variable("qz1/loc", [N, K]),
             scale=tf.nn.softplus(tf.get_variable("qz1/scale", [N,
    K])))
qz2 = Normal(loc=tf.get_variable("qz2/loc", [N, K]),
             scale=tf.nn.softplus(tf.get_variable("qz2/scale", [N,
    K])))
qb = Normal(loc=tf.get_variable("qb/loc", 1),
             scale=tf.nn.softplus(tf.get_variable("qb/scale", 1)))

qscale_z = Normal(loc=tf.get_variable("qscale_z/loc", [1, K]),
             scale=tf.nn.softplus(tf.get_variable("qscale_z/scale",
     [1, K])))

```

```
qscale_b = Normal(loc=tf.get_variable("qscale_b/loc", 1),
             scale=tf.nn.softplus(tf.get_variable("qscale_b/scale",
     1)))

inference = ed.KLqp({z1: qz1,
                     z2: qz2,
                     scale_z: qscale_z,
                     scale_b: qscale_b,
                     b: qb},
                     data={x: x_train})
inference.initialize(n_iter=20,logdir='log')

tf.global_variables_initializer().run()

info_loss = np.zeros(20)


for _ in range(inference.n_iter):
  info_dict = inference.update()
  inference.print_progress(info_dict)
  info_loss[_] = info_dict['loss']

inference.finalize()8
```

Listing A.2: latent space model implemented in Edward

Stochastic block model

```

#Model
N = x_train.shape[0]  # number of vertices
K = 10  # number of clusters
gamma = Dirichlet(concentration=tf.ones([K]))
Pi = Beta(concentration0=tf.ones([K, K]), concentration1=tf.ones([K
    , K]))
Z = Multinomial(total_count=1.0, probs=gamma, sample_shape=N)
Z1 = tf.matmul(Z, tf.matmul(Pi, tf.transpose(Z)))
Z1 = Z1-tf.multiply(Z1,tf.diag(tf.ones(N))) + 1e-12*tf.diag(tf.ones
    (N))
X = Bernoulli(probs = Z1)

#Inference
qgamma = PointMass(tf.nn.softmax(tf.get_variable("qgamma/params", [
    K])))
qPi = PointMass(tf.nn.sigmoid(tf.get_variable("qPi/params", [K, K])
    ))
qZ = PointMass(tf.nn.softmax(tf.get_variable("qZ/params", [N, K])))

inference = ed.MAP({gamma: qgamma, Pi: qPi, Z: qZ}, data={X:
    x_train})

inference.initialize(n_iter=200)
tf.global_variables_initializer().run()
info_loss = np.zeros(200)
```

```
22
23 for _ in range(inference.n_iter):
24   info_dict = inference.update()
25   inference.print_progress(info_dict)
26   info_loss[_] = info_dict['loss']
27 inference.finalize()
```

Listing A.3: Stochastic block model implemented in Edward

Gaussian mixture model

```
1 #Model:
2 pi = Dirichlet(tf.ones(K)) #Defining priors
3 mu = Normal(tf.zeros(D), tf.ones(D), sample_shape=K)
4 sigmasq = InverseGamma(tf.ones(D), tf.ones(D), sample_shape=K)
5
6 x = ParamMixture(pi, {'loc': mu, 'scale_diag': tf.sqrt(sigmasq)},
7                   MultivariateNormalDiag,
8                   sample_shape=N)
9 z = x.cat #z is now defined as the component prescribed to the
      observed variable x.
10
11 #Inference: #a conclusion reached on the basis of evidence and
      reasoning
12 T = 800 #number of MCMC samples
13 qpi = Empirical(
14     tf.get_variable(
15     "qpi/params", [T, K],
16     initializer=tf.constant_initializer(1.0 / K)))
17
18 qmu = Empirical(tf.get_variable(
19     "qmu/params", [T, K, D],
20     initializer=tf.zeros_initializer()))
21
22 qsigmasq = Empirical(tf.get_variable(
23     "qsigmasq/params", [T, K, D],
24     initializer=tf.ones_initializer()))
25
26 qz = Empirical(tf.get_variable(
27     "qz/params", [T, N],
28     initializer=tf.zeros_initializer(),
29     dtype=tf.int32))
30
31 #Running Gibbs Sampling:
32 inference = ed.Gibbs({pi: qpi, mu: qmu, sigmasq: qsigmasq, z: qz},
33                      data={x: x_train})
34 inference.initialize()
35 sess = ed.get_session()
36 tf.global_variables_initializer().run()
37
38 t_ph = tf.placeholder(tf.int32,[]) # Empty placeholder
39 running_cluster_means = tf.reduce_mean(qmu.params[:t_ph], 0)
40
41 for _ in range(inference.n_iter):
```

```
  info_dict = inference.update()
  inference.print_progress(info_dict)
  t = info_dict['t']
  if t % inference.n_print == 0:
    print("\nInferred cluster means:")
    print(sess.run(running_cluster_means, {t_ph: t - 1}))

pi_sample = qpi.sample(100)
mu_sample = qmu.sample(100)
sigmasq_sample = qsigmasq.sample(100)
x_post = Normal(loc=tf.ones([N, 1, 1, 1]) * mu_sample,
                scale=tf.ones([N, 1, 1, 1]) * tf.sqrt(
    sigmasq_sample))
x_broadcasted = tf.tile(tf.reshape(x_train, [N, 1, 1, D]), [1, 100,
     K, 1])
x_broadcasted = tf.cast(x_broadcasted, tf.float32)

#Log_liks:
log_liks = x_post.log_prob(x_broadcasted)
log_liks = tf.reduce_sum(log_liks, 3)
log_liks = tf.add(log_liks,tf.log(pi_sample))
log_liks = tf.subtract(tf.reduce_logsumexp(log_liks, 1),tf.log
    (100.0))
log_dens=tf.reduce_logsumexp(log_liks,1)
```

Listing A.4: Gaussian mixture model implemented in Edward

Anomaly detection

nr_pseudonyms	avg_pos_period	mean_value_in	nr_transactions	mean_value_out	avg_out_degree	avg_in_degree	active_duration	dens	cluster
4,04E+02	3,04E+15	1,63E+16	1,76E+01	1,63E+16	1,73E+16	1,53E+16	757710.0	-7,49E+06	0,00E+00
1,00E+00	8727752.0	2,23E+16	1,75E+01	2,44E+15	1,93E+01	8,33E-02	8727752.0	-3,11E+16	0,00E+00
1,00E+00	13641454.0	3,54E+15	1,91E+01	3,54E+15	2,00E+00	5,42E-01	13641454.0	-3,03E+07	0,00E+00
1,00E+00	26969175.0	4,33E+15	1,70E+01	4,34E+15	1,27E+01	2,08E-01	26969175.0	-2,98E+16	0,00E+00
1,00E+00	23921619.0	1,04E+16	2,58E+01	1,04E+16	1,63E+00	4,58E-01	23921619.0	-2,98E+07	0,00E+00
4,10E+01	1,15E+16	5,64E+15	1,69E+01	5,64E+15	1,64E+14	1,64E+16	37451600.0	-2,88E+07	8,00E+00
3,00E+00	3,31E+16	2,78E+16	1,75E+01	2,78E+16	1,38E+16	2,08E-01	33710103.0	-2,82E+07	0,00E+00
4,56E+02	2,85E+16	1,10E+16	1,90E+01	1,10E+16	1,03E+16	8,33E-02	455037.0	-2,68E+07	0,00E+00
1,00E+00	11682349.0	3,17E+16	1,69E+01	3,17E+16	8,25E+00	1,67E-01	11682349.0	-2,67E+07	0,00E+00
1,00E+00	18298561.0	4,46E+15	1,69E+01	2,99E+16	1,63E+01	3,75E-01	18298561.0	-2,61E+07	0,00E+00

Figure A.10: Top 10 most abnormal users

Bibliography

[1] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. *Www.Bitcoin.Org*, page 9, 2008.

[2] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A Kroll, and Edward W Felten. Research Perspectives and Challenges for Bitcoin and Cryptocurrencies. *IEEE Symposium on Security and Privacy*, pages 104–121, 2015.

[3] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. A fistful of Bitcoins. *Communications of the ACM*, 59(4):86–93, 2016.

[4] Bitcoin.info. Bitcoin.info Homepage. *Blockchain Luxembourg S.A.R.L*, 2018.

[5] The Economist. The Trust Machine. *The Economist*, 2015.

[6] Forbes Magazine. The Craziest Bubble Ever. *The Economist*, 2017.

[7] Federal Bureau of Investigation. Bitcoin Virtual Currency: Unique Features Present Distinct Challenges for Deterring Illicit Activity. *Intelligence Assessment*, 2012.

[8] Fergal Reid and Martin Harrigan. An analysis of anonymity in the bitcoin system. *Security and Privacy in Social Networks*, pages 197–223, 2013.

[9] Stefan Poikonen. Unsupervised Learning of Bitcoin Transaction Data. pages 1–16, 2014.

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

[11] Patrick Monamo, Vukosi Marivate, and Bheki Twala. Unsupervised learning for robust Bitcoin fraud detection. *2016 Information Security for South Africa (ISSA)*, pages 129–134, 2016.

[12] Jason Hirshman and Stephen Macke. Unsupervised Approaches to Detecting Anomalous Behavior in the Bitcoin Transaction Network. 2008.

[13] David Liben-Nowell and Jon Kleinberg. The Link Prediction Problem for Social Networks. *Proceedings of the Twelfth Annual ACM International Conference on Information and Knowledge Management (CIKM)*, (November 2003):556–559, 2003.

[14] Adrian E. Raftery Peter D. Hoff and Mark S. Handcock. Latent Space Approaches to Social Network Analysis. *American Statistical Association*, 2002.

[15] Hyun Song Shin and Isabel Schnabel. Money and trust: lessons from the 1620s for money in the digital age. page 3, 2018.

[16] The Bitcoin Core. The Bitcoin Blockchain. 2018.

[17] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. 1991.

[18] Christopher M. Bishop. Pattern Recognition and Machine Learning. *Springer*, 2006.

[19] Tue Herlau, Mikkel Schmidt, and Morten Mørup. Introduction to Machine Learning and Data Mining. 17:1–319, 2016.

[20] David M. Blei. Build, Compute, Critique, Repeat: Data Analysis with Latent Variable Models. *Annual Review of Statistics and Its Application*, 1(1):203–232, 2014.

[21] Michael Jordan, Zoubin Ghahramini, Tommi Jaakkola, and Lawrence Saul. An Introduction to Variational Methods for Graphical Models. pages 186–188, 1999.

[22] Robert Bassett and Julio Deride. Maximum a posteriori estimators as a limit of bayes estimators. pages 1–16, 2018.

[23] Likas Aristidis C. Tzikas, Dimitris G. and Nikolaos P. Galatsanos. The variational approximation for bayesian inference. *IEEE Signal Processing Magazine*, pages 1–3, 2008.

[24] Charles Fox and Stephen Roberts. A Tutorial on Variational Bayesian Inference.

[25] Matt Hoffman, David Blei, Chong Wang, and John Paisley. Stochastic Variational Inference. 2013.

[26] David: (https://www.cs.princeton.edu/courses/archive/fall11/cos597C/lectures/variational-inference-i.pdf) Blei. Variational Inference.

[27] Philip Resnik and Eric Hardisty. Gibbs sampling for the uninitiated. 2010.

[28] Dustin: https://theclevermachine.wordpress.com/2012/11/05/mcmc-the-gibbs-sampler/ Stansbury. Mcmc: The gibbs sampler, 2012.

[29] Arjen de Vries Thijs Westerveld and Franciska de Jong. Generative Probabilistic Models. *University of Twente*, 2012.

[30] Maja Rudolph. Mixture Models. *Edward*, 2017.

[31] OpenAI: https://blog.openai.com/generative models/. Generative models.

[32] Hal Daumé III. A Course in Machine Learning. 2012.

[33] D. Yu and L. Deng. Automatic Speech Recognition. *Springer*, 2015.

[34] Andrew Gelman. Prior Distribution for Variance Parameters in Hierarchical Models. *Bayesian Analysis*, 1(3):515–533, 2006.

[35] Kathryn Blackmond Laskey Paul W. Holland and Samuel Leinhardt. Stochastic Blockmodels: First steps. *Carnegie- Mellon University*, 1983.

[36] Ismael Lemhadri and Youssouf (https://www.youtube.com/watch?v=Trq-1h-s4l0) Emin. Presentation: Community recovery in the stochastic block-model.

[37] Aaron Clauset. Network Analysis and Modeling, Lecture 16. *CSCI 5352*, 2013.

[38] Emmanuel Abbe. Community Detection and Stochastic Block Models: Recent Developments. *Princeton University*, 2017.

[39] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. 2016.

[40] Dustin Tran, Alp Kucukelbir, Adji B. Dieng, Maja Rudolph, Dawen Liang, and David M. Blei. Edward: A library for probabilistic modeling, inference, and criticism. *arXiv preprint arXiv:1610.09787*, 2017.

[41] Edward: (http://edwardlib.org/api/). Box's loop.

[42] Aaron Clauset, Cristopher Moore, and Mark E J Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, 2008.

[43] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. 2009.

[44] Lawrence Hubert. Comparing Partitions. 1985.