Predicting Bitcoin

A Robust Model for Predicting Bitcoin Price Directions Based on Network Influencers

Jonathan Gillett

Submitted in partial fulfillment of the requirements for the degree of Master of Applied Science

to the

Faculty of Engineering and Applied Science
University of Ontario Institute of Technology

October 2016

The root problem with conventional currency is all the trust that's required to make it work. The central bank must be trusted not to debase the currency, but the history of fiat currencies is full of breaches of that trust.

— Satoshi Nakamoto

Dedicated to Odette Gaudreault, your dedication to teaching and belief in education has always inspired me.

1941 – 2014

ABSTRACT

The ability to predict financial markets has tremendous potential to limit exposure to risk and provide better assurances of annualized gains. In this thesis, a model for predicting the future daily price of Bitcoin is proposed and evaluated in comparison to that of a purely random model. Bitcoin is a novel digital currency that relies on cryptography instead of a central authority to verify transactions. Without a central authority, Bitcoin requires a complete list of all transactions to be made public so that they can be verified by all users. This unique feature of Bitcoin, where all transactions are public, is exploited by the model to predict the future price directions based on the actions of Bitcoin users. The daily activity of the markets, aggregate network features, and the actions of major network influencers are all used as features for the predictive model. Where major network influencers are defined as users that accumulate a disproportionate amount of wealth within the Bitcoin network compared to others. The information about the actions of all Bitcoin users are extracted from the blockchain and stored in a relational database for ease of use. Two metrics were created to identify the major network influencers based on the history of their actions recorded on the blockchain. The first metric, based on the concept of an h-index, often used in academia to rank authors by their citations, is used to rank users by the amount of wealth they accumulate monthly. The second metric is based on the optimization of multiple objectives, the maximum increase in wealth with the least amount of activity using Pareto optimization. All of the major network influencers identified were then used as features, in combination with aggregate network features, and market data, to test and evaluate several predictive models. The models created were based on non-linear equations, support vector machines, decision trees, and XGBoost; all evaluated and compared using the same data. The XGBoost model consistently proved to be much more accurate than all other models and was used for the final set of experiments. The XGBoost model was compared to that of a purely random Monte Carlo model using the entire history of data for the period of 2013–2016. The first set of experiments were conducted using various sizes of training and testing data, in each case the XGBoost model had an accuracy 20% greater than that of the Monte Carlo model. For the final experiments, the model was tested in a realistic scenario, predicting the price direction for each future day, while also being re-trained using the results of each new day. The XGBoost model achieved a much better performance in comparison to the Monte Carlo model, which had approximately 50% accuracy, whereas the XGBoost model had 70%–79% accuracy.

We have seen that computer programming is an art,
because it applies accumulated knowledge to the world,
because it requires skill and ingenuity, and especially
because it produces objects of beauty.

— Donald E. Knuth [1]

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my supervisors Dr. Shahryar Rahnamayan and Dr. Masoud Makrehchi for their constant support and advice throughout my entire tenure at UOIT. They have both been amazing teachers and mentors and have nurtured my academic development, without their support I would not be where I am today. I will always appreciate their patience with me and constructive feedback, as well as their unique insights and solutions to the challenges that we had to overcome during the research process.

My sincere appreciation goes to Wesley Taylor, a close colleague and dear friend. Thank you for your consistent support and encouragement, as well as your help, even when I didn't ask. I also greatly appreciate your assistance with numerous LaTeX formatting issues as well as insightful stylistic critique and feedback. I would also like to personally thank Michele "Miki" Spagnuolo for taking the time out of his busy work schedule to have a call with me to clarify questions I had regarding the pre-processing steps in his paper, as well as the address clustering technique that he implemented.

Finally I would like to thank my parents for supporting me throughout my education and making many sacrifices to ensure that I could pursue higher learning and achieve my dreams. I would like to dedicate this thesis to my grandmother Odette, a teacher and educator, who always instilled in me a yearning for knowledge and to seek out and discover the world around me. My grandmother Odette meant the world to me and always dreamt of this day but did not have a chance to share in my joy.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

API Application Programming Interface

AUC Area Under Curve

AUROC Area Under Receiver Operating Characteristic

CART Classification And Regression Tree

CNY Chinese Yuan

DJIA Dow Jones Industrial Average

EUR Euro

FPR False Positive Rate

FX Foreign Exchange

GLM Generalized Linear Model

LHS Left Hand Side

NYSE New York Stock Exchange

OHLC Open High Low Close

OLAP Online Analytical Processing

P2P Peer-to-Peer

PPV Positive Predictive Value

PRNG Pseudo-Random Number Generator

RDBMS Relational Database Management System

RHS Right Hand Side

ROC Receiver Operating Characteristic

RSS Residual Sum of Squares

SQL Structured Query Language

SVM Support Vector Machine

TPR True Positive Rate

USD United States Dollar

Chapter I

INTRODUCTION

1

INTRODUCTION

Bitcoin is a new and emerging digital currency that has revolutionized the way we think about money. It is continuing to increase in adoption and as it gains more widespread appeal; it is also attaining substantial real-world applications and value. From its initial creation in 2008 to our present time of writing in 2016, Bitcoin has gone from complete obscurity as a new digital currency to one that has tens of millions of users, an average value of approximately $330 per bitcoin, and real financial markets where bitcoins are exchange for fiat currencies totalling billions of dollars monthly. With all of the attention that Bitcoin has been receiving, in addition to its increasing real-world applications as a medium of financial exchange, the possibility of understanding the market and even predicting the future market direction of the technology is alluring. Furthermore, Bitcoin is unique in that all of the transactions that take place on the network are recorded publicly, as such it makes it possible to track the precise activities of every user (anonymously), but uniquely based on their addresses on the network.

With the increased adoption of Bitcoin and the availability of numerous financial services where bitcoins can be exchanged for fiat currencies, we see an ideal opportunity for the creation of a predictive model that can be used to predict the future price direction of the market. Given the high value of each bitcoin and the public availability of every transaction made on the network, we have a level of insight into the activities that are related to changes in price not available in other financial markets such as the stock market. With all of the information pertaining to transactions publicly accessible, we wish to use this knowledge to our advantage to identify the major users within the network that influence the market, and base our predictions of the future price direction on their actions.

1.1 INTRODUCTION TO BITCOIN

Bitcoin is an innovative technology that has revolutionized the way we think about money. While the concept of a digital currency is not unique, the application of public key cryptography to make a digital *cryptocurrency* was first introduced by Nakamoto in *Bitcoin: A Peer-to-Peer Electronic Cash System*, a self-published paper by the pseudonymous creator of Bitcoin, whose true identity has never been confirmed. In the paper, Nakamoto describes a new form of electronic currency called *Bitcoin*, where the fundamental features of this unique digital currency are its use of public key cryptography, a ledger of all transactions known as a *blockchain*, and the ability for a decentralized authority to verify transactions through a process known as *mining*. The most defining innovation of Bitcoin is its ability to prevent *double-spending*, whereby a user can spend the same amount twice, without the need for a centralized authority to verify all transactions.

Double-spending attacks are the most difficult challenge of any electronic monetary system, whereby an attacker sends funds to two separate entities simultaneously, and each party receives the funds even though the attacker only has enough to pay one of them [3, 4]. This is analogous to writing two cheques anonymously, where both recipients attempt to cash the cheques, and only the first to cash the cheque is paid. In systems, such as Bitcoin, where all users remain anonymous without their real identities attached to transactions, this becomes a serious threat that is challenging to resolve without centralization. Bitcoin manages to solve this issue without any centralization through the use of a *blockchain*, a public ledger that records all transactions so that for any transaction, the balance of a user can be determined by looking at the entire history of all prior transactions. However, as the blockchain is a public ledger where all transactions broadcast on the network are added, there is still the issue of maintaining the integrity of the transactions recorded on the ledger. Bitcoin introduces the concept of *mining*, whereby users validate the integrity of all transactions before adding them to the blockchain; with a validation signature that is computed using an extremely computationally expensive process known as *mining*. Lastly, after validating the

integrity of all of the transactions and adding them to the blockchain, the user that performed the mining, often referred to as a *miner*, is rewarded with the currency (BTC) for their efforts, this is the only way that currency is created.

The blockchain acts as a public ledger of all transactions and replaces the need for centralization through the process of *mining*. Bitcoin allows any user in the network to participate in *mining*, which validates the transactions on the blockchain, allowing the network to operate without any centralization. The network is often referred to as operating in a *decentralized* manner, where all of the actions between users within the network function similar to a Peer-to-Peer (P2P) network, with the exception that there is no centralization required. All of the transactions that take place are broadcast from each peer within the network to all other peers that they are connected to, and further replicated until the transaction has spread across the entire network. Furthermore, the blockchain does not require any centralization, and a copy of it is replicated across every peer on the network that operates as a *full node*, storing all the records.

1.2 MARKET OPPORTUNITY

Bitcoin provides many new and unique opportunities, not only as a new revolution in digital finance, but also in terms of the intrinsic and often volatile value it has. With each passing year, we see the amount of transactions recorded on the blockchain increasing and a corresponding increase in the amount of volume of BTC being transacted between users and with Bitcoin related services. This demonstrates a healthy market that is growing rather than shrinking, and poses the possibility for future growth as adoption increases with each succeeding year. We see in Figure 1 a clear demonstration of the increased adoption of Bitcoin, as well as the growing number of transactions and corresponding volume of BTC transacted. The line in blue shows the daily volume of BTC transacted between users within the network, which we see increasing dramatically each year reaching a peak just prior to 2013. This period, just prior to 2013, is when Bitcoin began to dramatically increase in value, making each bitcoin much

more valuable. For the line in red, we see a continually increasing trend in the number of transactions, dramatically increasing in 2012 as Bitcoin began to gain real monetary value. We continue to see the number of transactions increasing yearly to its current peak of between 100,000–250,000 transactions daily. The increasing trend in daily volume, even given Bitcoin's dramatic increase in value, demonstrates a market that is growing with increased adoption and presenting future opportunities for economic activity.

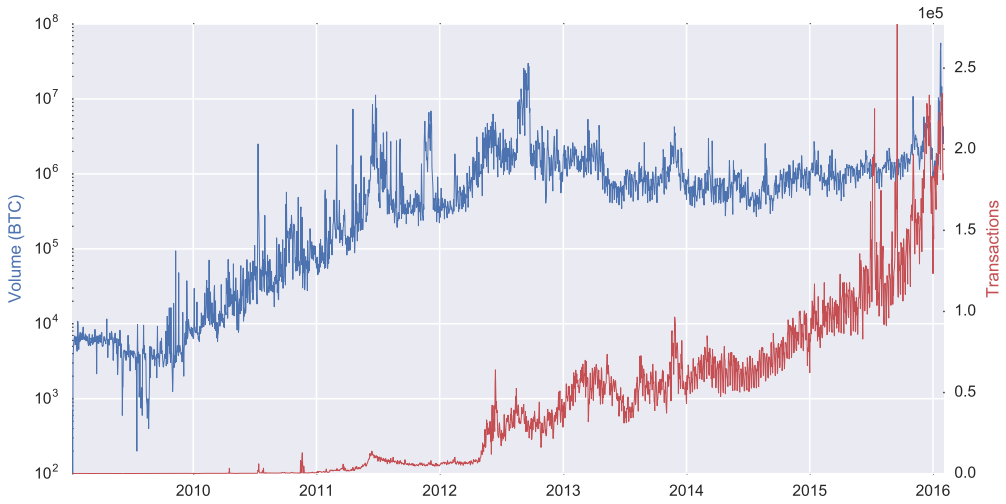


Figure 1: Daily Blockchain Transactions and Volume

With the increase in adoption and blockchain activity, we also see a similar trend starting at the beginning of 2013, in the real-world monetary value of Bitcoin. Prior to the widespread availability of exchanges, where users could exchange their bitcoins for fiat currencies such as United States Dollar (USD), Chinese Yuan (CNY), and Euro (EUR) Bitcoin had very little monetary value. With the availability and more widespread adoption of Bitcoin exchanges, the finite supply of bitcoins available became a catalyst for further speculation of the price. We see this trend in Figure 2, where there is a speculative bubble starting in September of 2013, leading to an inevitable crash where the price adjusted. Historically, there has been a consistent trend over the past several years (2013–2016) of the price fluctuating between $200–$400 USD for a single bitcoin. This a dramatic increase in the value of bitcoins as a medium of exchange, before the widespread availability of exchange services, when bitcoins were practically worthless. For

perspective, the first recorded Bitcoin transactions was in 2010 for two pizzas, worth approximately $20 USD that were purchased for 10,000 BTC, now worth over 3 million USD today [5].

Figure 2: Bitcoin Price in USD Markets

The consistent high value of Bitcoin, despite market fluctuations, gives more confidence in the future support of the market. Furthermore, with the widespread availability of exchanges for USD, CNY, and EUR, we see a healthy diversity in the availability of overall market liquidity. The list of all major exchanges are shown in Table 1, including their currency pairs, total volume traded (BTC), and percentage of the entire market for the period of our research from 2013-01-01 – 2016-01-01. We see a dominant amount of the total volume of BTC traded on exchanges heavily favouring the Chinese markets, with BTC China and OKCoin combined making up 69.25% of the total volume of BTC traded. The USD markets combined make up 28.91% of the total volume of BTC traded, with the EUR markets the smallest with the remaining 1.84%. In addition to the cumulative total volume in BTC for each exchange, we also see in Figure 3 the aggregated cumulative monthly volume in millions of USD for each market. In order to compare each market equally, the daily trading volume of each currency has been normalized to the end of day price in USD using the Foreign Exchange (FX) market data provided by OANDA [6].

Despite the relative size of each market, it is important to also consider that unlike traditional stock markets such as New York Stock Exchange (NYSE) or Nasdaq,

Exchange	Market	Volume (BTC)	Percent of Total
Bitfinex	USD	19,534,023	11.63%
Bitstamp	USD	15,581,657	9.27%
BTC-E	USD	10,515,384	6.26%
Coinbase	USD	2,940,149	1.75%
BTC China	CNY	45,627,787	27.15%
OKCoin	CNY	70,736,870	42.10%
bitcoin.de	EUR	930,244	0.55%
BTC-E	EUR	175,500	0.10%
Kraken	EUR	1,989,286	1.19%

Table 1: Bitcoin Markets

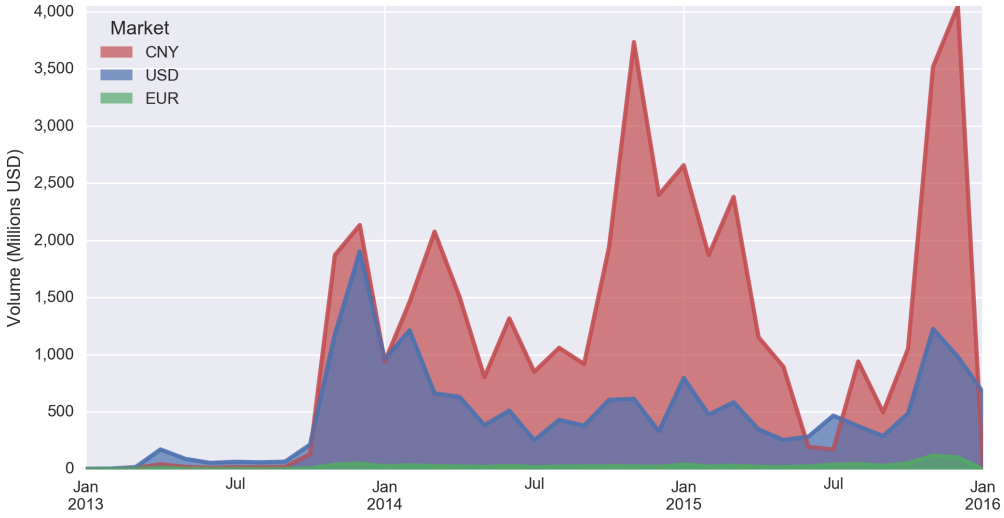


Figure 3: Bitcoin Monthly Volume for USD, CNY, EUR Markets

which operate only for a portion of the day, the Bitcoin markets are all active *24 hours a day*. Not only does each respective market run 24 hours a day, they also run for all 365 days of the year, compared to the typical 252 trading days a year of traditional markets. This provides a large window of opportunity for any trading activity that could be made based on the prediction of a future price direction, as there is no risk of the market being closed as the markets are always active.

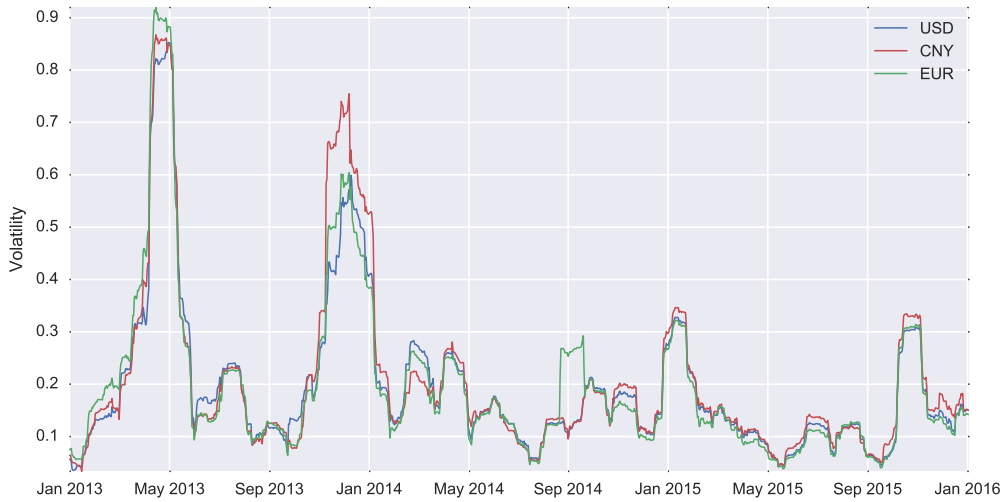


Figure 4: Bitcoin Volatility for USD, CNY, EUR Markets

While the continuous operation of the markets provides many opportunities for trading activities, the Bitcoin markets also display a high tendency for volatility as shown in Figure 4. The volatility is calculated as the rolling standard deviation of the price percentage change using a window of the past month. We see that there are numerous periods of very high volatility, with many periods having a volatility over 0.10 (10%). While volatility can provide opportunities for significant changes in price that can increase returns, it also provides an equal opportunity for risk. For our predictive model to take advantage of the high volatility of the market, it must have an accuracy significantly greater than that of a purely random model. Our model must be able to maximize the earnings from large upwards price changes and minimize losses from large downwards price changes.

1.3 HYPOTHESIS

We establish our hypothesis on the concept of *predicting Bitcoin* based on the actions of *network influencers*. We define *network influencers* as users within the Bitcoin network that have accumulated a disproportionate amount of wealth respective to other users in the network through their actions using exchanges. While there could be *network influencers* based on other criteria, such as the users that

attract the most new users to the network, or interact with the greatest number of users. For our purposes, we explicitly focus our attention on the influencers that have accumulated a significant amount of wealth through their activities on exchanges. Formally, we define our hypothesis as follows:

> A predictive model can be created for the Bitcoin market that can predict the price direction for each future day with better accuracy than pure randomness, based on the historical market data, overall Bitcoin network features, and specifically the activities of the major influencers of the network. Where we define the major network influencers as those that accumulate a disproportionate amount of wealth within the Bitcoin network, compared to other users based on their activities using Bitcoin exchanges.

Given our hypothesis, we must process the Bitcoin blockchain data to extract the features of the network, establish a metric to quantify the influence of users based on their wealth accumulation by using exchanges, and lastly develop a predictive model that demonstrates better accuracy than pure randomness.

1.4 OBJECTIVES

Based on our hypothesis we established several research objectives that are met as part of our research process described in our methodology in Chapter 3. Each of the objectives is important to the creation of a predictive model and the establishment of our experimental setup and evaluation. We have broken down our objectives into three specific items: defining what we commonly refer to as *major network influencers*; creating a metric to describe their influence and identifying the network influencers; and lastly creating a model to predict future price directions based on their actions.

1.4.1 *What is a Major Network Influencer?*

The *major network influencers* are an important part of our research, and as such, we have clearly defined them in our hypothesis and throughout our methodology. This is to ensure that there is no confusion regarding their definition within the context of our research. Formally, we define *major network influencers* within the context of our research as follows:

> A major network influencer is a user within the Bitcoin network that accumulates a disproportionate amount of wealth within the network compared to other users based on their activities using Bitcoin exchanges.

We explicitly define a *major network influencer* as a user that accumulates a disproportionate amount of wealth, implying that they have a vast amount of wealth in comparison to the majority of Bitcoin users. Furthermore, we are explicitly interested in users that acquired their wealth based on their activities using *Bitcoin exchanges*. This criteria is intended to restrict our selection of users to those that accumulate their wealth through the use of Bitcoin exchanges, rather than accumulate it through selling illicit goods, operating gambling websites, or other Bitcoin related services. We are focused on creating a predictive model based on the activities of users recorded on the blockchain that frequently engage in activities that may influence the Bitcoin market. Given this criteria we extract the activities of all *major network influencers* from the blockchain and use these as features for our predictive model.

1.4.2 *Can Major Network Influencers be Identified?*

As part of our hypothesis, we must identify the network influencers within the Bitcoin network based on a metric that can accurately describe their influence. Using this metric we then rank the users within the Bitcoin network so that we can identify users that have a disproportionate amount of wealth relative to all

other users, whom we refer to as the *major network influencers*. The activities of these users that are recorded on the blockchain are then used as features for the creation and evaluation of our predictive model.

The process of identifying network influencers based on our hypothesis and objectives involves identifying users that are predominantly engaged in activities that influence the Bitcoin market. In order to create a metric that optimally ranks and identifies network influencers, we evaluated two metrics in our methodology in Section 3.2. The first metric is based on the concept of calculating the most accurate representation of the historical activity of the user. The second metric is based on selecting the users that have the optimal outcome of three objectives: maximum increase in wealth, minimum transactions, and minimum history of activity. After applying both metrics to the Bitcoin users, we then choose the best metric to use in Section 3.2.3, based on the descriptive statistics of the *major network influencers* identified by each metric.

1.4.3 *Can an Accurate Predictive Model be Created?*

Our final research objective is to create a model that can predict the price direction (`UP/DOWN`) of the Bitcoin market for the following day with greater accuracy than that of a purely random model. As part of testing our hypothesis, the predictive model will be trained and evaluated based on the features extracted from the historical Bitcoin market data, aggregate blockchain activity, and most importantly the activities of major network influencers. We outline our process of extracting the features in Section 3.3.1 as well as the challenges inherent in the structure of the data. Following the feature engineering, we then evaluate various predictive models in Section 3.3.2, making our final selection for the experiments based on the best performing model out of all evaluated. Lastly, we evaluate our model extensively through a variety of experiments in Chapter 4 to compare its accuracy to that of a purely random model.

1.5 THESIS OUTLINE

What follows is an outline of the thesis describing each of the sections and their importance to testing the research hypothesis. Beginning with our current Chapter 1, we give an introduction to Bitcoin to provide the reader with the necessary background to understand the technology. Following our explanation, we then demonstrate the opportunities for a model to predict the market price direction, given the increasing adoption and activity of the Bitcoin network. The Bitcoin market shows much potential for a predictive model, given the high value of bitcoins, the continuous trading on exchanges, and high volatility of the market. We then define our hypothesis in Section 1.3, which is to create a model that can predict the price direction of the Bitcoin market with better accuracy than a random model. The predictive model is based on the market data, entire network features, and activities of users that accumulate vast amounts of wealth using exchanges.

In the following Chapter 2, we provide a detailed background and literature review of similar research. In the background review in Section 2.1, we provide the reader with the knowledge necessary to understand our methodology and experiments. The background review provides an explanation of the various predictive models used in our research, as well the metrics that are used to evaluate and compare the performance of predictive models. In the literature review in Section 2.2, we describe the body of prior research on analyses of the Bitcoin network based on the public record of all transactions in the blockchain. Following this, we also review prior research pertaining to predicting the Bitcoin market based on sentiment analysis of social media, where sentiment analysis is a metric to rank positive or negative emotions. Lastly, we provide a review of all prior predictive models for the Bitcoin market in Section 2.2.3 and discuss the performance of each, which achieve an accuracy between 50%–55%. The predictive models used in previous research were a basis for the design of several of our models, and their performance was used as a benchmark to refine and evaluate our own models.

For our methodology in Chapter 3, we describe in detail our entire pre-processing stage, which involves extracting the entire history of transactions from the Bitcoin blockchain, and storing them in a database in Section 3.1. Based on the extracted blockchain data, we then describe our method of clustering the addresses using parallelization in Section 3.1.2, and a statistical analysis of the data and wealth inequality in Section 3.1.4. Following the pre-processing, we describe our metrics for identifying the major network influencers in Section 3.2, where we evaluate two metrics. The first metrics is based on an optimal representation of historical activity of the user, and the second based on optimizing several objectives. Based on the best metric chosen to select the major network influencers, we then describe our methodology for creating the predictive model starting with the feature engineering in Section 3.3.1. After feature engineering the data, we then create several models in Section 3.3.2, with the model that has the best accuracy chosen for our experiments based on a variety of tests in Section 3.3.3.

After selecting the best out of all of our predictive models, we then conduct our final experiments in Chapter 4. Where we define our experimental setup in Section 4.1, including the Monte Carlo model based on pure randomness, that our model will be compared against. We then perform our holdout experiments in Section 4.3, to evaluate the predictive performance and robustness of our model when the amount of training data is increased, and the testing data is decreased. Lastly, we evaluate the predictive performance of our model in Section 4.4, where we train the model and use it to predict the price direction of each future day, using the resultant outcome to then re-train the model daily.

Finally, in Section 5.1, we provide our concluding remarks on the results of our experiments and the testing of our hypothesis. We then clearly define the contributions of our research, and explain the benefits of our research work to the academic community. Lastly, we explain the areas of future work that we would like to pursue, and define the limitations of our current research that we would like to expand upon in the future.

Chapter II

BACKGROUND AND LITERATURE REVIEW

2

BACKGROUND AND LITERATURE REVIEW

2.1 BACKGROUND REVIEW

The following section provides a background review of the machine learning concepts that are specific to the research work done. We discuss the three leading predictive models used for our research: Support Vector Machine (SVM), decision trees, and XGBoost. As well, for each predictive model we provide an explanation of their operation, as well as the benefits and drawbacks of each predictive model. Lastly, we provide a detailed summary of the machine learning metrics used to evaluate the performance of our models, including a comparison of the strengths and weaknesses of each for binary classification problems.

2.1.1 *Predictive Models*

Predictive models can be described as classifiers that are used to predict the outcome of future or withheld data based on the known classifications provided during training. A classifier categorizes the data provided into one of $[2, \ldots, N]$ classes based on the features of the data; when the class labels are known and provided, it is referred to as *supervised learning* [7]. The models that we explicitly focus on as part of our research are those of *supervised learning*, as the outcome (`UP/DOWN`) for predicting the price direction are known and provided.

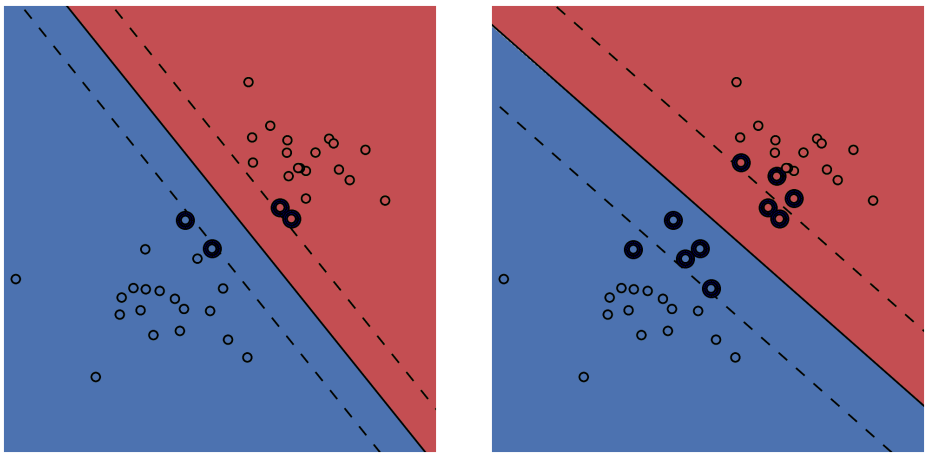
2.1.1.1 *Support Vector Machine*

One of the most widely used predictive models for classification and regression problems is a SVM. The SVM represents the data as a series of points within a space, where the points are separated into two categories using a hyperplane,

which attempts to make a gap between the two classes of points as wide as possible [8]. The concept of a SVM was first introduced in 1964 by Aizerman, Braverman, and Rozoner in "Theoretical foundations of the potential function method in pattern recognition learning" [9], in which they first introduced the concept as a learning machine that could be used to classify data with a very high number of features, similar to the capabilities of the human mind. The main issue with the initial publication of SVM is that it was not robust to non-linear classification problems, and as such was not widely adopted until Boser, Guyon, and Vapnik [10] introduced a kernel trick to transform the data from a non-linear classification rule to that of a linear classification rule. Even with the introduction of a kernel trick, SVM was not widely adopted until the introduction of a *soft margin*, by Cortes and Vapnik [8], in "Support-vector networks," which allowed for SVM to be extended to cases where the data is not linearly separable.

The SVM model works by creating a higher-dimensional model which assigns each new data provided to one category or another. In the case of a linear model, the data separation is achieved by constructing a hyperplane or set of hyperplanes in higher dimensional space. The separation of the data on either side of the hyperplane is the discriminator in assigning the data to a class, a set of two parallel hyperplanes are used to separate the data so that the distance between the two nearest opposing data points is maximized.These data points are known as the support vectors, which establish the hyperplanes for separating the data [8, 11]. In the case of linearly separable data, a hard-margin is used, which defines the support vectors as those that lie directly on the parallel hyperplanes. However, when the data is non-linearly separable, instead a soft-margin is used to establish the margin, this was the innovation introduced by Cortes and Vapnik [8] that allowed SVM to gain more widespread application.

The application of a soft-margin is frequently used due to its robustness and application to non-linearly separable problems. In Figure 5, we see the application of the soft-margin, where in the first figure a very small value of $\lambda = 0.01$ is used making it behave nearly identical to the hard-margin SVM, where the margins are established directly along the support vectors. In the second figure, a



(a) SVM Example C = 0.01 (b) SVM Example C = 1.0

Figure 5: SVM Hyperplane and Margin Example

much larger value of $\lambda = 1.0$ is used, thereby increasing the margin size to establish the support vectors based on the minimization of a *hinge loss* function [7, 8]. The calculation of the soft-margin for SVM involves *minimizing* the summation of a hinge loss function as given in Equation 1. Where n is the number of vectors and λ is the parameter controlling the margin-size, where if a small enough value of λ is chosen, it operates similarly to a hard-margin as shown in Figure 5. For the hinge loss, $\max(0, 1 - y_i(\vec{w} \cdot \vec{x_i} + b))$, it is at zero if the value of the constraint (1) is satisfied when $\vec{x}_i$ is on the correct side of the margin, otherwise the value returned is proportional based on the distance it has from the margin.

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max(0, 1 - y_i(\vec{w} \cdot \vec{x_i} - b))\right] + \lambda\|\vec{w}\|^2 \tag{1}$$

For many classifier applications in machine learning an SVM is often the first model evaluated. They are still effective when the dimensions of the data are greater than available samples, are relatively memory efficient, and have versatility through the use of different kernels when mapping the data to the higher dimensions [7]. However, the overall performance of SVM is highly dependent on the correct choice of kernel and corresponding parameter C, which are specific to certain problems and difficult to generalize [12]. While attempts have been made to better understand the relation between the kernel and regularization pa-

rameters, there is no established theory for choosing the appropriate parameters based on the classification task [12, 13].

2.1.1.2 *Decision Trees*

Another category of predictive models are decision trees, which are a supervised learning method that can be used as a classifier for a predictive model. Decision trees are unique in that they are *non-parametric*, whereby the parameters for the model are not fixed, but instead are determined based on the statistical properties of the data and the amount of training data available [14]. Rather than performing complex higher-dimensional mapping and hyperplane segregation as in SVM, decision trees generate a model dynamically based on the properties and amount of training data. The decision tree creates decision rules that result in an outcome at each level of the tree, where the rules are created based on the properties of the data [14, 15].

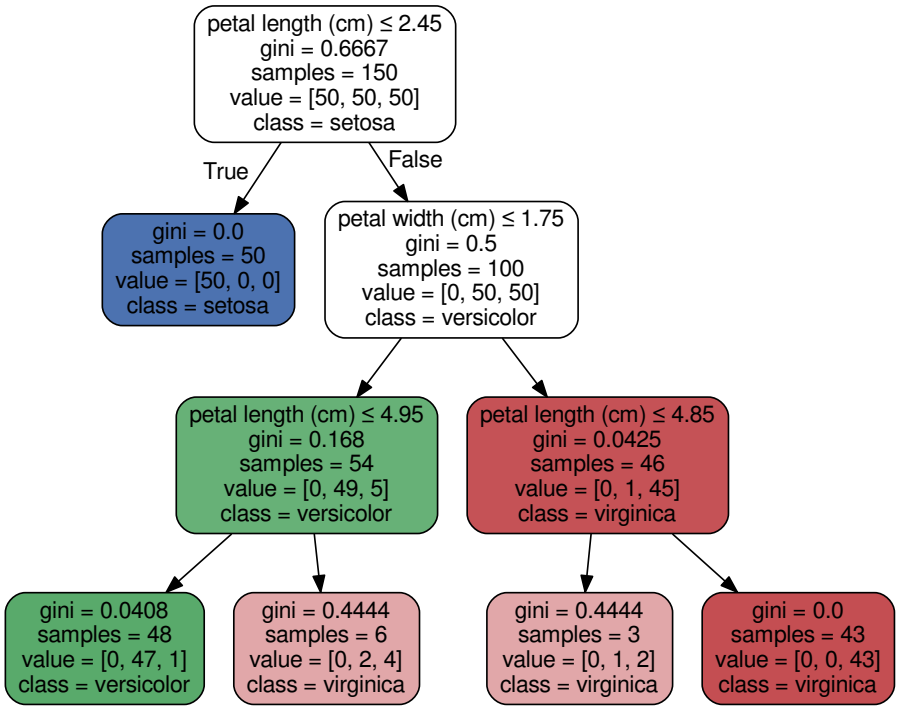


Figure 6: Decision Tree Example

While the concept of a decision tree has been used in many other contexts, the application of decision trees for classification and regression problems in machine learning was first introduced by Breiman et al. [16] in *Classification and regression trees*. The seminal work introduced the concept of decision trees as they are used in machine learning under the umbrella term Classification And Regression Tree (CART). The predictive model for a decision tree is learned by mapping the observations about the data represented in the form of branches where decisions are made. The conclusions about the class of the data, are based on the values represented in the leaves that form for each level of the tree [17]. The algorithms for building the tree operate in a top-down fashion by choosing a variable at each step that best separates the data, until an appropriate height is reached to separate the data into all available classes [17, 18]. An example of this process is shown in Figure 6, where it is used to classify the flowers of the commonly used *Iris Data Set* [19, 20]. At each step of the tree, a comparison is made based on the petal length and width to classify the data. In the first step if petal length (cm) $\leqslant 2.45$ then it is classified as a *setosa* flower, otherwise the process is repeated and the data is further separated into classes by performing additional comparisons.

At each leaf in Figure 6 a value is given called the *Gini impurity*, which is a measurement used by CART for multi-class classifiers to measure the amount of misclassification. The Gini impurity $\mathcal{I}_G$ measures how frequently a randomly chosen element from the set would have an incorrect class when classified at random given the distribution of possible classes in the subset of data at each leaf [15, 17]. It is calculated as the sum of probabilities f_i of class i being chosen multiplied by the probability of a mistake occurring $1 - f_i$, an example is given in Equation 2, where a dataset has N classes, and f_i is the probability of being classified with the class i in the dataset.

$$\mathcal{I}_G(f) = \sum_{i=1}^{N} f_i(1 - f_i) = \sum_{i=1}^{N} (f_i - {f_i}^2) = \sum_{i=1}^{N} f_i - \sum_{i=1}^{N} {f_i}^2 = 1 - \sum_{i=1}^{N} {f_i}^2 \qquad (2)$$

There are many benefits to using decision trees in comparison to other predictive models as they provide a different approach to classification and predictive modelling tasks. One of the main benefits of decision trees is that they produce an explanation of how decisions are made when classifying data, as it is possible to generate a visual representation of the model similar to Figure 6. Furthermore, they can handle both numerical and categorical data and do not require extensive data preparation in comparison to SVM; and are robust even in the case where assumptions are violated by the true model [15]. However, there are also numerous drawbacks to using decision trees, in particular, the task of generating the optimal decision tree is an NP-complete problem and often greedy algorithms are used in selecting the optimal tree, which can lead to biases [15, 18]. Furthermore, decision trees have a tendency to create overly-complex trees, which although explaining the decision process can be very hard to comprehend. The overly-complex trees result in over-fitting as the decisions do not create sufficient generalizations from the data [15]. A technique known as pruning can be used to mitigate the effects of over-fitting, but still does not provide a solution to selecting the optimal tree [21].

2.1.1.3 *Extreme Gradient Boosting*

A new predictive model based on the concept of decision trees, known as Extreme Gradient Boosting or just XGBoost, has found widespread use due to its robustness and strong performance by combining the outcome of multiple trees into a single model. XGBoost is also a supervised learning model, and was created by Chen while doing research on variations of existing tree boosting methods, to satisfy the need for boosted trees with conditional random fields [22]. The predictive model found widespread adoption after it was used to solve the Higgs Boson Challenge held by Kaggle and won first place in the competition. Following the conclusion of the competition, Chen and He proceeded to publish their results and it has since gained widespread recognition [22, 23]. XGBoost has since been implemented and provided as packages for a number of programming lan-

guages including Python, R, Java, etc. and has been used to win a number of leading machine learning competitions [24, 25].

Given the demonstrated strength of XGBoost in solving numerous machine learning competitions, we also chose to consider it in our list of predictive models evaluated for price direction prediction in Section 3.3.2. The XGBoost model is an extension of several concepts including decision trees, in the form of the widely used CART model, in addition to *Gradient Boosting*, whereby a predictive model is generated as an ensemble of the outcome of numerous decision trees [26, 27]. In the CART model used to create the decision tree a score is associated with the classification outcome of each of the leaves, such as the *Gini impurity* shown previously. An ensemble of multiple trees are combined, where the classification outcome given for each of the leaves is summed to generate a final score. In each case, the trees are combined to try and complement each other, while providing robustness to over-fitting [26].

The ensemble model used to combine trees to complement each other can be expressed in Equation 3, where N represents the number of trees in the ensemble and f is a function of $\mathcal{F}$ the set of all possible CARTs. The following objective to optimize based on the model is given in Equation 4, where L refers to the training loss function, which measures the predictive performance of the model on training data, and Ω is the regularization term [26]. The *extreme* aspect of XGBoost comes from the enhancements made to the learning and ensemble methods as well the focus of the library provided to push the computational limits to the *extreme* by a performance-tuned, scalable, and robust model that gives accurate results [26].

$$\hat{y}_i = \sum_{n=1}^{N} f_n(x_i), \quad f_n \in \mathcal{F} \tag{3}$$

$$\text{obj}(\Theta) = \sum_{i=1}^{k} L(y_i, \hat{y}_i) + \sum_{n=1}^{N} \Omega(f_n) \tag{4}$$

One of the main benefits of XGBoost is that it provides a robust solution that is more resilient to over-fitting than decision trees by using an ensemble of trees

which are combined. Furthermore, it has also consistently demonstrated strong performance in machine learning competitions, often receiving first place in comparison to other predictive models used in the competitions [24, 25]. Lastly, the authors have provided implementations for popular programming languages and have emphasized performance and parallel computation, allowing XGBoost to scale beyond billions of examples using fewer resources than existing systems [26]. However, there are still issues with XGBoost, it is still a new model that is in its infancy, with implementations that are very recent and may still contain unexpected bugs or errors, and the final model can still be complex and not necessarily comprehensible.

2.1.2 *Model Evaluation*

In binary classification problems a confusion matrix as shown in Table 2 is commonly used as it provides a summary of the four possible outcomes of any binary classification problem: *TP*, *FN*, *FP*, or *TN* [7, 28]. In the case of *TP*, a *True Positive*, it is the correct number of classifications of positive examples where the actual outcome was also positive, thus correctly predicting a positive outcome when the actual outcome was also positive. Whereas *FN*, a *False Negative*, is the number of incorrect classifications of positive examples, where the predicted outcome was negative when the true outcome was actually positive. In the case of *FP*, a *False Positive*, it is the number of incorrect classifications of negative examples, where the predicted outcome was positive when the actual outcome was in fact negative. Lastly, the *TN*, a *True Negative*, is the number of correctly classified negative examples, where the predicted outcome was negative and the actual outcome was also negative.

Actual Outcome	**Classified Positive**	**Classified Negative**
Positive	TP	FN
Negative	FP	TN

Table 2: Confusion Matrix

A hypothetical *perfect binary classifier*, would have a *TP* with the same number of actual positive (1) outcomes, and a *TN* with the same number of actual negative (0) outcomes, thus both the *FN* and *FP* measures would be 0. In this scenario, it could be said that the binary classifier can *perfectly predict* the actual outcome, as both the *TP* and *TN* would match the number of positive and negative results in the actual outcome.

2.1.2.1 *Precision and Recall*

The origins of the precision and recall metrics frequently used to evaluate the performance of information retrieval and classification systems were first defined in the publication, "Machine Literature Searching VIII. Operational Criteria for Designing Information Retrieval Systems" in 1955 by Kent et al. At the time of publication the metrics were specific to information retrieval contexts, and were defined in terms of a set of retrieved documents (e. g. the list of all documents retrieved that matched a search query). In addition to information retrieval systems, these metrics are now frequently used in machine learning applications [30, 31], in particular binary classification systems as the metrics use the possible outcomes shown in Table 2, which are based on either a positive (1) or negative (0) outcome.

The following equations describe the precision and recall metrics in terms of the context of the confusion matrix, which provide an evaluation of the classification performance of a classifier model. The precision, also be referred to as Positive Predictive Value (PPV), as defined in Equation 5, describes the number of positive examples correctly classified as positive (TP), divided by the total number of available examples that are classified as positive [7, 32]. Recall as defined in Equation 6, describes the number of positive examples that are correctly classified as positive (TP), divided by the total number of actual positive examples [7, 32]. For both the precision and recall metrics the outcome of possible values lie between the range of $[0, 1]$, where the best possible score is represented by 1 and the worst at 0. Intuitively, precision can be interpreted as the ability of the classifier to not incorrectly predict an outcome as positive that is actually nega-

tive, whereas recall is the ability of the classifier to accurately predict all actual positive outcomes.

$$\text{precision} = \frac{tp}{tp + fp} \tag{5}$$

$$\text{recall} = \frac{tp}{tp + fn} \tag{6}$$

2.1.2.2 *F1-Score*

Both the precision and recall metrics evaluate only one aspect of the classifier, as such a third metric is commonly used, known as the F_1 measure or F1-score. This can be interpreted as the combination of both the precision and recall metrics using a harmonic mean, and is frequently used as a measure of a classifier's accuracy [28, 33]. The F1-score is defined in Equation 7 as the harmonic mean of precision and recall, where the precision is multiplied by recall and then divided by the sum of precision and recall. Similarly to the precision and recall metrics, the outcome of possible values lie between the range of $[0, 1]$, where the best possible score is represented by 1 and the worst at 0. However, even with the application of the harmonic mean applied to precision and recall for the F_1 score, all three metrics can be considered biased as they ignore the effect of correctly handling negative examples, and thus can lead to underlying tendencies and biases [28].

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{7}$$

2.1.2.3 *Receiver Operating Characteristic*

As such, in order to address the biased tendencies of precision, recall, and F1-score, an additional metric is often used known as Receiver Operating Characteristic (ROC). The ROC or *ROC curve* as it is also known, is a visualization that has been adopted from signal processing to become a standard for evaluating the performance of binary classifiers [28]. The ROC curve is a two-dimensional

plot with the False Positive Rate (FPR) and True Positive Rate (TPR) plotted on the x and y axes at various threshold settings. Where the TPR is the ratio of *TP* to the total positive outcomes and the FPR is the ratio of *FP* to the total negatives [34, 35]. The benefits of using the ROC metric for applications in machine learning has been highlighted by Flach [36], in particular that it gives a geometric insight into the sensitivity of the classifiers being evaluated.

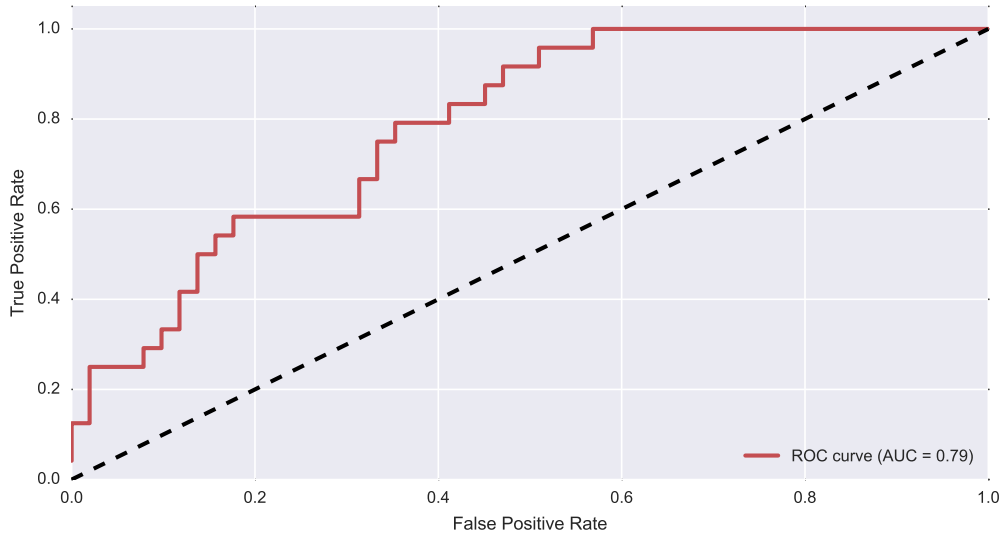


Figure 7: Receiver Operating Characteristic Example

An example of a ROC plot is shown in Figure 7, where the red line represents the relationship of the TPR to the FPR at various thresholds. An ideal prediction would result in a point at the coordinate $(0, 1)$ of the plot, representing no *FN* nor *FP*, in which case the model would have 100% sensitivity and 100% specificity and is known as a *perfect classification* [35]. The diagonal line represents the outcome of a perfectly random guess and is referred to as the *line of no discrimination* and is drawn from the left bottom at coordinate $(0, 0)$ to the top right at coordinate $(1, 1)$. The diagonal is used to divide the ROC space, as such that points that lie to the northwest above the diagonal represent good classification performance that is better than random. Whereas, points that lie below the diagonal line, to the south east, are considered poor and are worse than random [28, 35].

While the ROC plot provides a beneficial visualization to describe the performance of a classifier, often a single numeric value is needed in order to quickly

compare the performance of models based on the ROC metric without relying solely on visualizations. The Area Under Curve (AUC) also referred to as Area Under Receiver Operating Characteristic (AUROC) visually implies the area that is contained under the curve given by the classifier in the ROC plot. The AUC is equal to the probability of the classifier to rank positive instances higher than negatives when selected uniformly random [35, 37].

$$\mathrm{TPR}(T) = \int_{T}^{\infty} f_1(x)\,dx \tag{8}$$

$$\mathrm{FPR}(T) = \int_{T}^{\infty} f_0(x)\,dx \tag{9}$$

$$\begin{aligned} \mathrm{AUC} &= \int_{\infty}^{-\infty} \mathrm{TPR}(T)\mathrm{FPR}'(T)\,dT \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(T' > T) f_1(T') f_0(T)\,dT'\,dT \\ &= P(X_1 > X_0) \end{aligned} \tag{10}$$

The calculation of the AUC can be described in terms of the TPR and FPR in the case of a binary classifier. Each prediction outcome is described in terms of a continuous random variable, X which is given compared to a threshold, T, normally a value of $T = 0.5$. In each instance the outcome is positive given $X > T$, and negative if $X \leqslant T$. The probability density of X is given as $f_1(x)$ for instances that are positive and $f_0(x)$ for negative outcomes, giving the TPR and FPR as the integrals in Equation 8 and Equation 9 [35, 37, 38]. Lastly, the AUC, which is the probability of the classifier to rank positive instances higher than negatives, can be described in Equation 10, where the area under the curve is represented by the reversed integrals. Thus giving the probability $P(X_1 > X_0)$ of the positive instance, X_1 ranked higher than a negative instance X_0 [37, 38].

2.2 LITERATURE REVIEW

In the following section we present a comprehensive literature review of the numerous approaches that researchers have taken to better understand the Bitcoin blockchain and markets. We provide a review of the leading research conducted to analyze the Bitcoin blockchain and understand the actions of individual users, the wealth inequality of the network, and conclusions about the future of Bitcoin. Following this we then review the various approaches taken by researchers to better understand and predict the Bitcoin markets based on the emotions shared by users on social media. We find based on our review, that in nearly all cases studying the emotions of Bitcoin users on social media does not predict the market, but rather reflects the sentiment of the community towards market changes. Lastly, we then review the leading machine learning predictive models that are applied to Bitcoin in order to predict the future price direction, which we find have between 50%–55% accuracy.

2.2.1 *Bitcoin Network Analysis*

There have been many questions surrounding the rise in popularity of Bitcoin and the motive behind individuals adopting it in comparison to traditional financial instruments. In "Bitter to Better—How to Make Bitcoin a Better Currency," Barber et al. [39] conducted an in-depth investigation of the Bitcoin network to better understand the reasons behind its success in comparison to previously proposed solutions. These prior solutions, referred to as *e-cash* or *electronic cash*, which had not succeeded after over three decades of research, and the authors wished to understand the secret to Bitcoin's success [40, 41]. The authors found that while e-cash systems had repeatedly failed, Bitcoin achieved wide-spread adoption without the use of fancy cryptography. Rather, Bitcoin relied on ingenuity and sophistication, in combination with the introduction of mining, to incentivizes user adoption [39]. In order to better comprehend the rise in the adop-

tion of Bitcoin Barber et al. laid out a detailed analysis of the major factors to the success of Bitcoin [39], which we have summarized as follows:

NO CENTRAL POINT OF TRUST: Bitcoin is completely distributed in nature and is able to operate entirely without any central authority, the blockchain and mining process allow for a majority vote for dispute resolution.

INCENTIVES AND ECONOMIC SYSTEM: The ingenuity behind Bitcoin's adoption is the incentive for users to become miners, allowing anyone to participate in the transaction processing and be rewarded for doing so with bitcoins.

PREDICTABLE SUPPLY: Bitcoin currency can only be added to the available supply as a reward for the mining process. The total supply is mathematically defined and predictable.

DIVISIBILITY AND FUNGIBILITY: The currency can be easily divisible up to the minimal unit of 0.00000001 BTC, referred as a *Satoshi*, allowing users to transact nearly any amount and combine any amount.

VERSATILITY AND OPENNESS: The development of Bitcoin is completely open, the source code is developed using an open-source license and has hundreds of developers contributing to the software.

SCRIPTING: The Bitcoin blockchain allows for the inclusion of simple scripts, which can be used to create complex transactions and financial services such as escrow and dispute mediation.

TRANSACTION IRREVERSIBILITY: Transactions within the Bitcoin network are irreversible once seen by other peers on the network, and once added to the blockchain are practically impossible to reverse without a 51% attack.

LOW FEES: The Bitcoin network has extremely low transaction fees with no additional costs for international transfers, which in addition to transaction irreversibility is appealing to merchants.

NUMEROUS IMPLEMENTATIONS: There are numerous implementations of Bitcoin, supporting all major operating systems and devices, in addition to cloud based services making access readily available.

Addressing the question of why individuals are adopting Bitcoin in comparison to traditional financial instruments, Bohr and Bashir [42] conducted an exploratory analysis of Bitcoin users. Prior to the research by Bohr and Bashir very little was known about the users of Bitcoin, their research used the data from a publicly available survey given to the Bitcoin community in terms of wealth accumulation and optimism about the future. In total there were 1,193 responses collected from 2013-02-12 – 2013-04-04, the survey information was made public online to a number of different Bitcoin communities, but due to the limitations in the widespread adoption of Bitcoin, it was nearly impossible to guarantee a random sample [42].

The analysis of the survey revealed many insights into the network of Bitcoin users, the average respondent age was approximately 33 years old, and less than half indicated their residence in the United States. The gender was disregarded from the research as there was almost no variation, with 95% of all respondents identifying as male [42]. The political orientation of users was also considered, with nearly half of the sample identifying as Libertarian and many represented as Progressives, with the remaining including socialists forming a minority. In addition to political orientation, numerous features were assessed for their influence on the accumulation of wealth within the Bitcoin network, with age and activities being a significant feature. The analysis of the survey showed 25 year olds having approximately half as many bitcoins as 35 year olds, who had again nearly half as many as 45 year olds, after which the trend declined [42]. Furthermore, the influence of mining and illicit activities (e. g. narcotics, gambling, etc.) on wealth accumulation were also evaluated, with mining having no significant impact since 2011, declining since the early days of Bitcoin and the participation in illicit activities only having a marginal effect [42].

The most important results from the research conducted by Bohr and Bashir to our own research, was their final assessment of the leading factors contributing

to wealth accumulation within the Bitcoin network. The results of their regression experiments showed that increasing age (up to a peak at 50), involvement in investment activities, and participation in the online forum Bitcoin Talk [43], the original forum used by Bitcoin's creator, were all factors in explaining the accumulation of wealth. These three factors were all statistically significant in describing wealth accumulation, each having a p-value of $p < 0.01$, with the authors finding that users whom identified as investors had accumulated approximately four times as many bitcoins as those who had not [42]. The statistical significance of investment activities and involvement in online forums to wealth accumulation demonstrated by Bohr and Bashir were instrumental in the design of our metrics to identify major network influencers.

Rather than focusing on the adoption of Bitcoin and individual users, other researchers focused on analyzing the blockchain, a public record of all transactions that take place on the network. In "Quantitative Analysis of the Full Bitcoin Transaction Graph," Ron and Shamir [44] downloaded the complete blockchain data spanning a period of 2009-01-03 – 2012-05-13 and analyzed many of the statistical properties of its associated transaction graphs. In addition to conducting a detailed analysis of the blockchain, they provided an algorithm based on a variation of *Union-Find* [45], which made it possible to combine sets of addresses expected to belong to the same user based on combining transactions containing overlapping sets of addresses as the sender [44]. The algorithm was further expanded by Reid and Harrigan [46] and Androulaki et al. [47] to also incorporate *shadow addresses*, which were added by Bitcoin developers to increase security by generating a new address after each transaction to receive the change [47].

Through the application of the *Union-Find* algorithm, Ron and Shamir were able to combine 3,120,948 addresses, which had participated in sending at least one transaction, out of a total of 3,730,218 addresses into 1,851,544 individual users [44]. Out of the total 3,730,218 addresses there were 609,270 *sink addresses*, that had only received BTC but never sent any; these *sink addresses* combined (on 2012-05-13), held over 7,019,100 BTC, almost 78% of all BTC in existence [44]. Their analysis showed a consistent trend of huge variances in the statistics, for

example, in terms of the number of addresses associated to a single user, they had found a user with over 156,722 addresses, all of which were active [44].

The statistical analysis provided by Ron and Shamir was instrumental in directing our own analysis of the blockchain outlined in Section 3.1.4. We generated our descriptive statistics showing the number of addresses and wallets for each range of balances, based on the tables they created to analyze the wealth distribution. From their descriptive statistics there were a number of insightful outcomes, the balance of 98% of all users was $<$ 10 BTC ($50 USD on 2012-05-13), even when taking into consideration the maximum balance ever held by a user this amount was only reduced to 91% [44]. Not only do the vast majority of Bitcoin users only hold a relatively small amount, the vast majority (97%) of all users performed fewer than 10 transactions each, while 75 users were heavily active performing over 50,000 transactions each [44]. Given the majority of users have very little Bitcoin, perform very few transactions, and the fact that the majority (47%) of transactions are very small ($<$ 0.1 BTC each) [44]. The extreme variances in the statistical analysis of the network and the extreme unbalance in the influence that a minority of users have on the network becomes evident [44].

Following the detailed blockchain analysis provided by Ron and Shamir, much of future research focused on identifying the individual users and analyzing the services they used. In "A Fistful of Bitcoins: Characterizing Payments Among Men with No Names," Meiklejohn et al. [48] expanded previous address clustering techniques [44, 46, 47] to also cluster based on shared authority using identification attacks to identify and categorize Bitcoin related services [48]. They began by combining addresses into wallets owned by individual users using existing clustering techniques, following this the wallets were further combined and labelled using an identification attack. Where their attacking involved using a broad range of services, in order to discover the associated addresses belonging to the wallets owned by these services. After labelling major services, they proceeded to identify individuals by crawling the popular services Bitcoin Talk [43]

and Blockchain.info[1], associating the addresses published by users with their actual identity.

After labelling and categorizing Bitcoin related services and identifying individual Bitcoin users, they proceeded to analyze the activities of known services and users. Their results corroborated with Ron and Shamir [44] that a significant majority of bitcoins are in *sink addresess*, that have never been spent. They also demonstrated how gambling services are putting a lot of stress on the network, in particular the service Satoshi Dice. The gambling website has varying odds with a 2% house advantage, does tens of thousands of transaction a day, and has also spawned numerous clones, which all combined add an extra $30,000$ transactions to the blockchain daily [48]. While contributing a significant amount of transactions to the network, the gambling services had very little contribution to an overall increase in user balance or accumulation of wealth, instead the majority of active bitcoins were used with exchange services [48].

Building on the body of previous research Spagnuolo, Maggi, and Zanero [49] published the creation of a public system in "BitIodine: Extracting Intelligence from the Bitcoin Network." The purpose of this system, was to make available to the public the combined results of the known address clustering algorithms [44, 46, 47] and the labelling process used to identify services and users by Meiklejohn et al. [48]. The system that Spagnuolo, Maggi, and Zanero created was influential to the design of our own pre-processing system in Section 3.1.1. As well, the author provided us with consultation regarding the blockchain processing and address clustering methodology, which we expanded upon further to make parallel for enhanced performance.

The work of Kondor et al. [50] in "Do the Rich Get Richer? An Empirical Analysis of the Bitcoin Transaction Network," was very influential to our own research, as it provided a measure of the wealth and transaction statistics of the network; which were significant in the design of our own metrics for identifying network influencers. Kondor et al. identified two phases in the Bitcoin network; the first phase, referred to as the *initial phase*, which lasted until fall of 2010, was a point

1 Blockchain.info address self-identification service: `https://blockchain.info/tags`

where the system had low activity and was experimental. The second phase, referred to as the *trading phase*, is when Bitcoin began to function as a real currency and occurred when bitcoins gained monetary value through the introduction of online exchanges [50]. Following the start of the *trading phase*, all of their network measures began to converge to their now typical values and since mid 2011 did not change significantly afterwards [50].

In order to better understand the distribution of connectedness and wealth within the Bitcoin network Kondor et al. applied the *Gini coefficient*,

$$G = \left(2 \sum_{i=1}^{n} i \cdot x_i \Big/ n \sum_{i=1}^{n} x_i \right) - \frac{n+1}{n}, \tag{11}$$

which is used in economics to characterize the amount inequality present in the wealth distribution of a population [51]. In Equation 11, x_i represents a sample of size n, where the x_i are ordered monotonically such that $x_i \leqslant x_{i+1}$. A value of $G = 0$ represents perfect wealth equality, where every individual has the same wealth, $G = 1$ is complete inequality, where a single individual controls all of the wealth [50–52].

Using the Gini coefficient it was found that during the *initial phase* of Bitcoin that the in-degree distribution (the number of transactions from unique addresses) was close to $G \approx 1$. This indicates, that prior to the *trading phase*, most users hoarded their bitcoins, as there were few Bitcoin services available or exchanges to convert their bitcoins to fiat currencies [50]. Following the *trading phase*, the amount of in-degree and out-degree approached a steady value of $G_{in} \approx 0.629$ and $G_{out} \approx 0.521$ [50], indicating more even distribution of transactions between addresses. However, it is interesting to note that even with the increase in connectedness, the Gini coefficient for wealth distribution went from $G_{wealth} \approx 0.85$ to $G_{wealth} \approx 0.985$ [50]. Leading Kondor et al. to conclude that while it is well-known that the wealth distribution of society follows the Pareto rule, where the top 20% of the population control 80% of the total wealth [53, 54], the wealth distribution of the Bitcoin network is highly heterogeneous with 6.28% of the addresses possessing 94.72% of the total wealth [50]. Even more interestingly, of

those that have a small amount of wealth, they also found that there are a significant number of addresses that lose all of their wealth within a time frame of one month, a phenomenon specific to Bitcoin [50].

The insightful research conducted by Kondor et al. further re-affirmed our methodology for identifying major influencers based on wealth accumulation, referred to as *preferential attachment*. Whereby the well-connected and wealthy individuals increase their wealth and influence faster than those with less [55, 56], the "rich get richer" and continue to exert their influence on the network [50]. Furthermore, the prior research on address clustering conducted by Ron and Shamir et al. [44, 46, 47] and the detailed labelling of services and user identification research of Meiklejohn et al. [48, 49], provided a detailed and constructive body of research to build upon for conducting our methodology.

2.2.2 *Bitcoin Sentiment Analysis*

Previous research work has been conducted to better understand the effect that external sentiment has on the Bitcoin market and price volatility. In "Nowcasting the Bitcoin Market with Twitter Signals," Kaminski and Gloor [57] were the first to apply existing techniques of social media sentiment analysis to the Bitcoin market, which operates 24 hours a day. Their work expanded upon the previous research of Bollen, Mao, and Zeng [58], which demonstrated an accuracy of 87.6% in predicting the daily direction changes of the Dow Jones Industrial Average (DJIA). As well as Zhang, Fuehres, and Gloor [59], which concluded that tweets from Twitter have a high causality relation with stock market indices and can be considered as predictors of financial market movements. For their research Kaminski and Gloor collected a total of 161,200 tweets from 57,727 unique Twitter users within the time-frame of 2013-11-23 – 2014-03-07. In total the collected tweets had a combined sum of 300 million impressions, where impressions are the number of times that tweets have been viewed by Twitter users [57]. Market data from the four leading Bitcoin exchanges: BitStamp, Bitfinex, BTC-E, and BTC China was used to extract the following price metrics: Open High Low

Close (OHLC), Volume (BTC), Volume (currency), intraday-spread $(high - low)$, intraday-return $(open - close)$, and Δ price $\left(close_{day+2} - close_{day}\right)$ [57].

The results of the research conducted by Kaminski and Gloor showed a correlation with the amount of emotions and sentiments of uncertainty correlated with high trading volumes in the previous 24 hours. Furthermore, they found that high amounts of negative sentiment were a key influencer for the trading volume of the past 24 hours [57]. In particular, the sum of negative and uncertainty sentiments had a strong positive correlation with the trading volume of BitStamp, and the sum of total emotions and uncertainty sentiment also correlated to the intraday-spread on BitStamp. Lastly, they found that an increase in negative and uncertainty sentiments on a given trading day were likely to result in a lower close price, showing that there is a correlation between negative market performance and sentiment [57].

In addition to the research of Kaminski and Gloor, others also pursued analyzing correlations between social media sentiment and the Bitcoin market. In "Bitcoin Spread Prediction Using Social And Web Search Media," Matta, Lunesu, and Marchesi [60] investigated the spread of Bitcoin's price and its correlation to the volume of social media sentiment and Google trend data for a period of 2015-01-01 – 2015-03-01. Their research affirmed that of Kaminski and Gloor and demonstrated a correlation between positive social media sentiment and increased Google trends with positive price increase, however, their conclusions drew assumptions that correlation implies causation. The work of Georgoula et al. [61] in "Using Time-Series and Sentiment Analysis to Detect the Determinants of Bitcoin Prices," further affirmed the correlation between social media sentiment and the Bitcoin market. Similarly to other researchers, they also assessed the sentiment of Twitter, but also took into consideration the mining difficulty and number of searches related to Bitcoin on Wikipedia. The data was captured during a period of 2014-10-27 – 2015-01-12 and the type of sentiment contained in the tweets was classified using SVM, a series of regressions were conducted to evaluate the correlation. The results demonstrated a correlation between search

interest on Wikipedia and mining difficulty to the price of Bitcoin, in addition to positive Twitter sentiment having a short window of correlation to the price.

Rather than assess the correlation between Twitter social media sentiment and price, Bukovina, Marticek, et al. [62] took a very different approach and focused on the influence of sentiment on the overall volatility of the market. In *Sentiment and Bitcoin Volatility*, they assessed the impact of less rational factors such as the lucrative attractiveness of Bitcoin and speculation as influential factors of volatility, focusing particularly on the correlation between social media sentiment and volatility. Bukovina, Marticek, et al. define the term *sentiment* within the context of behavioral finance, where investor sentiment is a set of beliefs about the likelihood of a return on the initial investment and the known risks based on available facts [63]. They define the decomposition of the Bitcoin price as the following given in Equation 12, where BTC represents one bitcoin, T is the average number of transactions, k is a coefficient relating to the block reward given to miners, and R is the average revenue for each transaction [62]. Instead of using Twitter for their sentiment analysis, the data was provided by a third party service Sentdex [64], which collects social media sentiment from Reddit[2]. Their results indicated only a minor correlation between sentiment and market volatility, however during periods of high volatility, there was a much stronger correlation. The authors attribute the weak correlation results compared to other researchers as the limitation of only using Reddit for sentiment [62].

$$BTC = \frac{T}{\mathrm{k}} \times R \tag{12}$$

Instead of focusing solely on establishing correlations, Kaminski and Gloor extended their analysis of social media sentiment and market performance to also test the common anecdote of statistics, "correlation does not imply causation". Their research further evaluated the predictive possibilities of social media sentiment by applying Granger causality analysis [65, 66] to the time series market data and Twitter sentiments. The Granger causality was applied to the two linear

2 An online social network similar to a bulletin board, where users post submissions that can be voted up or down and commented on. http://reddit.com

regression models in Equation 13 and Equation 14, where the first model uses a lagged value p of the Bitcoin market data to predict Y_t and the second model uses a lagged value of Twitter sentiments, denoted by X_{t-i} [57].

$$Y_t = c_t + \sum_{i=1}^{p} \beta_i Y_{t-i} + e_t \tag{13}$$

$$Y_t = c_t + \sum_{i=1}^{p} \alpha_i Y_{t-i} + \sum_{i=1}^{p} \beta_i X_{t-i} + u_t \tag{14}$$

$$\text{with} \qquad H_0 = \beta_1 = \beta_2 = \cdots = \beta_p = 0$$

The two linear regression models are then evaluated using Residual Sum of Squares (RSS) as shown in Equation 15 to test the null hypothesis of the Granger causality given in Equation 14. Where RSS_0 and RSS_1 are the RSS for the two linear regression models given in Equation 13 and Equation 14. If the F statistic is outside a critical value for an F distribution, then the null hypothesis H_0 that Y does not have Granger causality of X is rejected, implying that Y is the Granger causality of X [65, 66].

$$f = \frac{(n - 2p - 1)\,(RSS_0 - RSS_1)}{p \times RSS_1} \sim F_{p,n-2p-1} \tag{15}$$

After applying Granger causality analysis to evaluate whether there was any causality between Twitter sentiment and market outcome based on the correlation assessed, Kaminski and Gloor found no statistical significance of Twitter sentiment as a predictor of Bitcoin price, intraday-spread, or intraday-return [57]. They concluded that the Twitter platform is more akin to a virtual trading floor that reflects the emotions of the movements that occur in the Bitcoin market, rather than predicting it [57]. Their outcome was influential in our own methodology and decision to focus on identifying the major market influencers based on wealth accumulation demonstrated by Kondor et al., rather than focusing on market influence based on sentiment.

2.2.3 *Predictive Models for Bitcoin*

There have been several instances where researchers have attempted to predict the Bitcoin market using the application of machine learning methods. The first attempt was published in "Bayesian regression and Bitcoin," where Shah and Zhang [67] applied the concept of Bayesian regression in the form of a "latent source model" as introduced by Chen, Nikolov, and Shah [68] to function as a binary classifier. In their paper they utilized the latent source model of Bayesian regression to predict the numerical change in price of Bitcoin, following this predictive model they proceeded to devise a simple trading strategy and backtest it again their withheld testing data. The data used for the experiments was taken from a major Chinese exchange OKCoin for the time period of 2014-02-01 – 2014-07-01, the combined total number of data points was over 10 million as they used the real-time trading events which were acquired at an interval of every two seconds [68].

Based on the work of Chen, Nikolov, and Shah [68], the latent source model was adapted for Bayesian regression of the Bitcoin market. When the problem is applied to a latent source model it becomes a simplified Bayesian inference problem; first given n amount of data $(x_i, y_i), 1 \leqslant i \leqslant n$ the empirical conditional probability is given in Equation 16 [69]. From the empirical conditional probability model Shah and Zhang then expanded it to the concept of a linear estimator where $X(x) \in \mathbb{R}^n$ such that it satisfies the equation given in Equation 17. From this point they were then able to derive the final predictive model where $\mathbf{y} \in \mathbb{R}^n$ with the value at index i being y_i, where $\hat{y} \equiv \mathrm{E}_{emp}[y \mid x]$, following this equivalence for the linear estimator gives the final model in Equation 18, which was used as the predictive model for the time series data [67].

$$\mathrm{P}_{emp}(y \mid x) = \frac{\sum_{i=1}^{n} \mathbb{1}(y = y_i) \exp\left(-\frac{1}{4}\|x - x_i\|_2^2\right)}{\sum_{i=1}^{n} \exp\left(-\frac{1}{4}\|x - x_i\|_2^2\right)} \tag{16}$$

$$\mathrm{E}_{emp}[y \mid x] = \frac{\sum_{i=1}^{n} y_i \exp\left(-\frac{1}{4}\|x - x_i\|_2^2\right)}{\sum_{i=1}^{n} \exp\left(-\frac{1}{4}\|x - x_i\|_2^2\right)} \tag{17}$$

$$\hat{y} = X(x)\mathbf{y} \tag{18}$$

The process of understanding how the information from the historical data for Bitcoin can be applied to predict the future price, relies on the principles of quantitative financial strategies. These strategies assume that markets follow patterns such that the past activities can be used to predict future prices to a certain extent [70]. The purpose of the latent source strategy is to model the existence of these underlying patterns that have an effect on price variations; rather than relying on a human expert to identify patterns in the data. Thus, the Bayesian regression model was applied by Shah and Zhang to utilize the presence of patterns for prediction without explicitly defining them. A simple trading strategy was applied where a position is maintained at either $+1, 0, -1$ bitcoin based on the average price change Δp predicted by the Bayesian regression model. When $\Delta p > t$, a threshold, then buy bitcoin and if $\Delta p < -t$ then sell, otherwise do nothing [67]. The core strategy of predicting the price change Δp is based on the application of the Bayesian regression model in Equation 18 based on historical data x^1, x^2, x^3 of intervals $30, 60, 120$ minutes. The final estimation of Δp is given in Equation 19, where $\boldsymbol{w} = (w_0, \ldots, w_4)$ are learnt parameters that provide the linear fit over all three historical time periods selected [67].

$$\Delta p = w_0 + \sum_{j=1}^{3} w_j \Delta p^j + w_4 r \tag{19}$$

The results of the Bayesian regression predictive model and trading strategy produced an inverse relationship where the profit from each trade was reduced given the number of trades [67]. This is intuitive due to the volatility of Bitcoin demonstrated in Section 1.2. The best model created was applied to the testing data and achieved an average investment return of $3,362$ CNY profit after executing $2,872$ trades with an investment of $3,781$ CNY; giving a return of approximately 89% over the course of 50 days and a Sharpe ratio of 4.10 [67]. The Sharpe ratio as given in Equation 20, where L is the number of trades during the time interval and $p_1, \ldots, p_L$ are the profits and losses; C is the modulus of difference

between start and end price for the interval [67, 71]. The Sharpe ratio is frequently used to compare how well a strategy consistently performs when compared to a risk-free strategy [71].

$$S = \frac{\sum_{\ell=1}^{L} p_\ell - C}{L \times \frac{1}{L}\left(\sum_{\ell=1}^{L} (p_\ell - \overline{p})^2\right)} \tag{20}$$

While the return of 89% and Sharpe ratio of 4.10 demonstrated by Shah and Zhang is significant, their strategy has numerous limitations. Their trading strategy managed to generate a return of 89% over the course of 50 days, however during this period the value of Bitcoin in CNY more than doubled. In addition, their strategy executed 2,872 trades without taking into consideration the associated trading fees, which can vary between 0.25%–0.50%. Furthermore, their strategy has limitations in the amount of volume that can be traded, while they were only trading with a single bitcoin it does not scale to larger amounts such as 50 bitcoins. This is primarily due to the limited size of order books and amount of volume, which are still much less than normal financial markets. Lastly, their proposed latent Bayesian regression model is computationally infeasible; in order to consider all possible time series data for future predictions, rather than a small window of previous history, would require computation at a massive scale [67].

Extending the work of Shah and Zhang, other researchers attempted to build more robust predictive models, rather than focusing on the Δprice prediction, the future direction (`UP/DOWN`) of the price was predicted instead. In *Automated Bitcoin Trading via Machine Learning Algorithms*, Madan, Saluja, and Zhao [72] attempted to predict the price direction over the course of a five year period using daily data. Similarly to Shah and Zhang, they used OKCoin for the Chinese market while also using market data from Coinbase for the US market. The authors used a combined approach of a Generalized Linear Model (GLM) and random forest to predict the price direction within a time period of 10 minutes rather than the 10 second window that Shah and Zhang used. Based on their strategy, the outcome of their model demonstrated a prediction accuracy of 50%–55% [72].

Instead of focusing on price patterns, in "Using the Bitcoin Transaction Graph to Predict the Price of Bitcoin," Greaves and Au [73] evaluated the predictive

power of the network features of the blockchain in predicting the future price direction of Bitcoin. Their work was inspired by the prior research of both Shah and Zhang and Madan, Saluja, and Zhao, and they sought to provide a comprehensive comparison of logistic regression, SVM, and neural networks. The dataset used for the research consisted of all transactions from the Bitcoin blockchain for the period of 2009-01-03 – 2013-04-07, market data was taken from Bitcoin Charts [74], the same service used in our own research methodology. In total the dataset contained approximately 37 million transactions between 6 million addresses, they applied the *Union-Find* algorithm introduced by Ron and Shamir [44] to combine the addresses into just over 3 million users.

As part of the pre-processing, Greaves and Au represented the data as a directed graph, where all of the nodes represented users, with each edge as a transaction between users. As well, any bitcoins acquired from mining were represented as a self-loop back to the user that received the mining reward. Following the pre-processing, the feature engineering consisted of identifying features based on the activity of the network at hourly intervals. The features included the number of transactions, mean transaction value, number of new addresses, average node in and out degrees, and several others. Furthermore, their feature engineering also consisted of identifying three network influencers based on their total transaction volume, with over 10% of all bitcoins ever sent passing through at least one of these users [73]. The users are labelled as *A, B, and C*, and each proved to be influential to the market, where they found that the transaction volume and closeness centrality of each were the most significant. Greaves and Au also found the historical Bitcoin price and number of blockchain transactions were also significant features in addition to the actions of users *A, B, and C* [73]. The significance of just the top three influential users as features for the predictive model used by Greaves and Au was influential to own our methodology in Section 3.2, where we created metrics to identify all major network influencers.

The data was collected hourly, with the data from 2012-02-01 – 2013-02-01 used for training and from 2013-02-01 – 2013-04-01 for testing, approximately a 10% holdout. Their experimental setup is similar to one of several holdout experi-

ments that we conducted for our own experiments in Section 4.3. Each of the predictive models evaluated by Greaves and Au used the training data (90% of all data) to predict the remaining 10% holdout that was used to test the model; the price direction (`UP/DOWN`) was predicted hourly and the accuracy of each model was compared. For their experiments a linear regression of only the historical price was used as a baseline, which was compared to logistic regression, SVM, and a feed-forward neural network, each trained using feature-engineered blockchain and market features [73]. The accuracy of each predictive model was compared for the test data, the results of which are shown in Table 3. However, solely comparing the accuracy is a poor metric that can lead to biases as shown in Section 2.1.2, our own experimental setup as described in Section 4.1.2 compared numerous metrics to accurately evaluate the performance of our predictive models.

Classification Model	**Accuracy**
Baseline	53.4%
Logistic Regression	54.3%
SVM	53.7%
Neural Network	55.1%

Table 3: Experimental Results of Greaves and Au [73]

The experimental results of Greaves and Au were significant to the establishment of our own research methodology and provided a clear reference in comparing the performance of several predictive models. Furthermore, their feature engineering demonstrated the importance of network influencers in predicting the price; but they limited their features to just tracking the three most influential users within the Bitcoin network. Based on their results we established a set of metrics in Section 3.2 for ranking the influence of users within the Bitcoin network, and then based on this metric extracted features for all major network influencers. Given the outcome of our own results shown in Chapter 4, this strategy proved to be beneficial in further improving the predictive accuracy of our models in comparison to the existing body of published research.

Chapter III

PROPOSED METHODOLOGY

3

PROPOSED METHODOLOGY

Our methodology consists of three distinct sections, each depending on the data and results from the previous section. The first section of our methodology, *Pre-Processing*, involves the analysis and formatting of the data, this was required in order to extract the necessary information from the Bitcoin blockchain, identify the groups of addresses belonging to each user, labelling known services, and performing statistical analysis of the Bitcoin network. Following the pre-processing, the next section, *Identifying Major Influencers*, describes the creation of our metrics used to identify all major influencers within the Bitcoin network based on their recorded activities on the blockchain. The metrics for identifying all major influencers on the Bitcoin network are essential to supporting our hypothesis that features extracted from major influencers can be used to predict the price direction of Bitcoin markets. Lastly, following the creation and application of the metrics to identify all major influencers we then describe our detailed process in *Predictive Model Creation*, for creating the final predictive models used in our experiments.

3.1 PRE-PROCESSING

The pre-processing is essential to all of our research, as the data must be transformed from its unstructured natural form into a format that can easily be analyzed and processed using Structured Query Language (SQL), which provides a convenient method of extracting the features needed from the blockchain data. For our pre-processing, we start by processing the Bitcoin blockchain and storing all of the information contained within the entire blockchain in a Relational Database Management System (RDBMS), allowing us to extract any information

or features needed using SQL. After storing the entire Bitcoin blockchain in a database, we apply our parallel address clustering algorithm, which is an extension of prior research, in order to efficiently identify all of the addresses belonging to each user within the network. We then gather external data that is publicly available including the market data listing the historical price and volume for all major exchanges, as well as labels to identify services and users. We rely on the labels in order to identify all major Bitcoin services and related companies, as well as known public figures and users. For the final pre-processing step, we generated various descriptive statistics of the Bitcoin network, extending previous research from 2012 – 2014, in order to assess the network statistics as of the current year, 2016.

3.1.1 *Blockchain Processing and Database*

There are many challenges inherent in analyzing and processing the Bitcoin blockchain. While the blockchain is intended to securely store information and keep a public ledger of all transactions, it is not practical to process or analyze. The blockchain was designed solely to record a ledger of all transactions, and provides no SQL support or similar querying capabilities, making it difficult to analyze unless the information is extracted and stored in different format.

For our research we required the creation of a RDBMS to store all of the information contained within the blockchain in a format that could be easily analyzed using SQL. We based our database schema off previous schema designs used by other researchers and supported by existing open-source libraries for extracting information from the blockchain [48, 49]. We designed the database schema around existing open-source tools and extended it with tables to support more detailed information pertaining to individual transactions and address clustering. As well, we added tables to keep historical aggregations for each address while processing the blockchain, rather than needing to perform many expensive queries after it had already been processed. For our RDBMS we used Post-

greSQL 9.5[1], an advanced open-source database, which has extensive support of SQL standards in addition to the ability to write complex SQL functions.

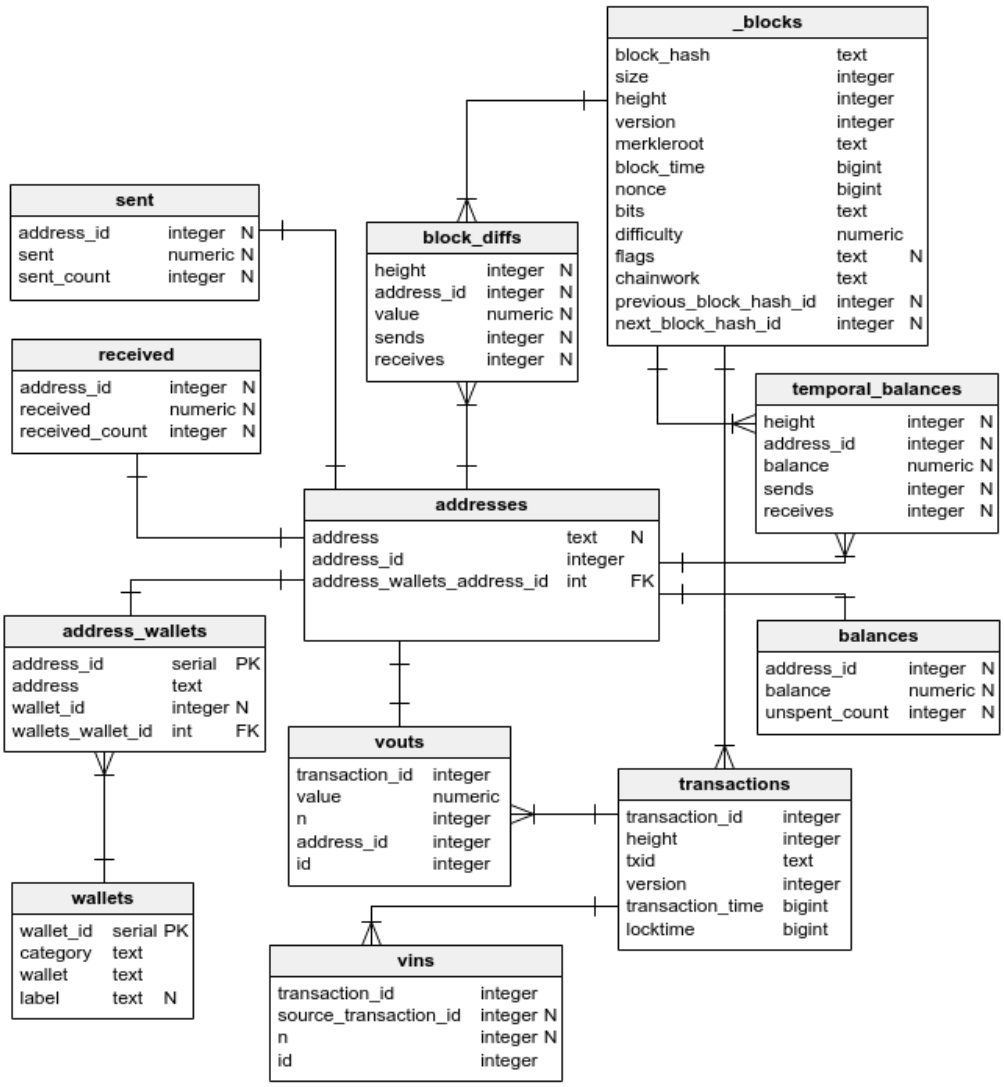


Figure 8: Bitcoin Blockchain Schema

The database schema we created for extracting information from the blockchain is given in Figure 8. We based the design around a star schema, which is commonly used when creating an Online Analytical Processing (OLAP) design for RDBMS. Our schema is centralized around the `addresses` table, which contains all of the Bitcoin addresses in the blockchain, nearly every other table references this table as addresses are central to the exchange of bitcoins in any transaction. In addition, we also created many tables which stored aggregate results for each

1 PostgreSQL: The world's most advanced open-source database. https://www.postgresql.org

address while processing the blockchain, the `sent` and `received` table record the amounts sent and received by each address. Furthermore, the `balances` table contains the final balances of each address, whereas the `temporal_balances` table contains the historical balance and number of transactions sent and received across each block in the blockchain. Lastly, we added two tables used to combine addresses into wallets that belong to individual users, as well as label services. The `address_wallets` table contains a mapping of each address to the wallet it belongs to after being clustered, and the `wallets` table contains a list of known services and public figures associated with each wallet.

We processed the Bitcoin blockchain and extracted the information using our own blockchain parser created in Python, we based the design on the software programs `bitcointools`[2] and `blockparser`[3], which are used by other researchers [48, 49]. The complete Bitcoin blockchain was processed and all transactions from the period of 2009-01-03 – 2016-02-01 were extracted and stored in our RDBMS, as well while processing the blockchain, all of the aggregations for each address were calculated and stored in the associated tables.

3.1.2 *Address Clustering*

Addresses individually do not capture all of the activities of Bitcoin users within the network, in order to properly assess major influencers we must determine all of the addresses owned by each user. The majority of Bitcoin users own more than a single address based on prior research [44, 47–49], and to effectively analyze all of the activities of Bitcoin users we must know all of the addresses belonging to each user. The process of combining addresses is known as *address clustering*, we also refer to the set containing all of the addresses belonging to a particular user as a *wallet*. We use the term *user* and *wallet* at times interchangeably to refer to all of the addresses that belong to the same entity. While often a collection of addresses belonging to a single entity is referred to as a *wallet*, we

2 Bitcoin tools library developed by Gavin Andresen. https://github.com/gavinandresen/bitcointools

3 Blockchain parser developed by znort987. https://github.com/znort987/blockparser

are not concerned about the collection of addresses in terms of a *wallet*, as we are interested in the user that is the owner of all the addresses. For all purposes we use the terms *user* and *wallet* interchangeably to refer to a single entity which has control over all of the addresses belonging to a set. The entity could be a Bitcoin related company or service, an individual person, or even an autonomous software service.

The most common algorithm for clustering addresses into wallets is the *Union-Find* algorithm first applied by Ron and Shamir [44] to combine multiple addresses belonging to the same user. Their research has been frequently corroborated and applied with small variations by numerous other researchers to cluster addresses [47–49, 73], and as such our own parallel algorithm is based on the existing techniques applied by others. The common strategy applied in clustering addresses relies on tracking the sets of addresses used in transactions made by each user, whereby the blockchain records every transaction as a set of inputs from one user to a set of outputs to another user.

Clustering addresses involves the application of two heuristics, the first based on the fact that any transaction has a set of addresses as inputs and all addresses in that set belong to a single users. The second heuristic is based on accounting for *shadow addresses*, which were added to Bitcoin to make transactions more private by generating a new address to receive the change from each transaction. The change from each transaction is always sent to a new address created for the user, rather than being sent to an existing address owned by the user.

HEURISTIC I The first heuristic relies on clustering the addresses by combining the sets of addresses used to send Bitcoin in each transactions incrementally over the entire history of the blockchain. Where $A = \{a_1, a_2, \ldots, a_n\}$ representing the set of all known addresses that have received a transaction, and $U = \{u_1, u_2, \ldots, u_n\}$ the set of all users. We use S_i and R_i to represent the set of sender and recipient addresses S_i, $R_i \in A = \{a_1, a_2, \ldots, a_n\}$, where in each transaction $T_i = S_i \rightarrow R_i$, the set of all input addresses are considered to be *owned* by the user u_k represented as $u_k = \{a_i \mid \forall a_i \in S_i\}$.

HEURISTIC II The second heuristic relies on clustering the addresses based on the creation of each new *shadow address* used to hold the change from each Bitcoin transaction. In each transaction, unless the full amount held in the address is spent, the remaining balance is then sent to a change address. The *shadow addresses* are the new addresses created in each transaction to hold the remaining unspent amount of bitcoins. Where S_i and R_i represent the set of sender and recipient addresses and each transaction $T_i = S_i \rightarrow R_i$; if the cardinality $|S_i| = 2$, then one of the addresses a_i is a *shadow address* used to receive the change. Given that for each transaction a new *shadow address* is created, we determine the shadow address a_s to be whichever address is not a known address in A, given as $a_s = \{a_i \mid \forall a_i \in S_i \wedge a_i \notin A\}$, the shadow address is then considered to be *owned* by the user u_k.

The address from both heuristics can be clustered to the same user by combining the sets such that u_k is the union of both heuristics given as the following where $u_k = \{a_i \mid \forall a_i \in S_i\} \cup \{a_i \mid \forall a_i \in S_i \wedge a_i \notin A\}$. While the combination of both heuristics provides more coverage, the second heuristic has several known limitations, it will only work in the case where the number of addresses for recipient R_i is two. If both addresses have not been involved in any previous transactions then it will incorrectly label both addresses as owned by the same user, a mistake than can cause an avalanche effect of errors, making a single user or service that frequently gets many new Bitcoin users owning many more addresses than in reality. Furthermore, the dependence on checking the existence of each address $a_i \notin A$ makes the algorithm non-parallelizable, as the blockchain must be processed in sequential order.

For our parallel address clustering algorithm we based it on *heuristic I*, as other researchers have discovered issues using *heuristic II* [47] and due to the limitations of the algorithm it cannot be parallelized without introducing more false positive address clusterings. While *heuristic II* has the added benefit of clustering *shadow addresses*, which may not be possible to cluster otherwise if they are *sink addresses* [44, 48], addresses that never perform a send transaction. The limitations of *heuristic II* have restricted our implementation to parallelize *heuristic*

I as it provides assurances of the accuracy of the clustering results and is easily parallelizable. The parallelization is based on the concept of a divide-and-conquer algorithm, which allows for each portion of the data to be repeatedly *divided* and *conquered* by clustering the addresses. The list of all blocks containing transactions are ordered sequentially from oldest to newest, the entire ordered history of blocks β containing each transactions $\{T_1, T_2, \ldots, T_N\}$ are then evenly divided into separate ordered portions $(\beta_1, \beta_2, \ldots, \beta_J)$ based on the number J, $\{P_1, P_2, \ldots, P_J\}$ of available threads or processing cores. For each portion of the blockchain history β, the transactions within the block are then ordered from oldest to newest transaction $\left(T_i \mid \forall T_i \in \beta_j\right)$ for the transactions in block β_j. The addresses are clustered into sets owned by users u_k in parallel using *heuristic I*, and after all of the transactions in the block have been processed the sets of addresses belonging to each user u_k are combined using *Union-Find* [44, 45]. After all parallel processes complete the address clustering for each portion of the blockchain $(\beta_1, \beta_2, \ldots, \beta_J)$ the sets of all addresses owned by all users $U_J = \{u_k \mid \forall u_k \in \{P_1, P_2, \ldots, P_J\}\}$ are combined a final time using *Union-Find*. The resultant output reduces the number of users each time they are combined based on the union of their owned addresses.

3.1.3 *Gathering External Data*

With all of the information extracted from the Bitcoin blockchain we needed detailed historical Bitcoin market data in order to support training and testing our predictive models. In order to satisfy our research goals we collected all of the available historical price and volume data for all major USD, CNY, and EUR Bitcoin markets. Based on our literature review, we used the services provided by Bitcoin Charts [74], which was frequently used by other researchers. It is frequently used by other researchers as it provides reliable and accurate historical market data for all major Bitcoin exchanges [4, 57, 73].

Using the services provided by Bitcoin Charts, we were able to download compressed archives of the entire available market history for all major exchanges.

The exchange market and available period of historical data for each is given in Table 4. Due to the limited early adoption of Bitcoin which resulted in a very low volume and price, we restricted the period of market data to 2013-01-01 – 2016-02-01 for all of our experiments. Although we parsed the entire Bitcoin blockchain for the period of 2009-01-03 – 2016-02-01, due to the limited market activity and very low market volume and price of Bitcoin prior to 2013 we limited our experimental period to the beginning of 2013-01-01 until the end of our blockchain history. While all of our experiments were limited to 2013-01-01 or later, we still needed the historical price of Bitcoin even with the limited market data available prior to 2011-09-01. In order to gain access to the historical price data prior to 2011-09-01, we used the services provided by OANDA [6], a leading foreign exchange company, to extract an accurate daily closing price of Bitcoin for the entire period of 2009-01-03 – 2016-02-01.

Exchange	**Market**	**Active Period**
Bitfinex	USD	2013-03-31 – 2016-02-01
Bitstamp	USD	2011-10-04 – 2016-02-01
BTC-E	USD	2011-09-01 – 2016-02-01
Coinbase	USD	2015-01-31 – 2016-02-01
BTC China	CNY	2011-06-30 – 2016-02-01
OKCoin	CNY	2013-06-30 – 2016-02-01
bitcoin.de	EUR	2011-09-01 – 2016-02-01
BTC-E	EUR	2013-01-01 – 2016-02-01
Kraken	EUR	2014-01-31 – 2016-02-01

Table 4: Bitcoin Market Data from Bitcoin Charts

Following the collection of the historical market data, we created a separate database to store the time series market data for all exchanges. The schema used is given in Figure 9, where each of the exchanges are stored in a separate `exchanges` table and given a unique identifier. As well for each exchange in the `exchanges` table, we also list the currently trading pair to Bitcoin in the case where an exchange supports multiple currencies we list it as multiple rows in our table. The `history` table contains the entire historical market data as time series data and the time stamp contains the precise time of each summary of market data, stored

in intervals of fifteen minutes. For each exchange at each fifteen minute interval in the time series database, we record the OHLC of the price as well as the volume in Bitcoin and the currency pair. Lastly, we also calculate the weighted average of the price during this period weighted based on the amount of volume.

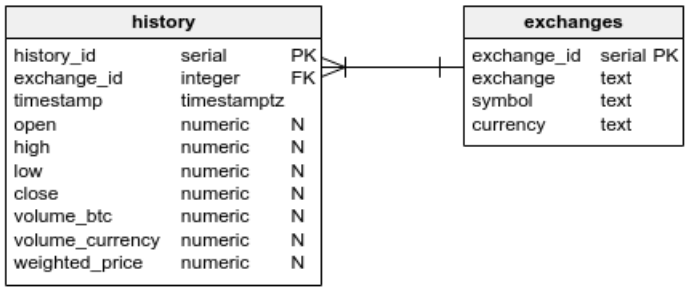


Figure 9: Exchange History Schema

Following the collection of the historical market data we also needed to collect a detailed set of labels in order to identify and name the Bitcoin services and influential players within the network. Initially we began collecting the known labels for wallets and identifying Bitcoin services and related companies by crawling the popular Bitcoin Talk [43] and Blockchain.info[4] websites as done by previous researchers [48, 49, 75]. While this strategy succeeded in labelling and identifying the owners of many wallets, we ran into numerous issues when attempting to identify the wallets owned by all of the major Bitcoin exchanges. We reached out to the creators of BitIodine, Spagnuolo, Maggi, and Zanero [49], who had used additional crawlers to collect even more labels. After contacting them, they graciously shared with us their detailed collection of labels for services and companies related to Bitcoin. Based on our own labels that we had collected, and the combination of the labels they provided, we were able to accurately identify and label all major Bitcoin exchanges and other related services.

4 Blockchain.info address self-identification service. https://blockchain.info/tags

3.1.4 *Statistical Analysis of the Blockchain*

As part of our pre-processing, we also created descriptive statistics to further analyze and understand the features of the Bitcoin network. We based our descriptive statistics on the work of prior research, in which many other researchers analyzed the number of addresses, wallets, and distribution of wealth [39, 44, 48, 50]. The statistical analysis of the network was applied to the entire blockchain history, spanning from the creation of the genesis block on 2009-01-03 – 2016-02-01. By analyzing the entire blockchain up until the time of writing (2016), it provided us with recent results compared to those published from several years prior.

In all of our analyses, we take into consideration the individual addresses in addition to the wallets, which are comprised of multiple addresses owned by a user. We began by analyzing the aggregate statistics for all of the features of each address and wallet, consisting of the balance, volume, and number of transactions. The results of our aggregate statistics for the entire Bitcoin network are given in Table 5; it is important to note the large distance between the mean and standard deviation from the median, this is indicative that the statistical distribution of the data is not normally distributed. Based on our literature review these results are corroborated with Kondor et al. [50], where they performed similar statistical analyses. They noted that nearly every feature follows a power-law statistical distribution, whereby each measurement can be approximated by an exponential to a degree of the other measurement [76]. Furthermore, we also found that the vast majority of addresses and wallets do not hold onto their wealth for a long period of time, rather the vast majority have no bitcoins remaining, with only 5.62% of addresses and 3.34% of all users having any bitcoins at all. This is a result corroborated with Kondor et al. [50] and unique to Bitcoin, we found that this feature greatly reduced the amount of users to analyze for our metrics, as the vast majority of users have no bitcoins and therefore exert no influence on the network.

Metric	Address			Wallet		
	Mean	Median	Std.	Mean	Median	Std.
Total	123,625,104			35,820,710		
Balance > 0	6,944,381 (5.62%)			1,197,847 (3.34%)		
Balance	2.18	0.001	155.14	1.88	0.001	139.99
Volume Sent	22.91	0.13	3,090.95	62.70	1.01	19,417.20
Volume Recv	22.08	0.12	3,026.65	62.76	1.01	19,424.81
Sent Trans.	2.41	1.00	314.75	6.89	1.00	4,278.25
Recv Trans.	2.58	1.00	311.02	7.34	1.00	4,307.36

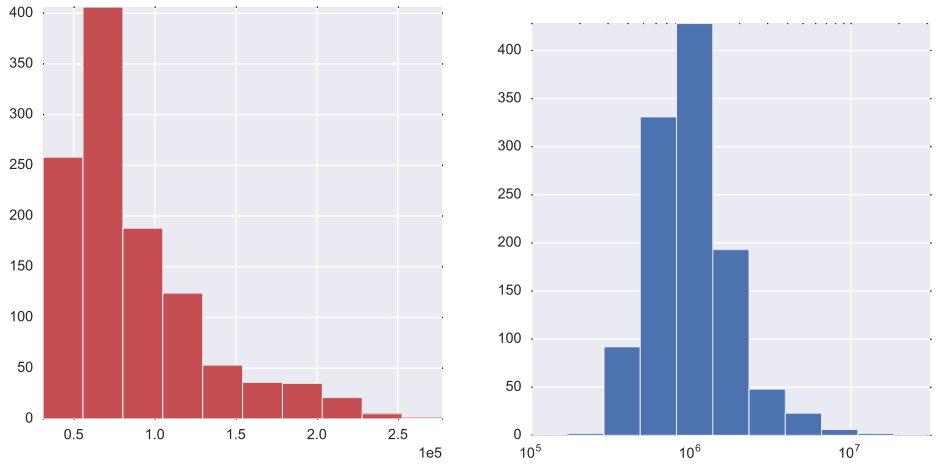
Table 5: Bitcoin Address and Wallet Statistics

In addition to analyzing the individual addresses and users of the network, we also created descriptive statistics to describe the daily number of transactions and the volume conducted by all users aggregated over several time periods. The results are given in Table 6 where we see that for both the number of transactions and volume that each follows a relatively normal distribution, without an extreme difference between the mean and median. The analysis of the daily activities of the entire network are important for our feature engineering as it implies frequent daily activity in addition to still expecting statistically minimal outlier events. We also plotted the distribution of the daily transactions and volume for the duration of our experiments spanning from 2013-01-01 – 2016-02-01 in Figure 10. For each histogram, we can see that the data is more normally distributed for the entire network when aggregated as a whole compared to the power-law distribution when analyzing individual users. Furthermore, we can see that the daily transactions fall between a range of 50,000–250,000 and the daily volume of bitcoins transacted lies between $10^{5.5}$–10^{7}. In each case, we see that there are no periods of outages where activity on the network halts, or days of exceptionally low network activity.

Following our assessment of the aggregate Bitcoin network statistics, we also evaluated the distribution of wealth amongst users within the network using a similar presentation to other researchers [44, 50]. In order to aid in in comprehending the real-world wealth accumulated by some of the wealthiest Bitcoin

Period	Transactions			Volume		
	Mean	Median	Std.	Mean	Median	Std.
Day	4.0×10^4	2.7×10^4	4.6×10^4	9.8×10^5	6.0×10^5	1.8×10^6
Month	1.2×10^6	9.2×10^5	1.4×10^6	3.0×10^7	2.1×10^7	4.3×10^7
Year	1.4×10^7	8.5×10^6	1.6×10^7	3.6×10^8	3.0×10^8	3.0×10^8

Table 6: Bitcoin Transactions and Volume



(a) Transactions (b) Volume

Figure 10: Histogram of Daily Transactions and Volume

users, we have provided the value in USD using an average of the price of Bitcoin throughout the period of our research from 2013-01-01 – 2016-02-01, where 1 BTC $\approx$ \$330 USD. We see in Table 7, the cumulative number of addresses and wallets with a balance of bitcoins greater than or equal to each amount is given. For each amount we see the number of addresses and wallets quickly decreasing with the vast majority of Bitcoin users having less than 0.1 BTC, worth \$33 USD.

In addition to the cumulative balance, we also evaluated the balance within a specific range of balances, something not done previously by other researchers. We found this analysis to be very beneficial as it highlights the amount of users within each specified range of balances rather than a cumulative breakdown. For each row of the table, we also show the number of addresses and wallets within the range of balances as a percentage out of the total of 6,944,381 addresses and 1,197,847 wallets. The results are given in Table 8, similarly to before we also provide the range of balances in USD, where 1 BTC $\approx$ \$330 USD. We see a

Balance (BTC)	Balance (USD)	Addresses	Wallets
> 0	0	6,944,381	1,197,847
0.1	33	1,011,135	152,245
0.25	82.5	758,208	108,634
0.5	165	588,965	83,303
1	330	413,603	60,822
2	660	299,024	42,482
5	1,650	188,748	25,175
10	3,300	129,501	15,966
25	8,250	84,146	8,093
50	16,500	29,308	4,609
100	33,000	12,926	2,538
250	82,500	6,085	1,098
500	165,000	3,375	551
1,000	330,000	1,583	281
2,500	825,000	455	84
5,000	1.65 M	212	37
10,000	3.30 M	108	19
25,000	8.25 M	45	8
50,000	16.5 M	16	2
$> 100,000$	33 M	2*	0

Table 7: Number of Addresses and Wallets by Cumulative Balance

* The two richest addresses are `3Kg7Cmooris7cLErTsijq6qR1FH3cTiK2G` and `3Nxwenay9Z8Lc9JBiywExpnEFiLp6Afp8v`, with 157,996 and 162,231 BTC; they have been active since 2014 and 2015, respectively.

vast wealth disparity in the Bitcoin network, with the majority of all addresses (85.44%) and when clustered into wallets (87.29%) having less than $33 USD worth of bitcoins. Even for up to a single bitcoin, we see that out of the entire network, 94.05% of all addresses do not have more than 1 BTC. When taking into consideration the clustered addresses into wallets, we see this increase to 94.92% of all users having no more than a single bitcoin. This vast wealth disparity within the Bitcoin network further reinforces our hypothesis that the major network influencers, the users with the vast majority of the wealth have an influence on the market price of Bitcoin.

Balance (BTC)	Balance (USD)	Addresses	Wallets
$(0, 0.1]$	($0, $33]	5,933,246 (85.44%)	1,045,602 (87.29%)
$(0.1, 0.25]$	($33, $82.5]	252,927 (3.64%)	43,611 (3.64%)
$(0.25, 0.5]$	($82.5, $165]	169,243 (2.44%)	25,331 (2.11%)
$(0.5, 1]$	($165, $330]	175,362 (2.53%)	22,481 (1.88%)
$(1, 2]$	($330, $660]	114,579 (1.65%)	18,340 (1.53%)
$(2, 5]$	($660, $1,650]	110,276 (1.59%)	17,307 (1.44%)
$(5, 10]$	($1,650, $3,300]	59,247 (0.85%)	9,209 (0.77%)
$(10, 25]$	($3,300, $8,250]	45,355 (0.65%)	7,873 (0.65%)
$(25, 50]$	($8,250, $16,500]	54,838 (0.79%)	3,484 (0.29%)
$(50, 100]$	($16,500, $33,000]	16,382 (0.24%)	2,071 (0.17%)
$(100, 250]$	($33,000, $82,500]	6,841 (0.10%)	1,440 (0.12%)
$(250, 500]$	($82,500, $165,000]	2,710 (0.04%)	547 (0.05%)
$(500, 1,000]$	($165,000, $330,000]	1,792 (0.03%)	270 (0.02%)
$(1,000, 2,500]$	($330,000, $825,000]	1,128 (0.02%)	197 (0.02%)
$(2,500, 5,000]$	($825,000, $1.65M]	243 ($< 0.01\%$)	47 ($< 0.01\%$)
$(5,000, 10,000]$	($1.65 M, $3.3 M]	104 ($< 0.01\%$)	18 ($< 0.01\%$)
$(10,000, 25,000]$	($3.3 M, $8.25 M]	63 ($< 0.01\%$)	11 ($< 0.01\%$)
$(25,000, 50,000]$	($8.25 M, $16.5 M]	29 ($< 0.01\%$)	6 ($< 0.01\%$)
$(50,000, 100,000]$	($16.5 M, $33 M]	14 ($< 0.01\%$)	2 ($< 0.01\%$)
$> 100,000$	$> $33 M	2 ($< 0.01\%$)	0 (0%)

Table 8: Number of Addresses and Wallets by Bounded Balance

After assessing the wealth distribution in the network we decided to track and label the top 25 wallets based on the number of addresses in the wallet, as well as the top 25 wallets based on the total combined balance of all addresses within the wallet. The results of the assessment of the top 25 wallets based on number of addresses and total balance can be found in Table 9 and Table 9. For each table where the owner of the wallet is known the wallet is labelled, otherwise if the owner is not known, we label the wallet as *unknown*. We see in Table 9 where the top 25 wallets by number of addresses are shown, that the vast majority of the wallets are labelled and belong to various services such as exchanges, where Bitcon can be exchanged for fiat currencies, and other Bitcoin related services and companies. In particular we also see the remaining balance of the Silk Road, one of the largest online markets for illicit goods, with more than $214 million USD in sales prior to its subsequent seizure by law enforcement in 2014 [77]. However,

when we look at Table 10 we see a very different result, the vast majority of the owners of each wallet are *unknown*, furthermore we see that there are numerous wallets containing only a single address for large amount of bitcoins.

The statistical analysis conducted provided insight into the inner-workings of the Bitcoin network and the individual users and services that make up the network. We found that the descriptive statistics of individual users follow a power-law distribution, results that corroborate with other research work [44, 50]. Furthermore, we found that there is still a vast disparity of wealth within the network, which has only gotten worse since prior analyses were published [44, 48, 50]. Lastly, after conducting an analysis of the top 25 wallets based on number of addresses and balance, we found that while the top 25 wallets based on number of addresses belong to Bitcoin related services, the majority of the wealthiest wallets likely belong to individual users, which we consider to be the network influencers.

Wallet	Category	Balance	Size	Active Period
Bitcoin-24.com	Exchanges	46,039.61	9,252,819	2010-08-19 – 2016-02-01
LocalBitcoins.com	Exchanges	40.78	571,994	2012-07-24 – 2016-02-01
AgoraMarket	Darknet/Market	227.89	497,995	2013-12-04 – 2016-02-01
EvolutionMarket	Darknet/Market	56.19	420,632	2014-01-16 – 2016-01-11
Unknown	*Unknown*	9.96	386,703	2012-06-06 – 2016-02-01
SilkRoad*	Darknet/Market	97.85	372,753	2012-06-17 – 2016-01-23
SilkRoad2	Darknet/Market	196.91	349,874	2013-11-11 – 2016-01-11
BTC-e.com	Exchanges	159.19	348,438	2011-08-08 – 2016-02-01
Unknown	*Unknown*	1.03	272,998	2013-12-27 – 2016-02-01
Cryptsy.com-old	Exchanges	1.92	259,048	2013-05-20 – 2016-02-01
Unknown	*Unknown*	51.25	223,988	2013-09-17 – 2016-02-01
Unknown	*Unknown*	1.02	220,609	2014-09-16 – 2015-11-10
BitcoinFog	Services/Others	0.00	187,458	2011-11-10 – 2016-02-01
BitPay.com-old	Services/Others	0.43	166,055	2011-07-02 – 2016-02-01
BitPay.com	Services/Others	1.75	155,696	2015-02-26 – 2016-02-01
Unknown	*Unknown*	126.10	148,320	2014-01-13 – 2016-02-01
Bittrex.com	Exchanges	510.11	143,479	2014-02-13 – 2016-02-01
Bitstamp.net	Exchanges	0.05	138,167	2011-08-17 – 2016-02-01
Unknown	*Unknown*	3.16	134,379	2013-06-16 – 2015-07-17
Cryptsy.com	Exchanges	5.81	126,805	2014-07-29 – 2016-02-01
Unknown	*Unknown*	1.21	112,186	2014-07-20 – 2016-02-01
Instawallet.org	Old/Historic	193.27	109,151	2011-05-03 – 2016-01-30
Unknown	*Unknown*	0.00	103,313	2014-10-09 – 2016-01-26
AbraxasMarket	Darknet/Market	292.85	102,527	2014-12-22 – 2016-02-01
CoinGaming.io	Gambling	0.00	100,685	2013-12-05 – 2016-02-01

Table 9: Top Wallets by Wallet Size

* Silk Road was the largest online market for drugs and elicit goods, with more than $214 million USD in sales. [77]

Wallet	Category	Balance	Size	Active Period
Unknown	*Unknown*	87,111.953	35	2011-06-13 – 2016-01-21
Unknown	*Unknown*	69,370.11	1	2013-04-09 – 2016-01-21
Bitcoin-24.com	Exchanges	46,039.61	9,252,819	2010-08-19 – 2016-02-01
Kraken.com	Exchanges	39,605.62	98,614	2013-09-09 – 2016-02-03
Unknown	*Unknown*	31,000.05	7	2010-05-04 – 2016-01-21
Unknown	*Unknown*	28,198.53	15	2010-07-23 – 2016-01-21
Unknown	*Unknown*	28,050.00	8	2015-02-16 – 2016-02-01
Unknown	*Unknown*	26,040.58	56	2014-10-08 – 2016-02-01
Unknown	*Unknown*	21,744.01	1	2013-12-19 – 2016-01-21
BTCCPool	Mining Pools	18,195.120	10,290	2013-11-16 – 2016-02-03
Unknown	*Unknown*	12,554.49	1	2012-12-24 – 2016-01-21
Unknown	*Unknown*	12,276.56	1	2015-08-06 – 2016-01-27
Unknown	*Unknown*	12,039.16	1	2013-02-26 – 2016-01-21
Unknown	*Unknown*	11,887.72	2	2013-03-27 – 2016-01-21
Unknown	*Unknown*	11,336.84	8	2014-11-24 – 2016-01-21
Unknown	*Unknown*	11,111.00	2	2011-10-17 – 2016-01-21
Unknown	*Unknown*	10,682.020	1	2013-02-28 – 2016-01-21
Unknown	*Unknown*	10,675.302	483	2011-08-28 – 2016-02-01
Huobi.com	Exchanges	10,379.675	10,575	2014-09-04 – 2016-02-01
Unknown	*Unknown*	9,424.783	1	2010-09-28 – 2016-01-21
Unknown	*Unknown*	9,112.213	1	2015-04-02 – 2016-01-21
Unknown	*Unknown*	8,700.010	82	2010-10-22 – 2016-01-21
Unknown	*Unknown*	8,516.261	32	2013-04-01 – 2016-01-21
Unknown	*Unknown*	8,194.514	8	2010-07-03 – 2016-01-21
Unknown	*Unknown*	7,941.062	78	2010-04-09 – 2016-01-21

Table 10: Top Wallets by Balance

3.2 IDENTIFYING MAJOR INFLUENCERS

As part of our hypothesis and one of our main objectives in Section 1.4, identifying the major influencers of the Bitcoin network is critical to the creation of our predictive model. We evaluate two metrics created to identify the major network influencers, the first referred to as the *bit-index*, the second as the *Pareto front*, is based on the optimal selection of users given the trade-offs of several parameters. The bit-index is a temporal filter that considers the activity of the

user over the entire history of their transactions on the blockchain, and is based on the concept of the h-index metric frequently used to measure the influence of researchers. The second metric, the Pareto front based method, selects all of the optimal users that lie on a *front*, given the relative trade-offs of each parameter to optimize [76]. Lastly, we evaluate each method used to identify the major influencers against criteria pertaining to our research hypothesis, and select the best of the two metrics to use for our feature engineering and predictive model creation.

3.2.1 *Proposed Bit-index Metric*

The bit-index is a method which operates based on the activities of wallets over their entire history of transactions on the blockchain. We based our metric on the concept of the *h-index*, which is commonly used in academia to rank the influence of researchers. In the following sections we outline our process for the bit-index calculation, as well as providing an example of the application of the metric to the Bittrex exchange wallet. Lastly, we apply the bit-index metric to the entire transaction history of all users within the Bitcoin network and provide an analysis of our results.

3.2.1.1 *H-index: Background*

The bit-index is a metric to accurately rank the influence of users within the Bitcoin network based on the entire history of a user's actions based on the attribute being measured. The bit-index is based on the ranking metric, *h-index*, created by Hirsch [78], which is used widely in academia to rank the influence of researchers based on the citations their publications receive [78].

The h-index provided a good basis for our metric as it is widely used in academia and has even been applied to demonstrate its potential predictive power [79]. The metric is described as the following shown in Equation 21, whereby the proportionality constant defined as α, and the h-index, h, are used to describe an author's total number of citations, $N_{c,tot}$. The relationship between $N_{c,tot}$ and h

depend on the distribution properties of the academic field, and the proportionality constant a is adjusted accordingly [78].

$$N_{c,tot} = ah^2 \tag{21}$$

Generally, the h will increase approximately linearly with time, as older publications continue to gain more citations and researchers continue to produce new publications. This factor is unique to the h index and differs from our own application of the metric, which does not consider the outcome of transactions in the distant past as increasing the current influence of the user. However, in the case of the h-index, a simple model describes the increase in citations with time due to older publications. As shown in Equation 22, where p papers are published each year and each paper earns c new citations per year every subsequent year, and j describes the number of citations per paper. The total number of citations after $n+1$ years is represented as a summation over the entire history of publications made by the author.

$$N_{c,tot} = \sum_{j=1}^{n} pcj = \frac{pcn(n+1)}{2} \tag{22}$$

While these models are practical in most cases, and make assumptions about the model being approximately linear with time, these over-simplifications do not apply to our own application. The h-index calculation has numerous issues as it relies heavily on the statistical distributions common to academia, which follow a Gaussian distribution [78]. A Gaussian distribution is much different than those typically found in the Bitcoin network, which are heavily skewed and tend to follow a power-law distribution [50, 76]. Based on our own analysis of the statistical properties of the Bitcoin network in Section 3.1.4, we found that a non-linear approach would be required in order to accurately determine the bit-index for each user.

The application of the linear assumptive h-index calculation follows the same principles as our derived bit-index. For the h-index, in order to satisfy the parameters of the linear model as shown in equation Equation 21, an appropriate cor-

responding value of a, typically between $3-5$ must be selected so that the value of h satisfies the equation. The optimal selection of h corresponds to a value that evenly partitions the upper and lower portions of the area under the curve given the number of citations per publication. This is demonstrated as an example in Figure 11 given by Hirsch [78], where each publication is ordered descending from most to least citations. The selection of the h-index is the point at which the paper number is equal to or exceeds the number of citations $p_i \geqslant p_j$, this coincidentally also corresponds to an equal partitioning of the left and right side of the area under the curve, such that each partition has an equal area.

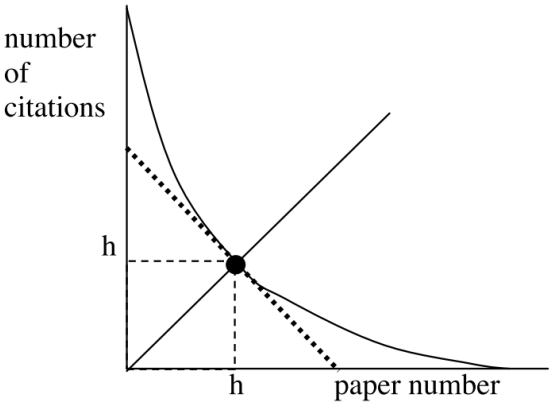


Figure 11: h-index Selection Based on Papers Ranked by Citations

Algorithm 1: h-index Calculation

input : A set of citations
output: The h-index

```
h-index({c_i, c_{i+1}, ..., c_n})
    // Sort in descending order the set of citations
    sorted ← Sort({c_i, c_{i+1}, ..., c_n})
    for i ← 1 to n − 1 do
        if sorted[i + 1] == ⩽ i + 1 then
            index ← sorted[i + 1]
            return (index)
        end
    end
end
```

As such, given the inherent relationship between $N_{c,tot}$ and h as shown in equation Equation 21, the following algorithm 1 is commonly used to calculate the h-index, based on the number of citations c for each consecutive publication

p. Our bit-index calculation follows a similar approach by first sorting the data in descending order, however, as shown in the following section, we calculate the optimal partitioning using a precise calculation of the area under the curve rather than the linear assumptions made for the h-index.

3.2.1.2 *Bit-index Calculation*

The bit-index is based closely on the description of the h-index provided by Hirsch [78], with the exception that the h-index makes assumptions about the approximate linear nature of the models, whereas the bit-index is applied to a non-linear model. Based on our prior analysis of the statistical properties of the Bitcoin network, many of the features are inherently a power-law statistical distribution, with a significant population of outliers, those users with very large amount of bitcoins relative to the other users. These extreme outliers and power-law distribution are not applicable to the linearly assumptive h-index, and as such, we had to modify our application of the h-index to account for the properties of our own data.

An accurate calculation of the h-index relies on an even partitioning of the upper and lower portions of the area under the curve captured by the metric *number of citations* for each paper sorted in descending order. The optimal partitioning as shown previously in Figure 11 is an equal area under the curve separated by the line at the origin, and results in the optimal h-index value. The h-index is the point at the intersection of the line with the curve, due to the linear nature of the relationship, most often the simple algorithm as demonstrated in algorithm 1 can be used to calculate the h-index. However, in the case of the bit-index, due to the nature of the power-law statistical distribution of our data, we used the non-linear methods proposed by Hirsch [78] as the basis for our metric. In order to accurately capture the bit-index for both outliers and values within the norm, the area under the curve given the ordering of the measurements must be calculated using integration such that each section partitioned by the intersection of the line for the bit-index has an approximately equal area.

The method can be applied to any measurement metric, for the h-index the preferred metric for measuring the performance of researchers is the number of citations per publication [78]; whereas in the case of the Bitcoin network the measurement can vary based on the feature of the network being studied. For the purpose of our hypothesis, the feature being measured is the gain, $\mathcal{G}$, which represents the gain in wealth for a user within the Bitcoin network. The gain is given in Equation 23 and is measured over the course of a specified period p, typically (day, week, month, quarter). It is the sum of the transaction amounts $\pm T$, (positive for BTC received, negative for BTC sent) divided by the total number of transactions n during that period. The gain metric describes the amount of wealth being accumulated by a user within each time period, we then use the gain at each time period to determine the optimal gain measurement that describes the user over the entire history of their transactions.

$$\mathcal{G}(p) = \frac{1}{n} \sum_{i=1}^{n} \pm T_i \tag{23}$$

The bit-index is calculated given the measurement and the period, such that the area partitioned by the line created by the intersection of the bit-index with the curve to the origin are equivalent. As shown in Equation 24, the gain measurements $\mathcal{G}$ for each period p over the history of the Bitcoin blockchain are sorted in descending order. The resultant sorted gain measurements $\mathcal{M}$ are then used to calculate the bit-index with respect to the change in each measurement until a intersection point from the origin is found that evenly partitions the left and right side of the area under the curve.

$$\mathcal{M}(\{p_1, \ldots, p_n\}) = \texttt{sorted}(\{\mathcal{G}(p_1), \ldots, \mathcal{G}(p_n)\}) \tag{24}$$

Given the measurements $\mathcal{M}$ from equation Equation 24, which are sorted in descending order, the calculation of the bit-index represented as i, as shown in Equation 25, is a value such that both side of the equation are approximately equivalent. The Left Hand Side (LHS) of the equation represents the area under the curve given by the ordered gain measurements $\mathcal{M}(p_j)$, where the area of the

triangle from the intersection point is subtracted from the total area. The Right Hand Side (RHS) represents the area under the curve for the measurements to the right of the intersection point, and it has the area that was subtracted from the LHS added. As such, the closer the approximation of both sides of the equation, the more accurate the value of the bit-index will be to describe the influence of the user.

$$\int_0^i \mathcal{M}(p_j)\,dj - \frac{\mathcal{M}(p_i)\cdot i}{2} \approx \int_i^n \mathcal{M}(p_j)\,dj + \frac{\mathcal{M}(p_i)\cdot i}{2} \tag{25}$$

Due to the limited number of measurements $\mathcal{M}$ as a result of the short existence of the Bitcoin network from the genesis block on 2009-01-03 [39] to time of writing; the area under the measurement curve must be calculated using a numerical approximation. This is due to the fact that the measurements are not a continuous series, but rather periodic measurements taken throughout the entire history of the user's transactions recorded on the blockchain. For its efficiency and simplicity of implementation, the numerical quadrature method of *trapezoidal rule* is used. The trapezoidal rule is widely used for numerical quadrature, and relies on creating a series of trapezoids under the area of the curve to approximate the resultant integral [80, 81]. This method provided consistent accuracy when calculated across all of the users, giving a bit-index that was representative of the influence of each user within the Bitcoin network.

3.2.1.3 *Bit-index Calculation Example*

In order to demonstrate the application of the bit-index metric, the following is a comprehensive example of the bit-index applied to one of the top wallets shown in Table 9. For our example, we have selected one of the Bitcoin exchange services, Bittrex[5], which was one of the top 25 wallets based on the number of addresses clustered in its wallet. The service has been in operation for the entire period of 2014-02-01 – 2016-02-01, we have calculated the monthly gain based on Equation 23, which divides the sum of transaction amounts (BTC) by the number

5 Bittrex is an alternative currency exchange for trading Bitcoin against many other clones. https://bittrex.com

of transactions. We see in Figure 12 that the monthly gain $\mathcal{G}$ never reaches beyond a maximum of approximately 0.06, and at times is also negative. The gain for each month is low, as there is a high churn in the average number of users sending their bitcoins to the service to exchange, and then withdrawing their bitcoins within the same month. The small monthly gain of the Bittrex service is indicative of the small influence that it exerts on the network, providing a service that allows others to exchange their bitcoins, while not accumulating significant wealth itself. We even see during the period of May 2015 to August 2015 a sharp decline in the monthly gain, that may be related to the dramatic price decrease of Bitcoin. During this period the price of Bitcoin stagnated and was at its lowest value since gaining attraction [82], likely leading many users to withdraw their bitcoins from the service and exit the market.

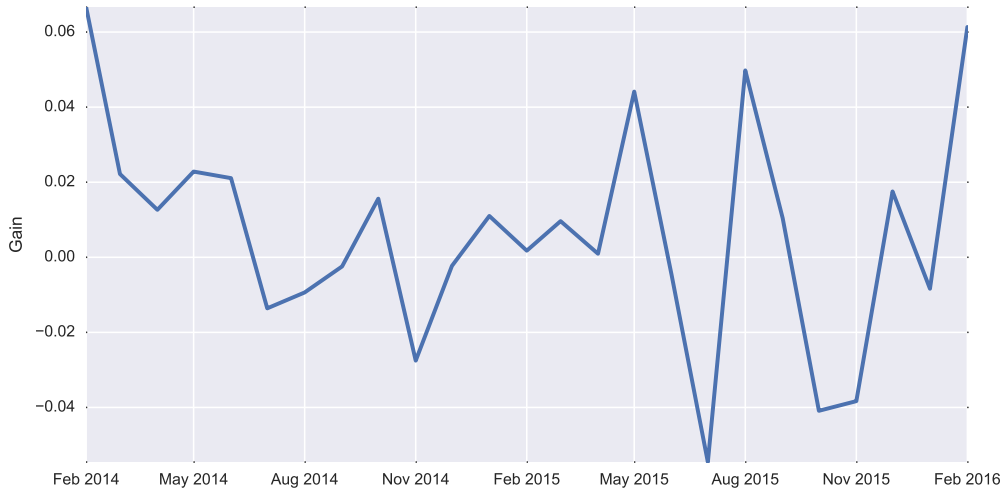


Figure 12: Bittrex Historical Monthly Gain

After calculating the monthly gain for Bittrex, the bit-index is then calculated as given in Equation 25 by sorting each monthly gain $\mathcal{G}$ measurement in descending order, and then finding the partitioning such that the area under the curve for both sides of the partition are approximately equivalent. We see in Figure 13 the partitioning applied to the Bittrex gain measurements, where the solid dashed line from the origin is the partitioning line. The point at which the line intersects with the curve is the bit-index calculated for Bittrex, a value of approximately 0.054, which is relatively low due to the high rate of transactions. Furthermore, the two areas highlighted in blue and red represent the LHS and RHS of Equa-

tion 25, and are approximately equivalent, with the small dashed lines representing the triangular area subtracted from the LHS and added to the RHS of the equation.

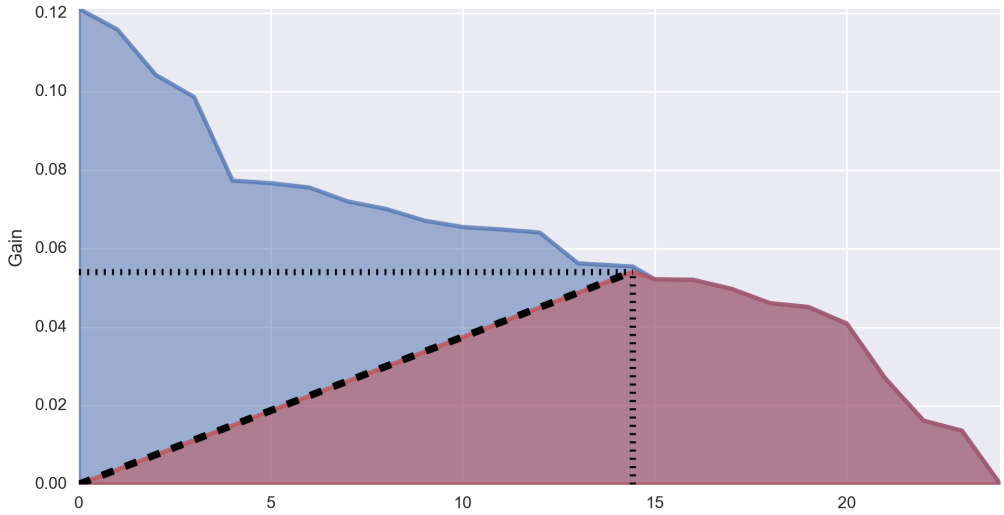


Figure 13: Bittrex bit-index Partitioning

A demonstration of the bit-index metric applied to several other users is given in Section A.2, where we see examples applied to several of the top 25 wallets based on number of addresses and balance. In Figure 37 we see the application of the bit-index to the Silk Road service, the largest online market for illicit goods, with a bit-index of 0.71. As well in Figure 39 we see a bit-index of 876.70 calculated for a user with 8,700 BTC accumulated over a period of five years, and one of the top wallets by balance listed in Table 10. Lastly, an example is given in Figure 41 of a user that is one of the top wallets by balance, that accumulated 21,744 BTC within a very short period of time from 2013-11-16 – 2016-02-03. Where even with only several monthly periods of gain measurements, the bit-index is accurately calculated as a value of 3,606.05 for the user.

3.2.1.4 *Bit-index: Summary*

Following the application of the bit-index metric to all of the users within the Bitcoin network, we evaluated the distribution of the gain $\mathcal{G}$ measurements aggregated for the entire network. As shown in Table 11, we see the descriptive statistics of the monthly gain and transactions per user for the network aggre-

gated yearly. The table is only for data prior to 2016, due to the limited amount of data for that year. It is significant to note that during the early years of Bitcoin's existence the gain was on average above one, however with each passing year since 2009 it has steadily decreased for each user, becoming negative after 2012. This result is insightful, and shows a similar trend of decreasing wealth per user demonstrated by other researchers [50], where the negative gain is showing that on average there is a tendency for a loss in wealth amongst the majority of users within the Bitcoin network. While the gain is steadily decreasing each year, the number of monthly transactions for each user is increasing yearly, reaching a stable point after 2012. This shows a continued interest and activity within the Bitcoin network, despite the overall trend of a loss in wealth for the majority of users.

	Transactions			**Gain**		
Period	**Mean**	**Median**	**Std.**	**Mean**	**Median**	**Std.**
2009	0.77	2.67	26.24	65.19	50.00	463.40
2010	4.67	2.00	111.66	11.47	0.00	302.96
2011	3.10	2.00	86.36	1.19	0.00	155.20
2012	5.48	2.00	675.89	0.45	0.00	57.07
2013	5.57	2.00	549.76	−0.03	0.00	34.17
2014	5.46	2.00	328.39	−0.01	0.00	24.37
2015	5.41	2.00	603.07	−0.04	0.00	10.37
Total	**5.47**	**2.00**	**515.58**	**0.07**	**0.00**	**41.90**

Table 11: Gain and Transaction Statistics

3.2.2 *Pareto Front Clustering*

The bit-index metric provides a way to quantitatively measure a representation of the wealth accumulated by a user relative to their number of transactions over their entire history of activity on the Bitcoin blockchain. However, the technique is limited to only one measurement, even for example when using h-index, it is

still applied to only the *citations* measurement [78, 79]. As such, we have evaluated the application of a second metric for identifying major network influencers based on the concept of a *Pareto front*, which allows for the ability to rank and select the best outcomes based on multiple metrics. We apply the Pareto efficiency technique on two different metrics of wealth accumulation, the first based on the user's balance of bitcoins, and the second $\mathcal{G}_{norm}$ based on the cumulative gain normalized to USD. After applying the Pareto efficiency technique to each user, we then compare the results of each technique in identifying the major network influencers.

3.2.2.1 *Pareto Front: Background*

A *Pareto front* or *Pareto frontier* is the inner or outer edge, depending if minimization or maximization, formed by the location of the optimal results given the relative trade-offs along the edge, where each of the optimal results is said to *non-dominated* by any other competing points. The Pareto optimization strategy is highly beneficial as it allows for the selection of the optimal choices based on the relative trade-offs of each parameter being minimized or maximized. The general convention is that either all objectives are being minimized, otherwise they are all being maximized. The Pareto frontier $\mathcal{P}_f(Y)$ is given in Equation 26 and can be formally described as follows. Where Y is the entire set of all criterion vectors y, consisting of the objectives $x \in X$, such that $Y = \{y \in \mathbb{R}^m : \ y = f(x), x \in X\}$ the Pareto frontier is made up each vector (point) y'' that is said to strictly dominate another point y' represented as $y'' \succ y'$, such that there are no other points that can be said to dominate y'' [83]. The final Pareto frontier is made up the set of all points y'' such that there are no other set of points (an empty set $\emptyset$), that are said to dominate y'' [83].

$$\mathcal{P}_f(Y) = \{y' \in Y : \{y'' \in Y : \ y'' \succ y', y'' \neq y'\} = \emptyset\}. \tag{26}$$

The Pareto frontier can often be best explained with a visual representation, we see in Figure 14 an example created to demonstrate the effectiveness of the Pareto method. In the figure on the left are a set of randomly distributed points

along x and y axes, each axis represents a measurement that is desired to be maximized, however there is not one objective, but two that must be maximized. We see by looking at the left that there are points which for both objectives are clearly greater in at least one objective and equal or greater to all other points within the other objective. We say that each of these points y'' that dominates any of the other equivalent points are the *non-dominated* points of our Pareto front. The final Pareto front as shown on the right half of Figure 14 is the set of all such *non-dominated* points as described in Equation 26. Given this visualization it is clear to see the strategy of maximizing the trade-offs of each objective, such that the final set of *non-dominated* points are presented as the frontier of the optimization problem.

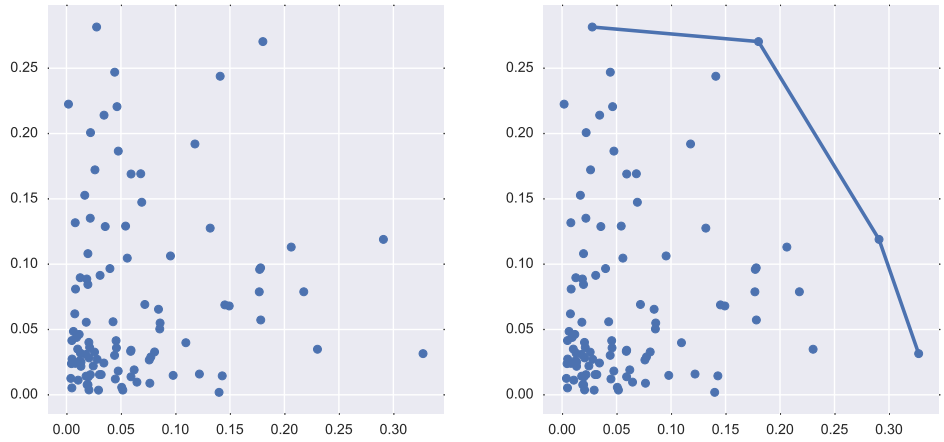


Figure 14: Pareto Frontier Along Non-Dominated Points

For our purposes we made one additional enhancement to the standard Pareto frontier, due to the power-law statistical properties inherent in the Bitcoin network, we have applied the concept of a delta, which considers points within the same delta as equivalent. We apply this technique in our application of the Pareto frontier, as there are many metrics where the difference between a fraction of a bitcoin or a small number of transactions are considered equivalent, as our intention is to identify only the major network influencers. The application of a delta produced much larger fronts consisting of many more users, rather than the precise separation of vectors used in the default application of Pareto optimization.

3.2.2.2 *Pareto Front Clustering Based on Balance*

We began by first clustering the major influencers based on their current balance, the number of days the wallet had been active, and the total number of transactions sent and received. Based on our objective in Section 1.4.2 we wish to identify the influential users within the network, whereby we define their influence as the amount of wealth that they have accumulated within the network. The Pareto front is generated in three dimensions and the front is the three dimensional surface along which the non-dominated points lie. In order to collect more users than just the first front alone, we repeat the process multiple times, each time excluding all of the points from all of the previous fronts. This technique we refer to as *Pareto front clustering*, whereby each front is considered a cluster of equivalently optimal outcomes.

For the process of creating each front a delta value is used for each parameter optimized, all values that are within the delta are considered equivalent, resulting in more values being selected for each front than would be otherwise. The following table Table 12 lists each of the features used in performing the Pareto front clustering in addition to the delta value used. As our primary concern is only identifying the major network influencers, it is beneficial to have less precision in separating each of the features, as it will result in more users being clustered in each front. As we are concerned with identifying the major network inlfuencers, our optimization criteria is to find the user with the *maximum* balance that have the *minimum* number of transactions and days active. This is so that we can identify any user that has accumulated wealth quickly, which we consider as having influence. We value the significance of minimizing *Days Active* to negate the effect of users that have been using Bitcoin since the initial creation, and have accumulated much wealth due to the currency having very low value for the first several years.

The Pareto front clustering based on balance was applied to all users with a balance > 0 BTC, in total we clustered $1,197,847$ users and generated the first 100 fronts as shown in Figure 15. Along the axes we see the features *Days Active*, *Balance*, and *Total Transactions*; where the balance is being maximized and the

Feature	Delta (Δ)	Maximize/Minimize
Balance	10	Maximize
Total Transactions	10	Minimize
Days Active	30	Minimize

Table 12: Pareto Front Clustering Parameters

remaining axes are being minimized. The figure shows the first 100 fronts, representing the 100 clusters of non-dominated users, transitioning from dark violet (front 1) to yellow (Front 100) with each successive front. The dark violet points are those that lie along the first initial fronts and are the most non-dominated. After we identify the points along the front, the points are then removed from the data and the Pareto optimization process is repeated on the remaining data to determine each successive front. We can see a clear separation of the users within the Pareto fronts and the remaining users plotted in blue, as each additional front is created the major influencers identified gradually moves closer towards the remaining users. Based on our own test we found that discontinuing the clustering process after 100 fronts produced the best results, where all of the users within the first 100 fronts have a significant balance of 100 BTC or more and demonstrating the accumulation of significant wealth.

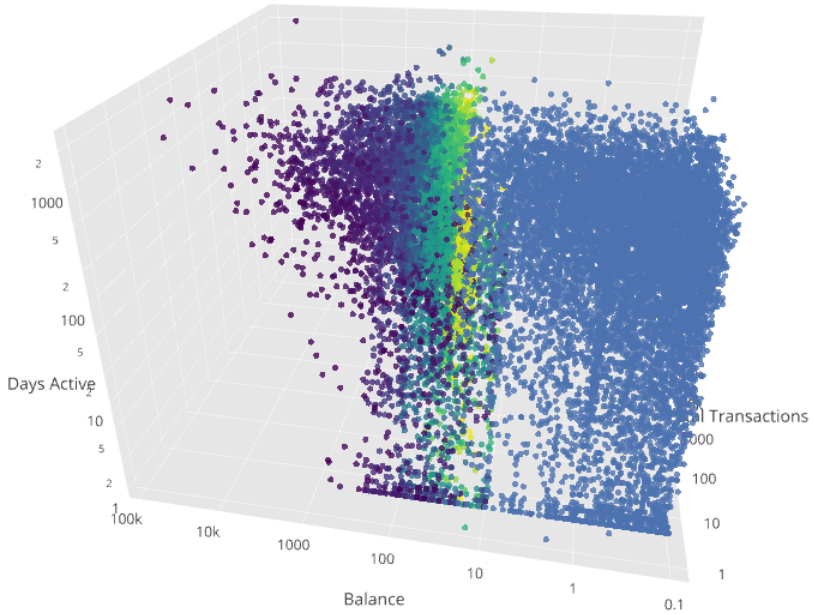


Figure 15: Pareto Clustering by Balance, 100 Fronts

3.2.2.3 *Pareto Front Clustering Based on Normalized Gain*

The Pareto fronts generated by maximizing balance and minimizing the transactions days active, produced clearly separate clusters for the major influencers from the remaining users within the Bitcoin network. However, while the days active and transactions are accumulative metrics across the whole history of each user on the blockchain, the balance is temporal and can vary depending on the time that the Pareto front clustering is performed. While the bit-index is based on the monthly balance and number of transactions performed by each user, and accounts for changing balances, the Pareto front clustering based on the final balance does not. Therefore, in order to assess this concern, we perform the clustering using another measurement for wealth accumulation, the cumulative normalized gain $\mathcal{G}_{norm}$, which is based on a user's total cumulative gain normalized to USD.

The wealth accumulation measurement for each user is defined as the cumulative normalized gain $\mathcal{G}_{norm}$ and is based on each user's total cumulative gain normalized to USD, and is given in Equation 27. We define their cumulative normalized gain as the sum of the amount of bitcoins for every transaction converted to the value in USD using the daily closing price provided by the CoinDesk Bitcoin Price Index [84]. Any transaction where bitcoins are received is a positive gain, and any transaction where bitcoins are sent is a negative gain; the cumulative sum of every transaction performed gives the final gain in USD. Unlike the gain measurements used for the bit-index, the $\mathcal{G}_{norm}$ is the sum of all transactions normalized to USD on the day the transaction occurred, and the results are not divided by the number of transactions performed by the user.

$$\mathcal{G}_{norm} = \sum_{i=1}^{n} \pm T_i \times \textit{Daily Close Price (USD)} \tag{27}$$

Similarly to the Pareto front clustering based on balance, there are three features being optimized, the cumulative normalized gain $\mathcal{G}_{norm}$, days active, and the total number of transactions. As well, a delta value is used for each, all values that are within the delta are considered to be equivalent. The delta for each

feature is given in Table 13, where we see that the delta for the normalized gain is 1,000, as the amount has been normalized to USD. This amount can be considered as a delta value of 1,000 USD, where each user is considered as having the same normalized gain in USD for their transactions.

Feature	**Delta** (Δ)	**Maximize/Minimize**
Normalized Gain ($\mathcal{G}_{norm}$)	1,000	Maximize
Total Transactions	10	Minimize
Days Active	30	Minimize

Table 13: Pareto Front Clustering Parameters

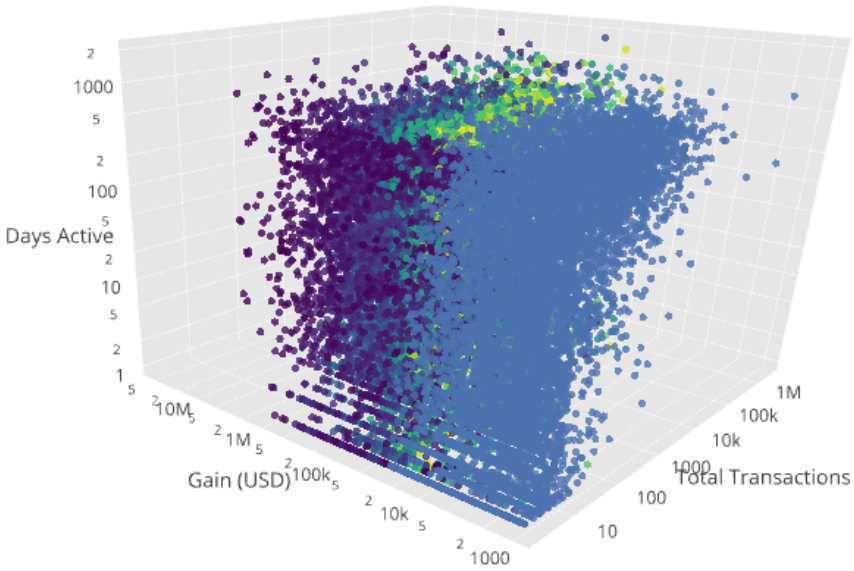


Figure 16: Pareto Clustering by Gain, 100 Fronts

For the Pareto front clustering, we used the cumulative normalized gain $\mathcal{G}_{norm}$ feature instead of balance and applied the clustering to all users within the network regardless of their current balance. Compared to the previous Pareto front clustering based on balance, where only the users with a balance > 0 were selected, we applied the Pareto front clustering to all 35,820,710 users regardless of their balance. The results of the Pareto front clustering for the first 100 fronts are shown in Figure 16. Again, we have three axes where the *Gain (USD)* is being maximized and the *Days Active* and *Total Transactions* are being minimized. Similarly to the previous three dimensional plot, the dark violet points are those that lie along the first initial fronts, and are the most non-dominated with each

successive front transitioning from violet to yellow. We see in the plot that the points are much more dense in comparison to Figure 15, due to the results of every user within the Bitcoin network being plotted. Furthermore, we see less of a clear separation between the first 100 fronts and the remaining data, instead seeing a more gradual transition with each successive front added. Lastly, we don't see any users with more than 10 - 30 million USD in normalized cumulative gain, likely due to the fact that in order to increase their wealth users had to similarly spend their bitcoins possibly on exchanges or other services.

3.2.3 *Major Influencers Selection Metric*

With both the bit-index and the Pareto front clustering techniques used to identify the major network influencers, we needed to select the optimal metric to use for our model creation. The metrics must identify the major network influencers which not only have accumulated significant wealth, but also influence the Bitcoin markets. We assess the users identified by each metric based on their involvement in services which are directly related to having an influence on the market. We found that while all metrics equally identified users that did have influence within the network, they did not all identify network influencers that were aligned with our hypothesis, that major influencers that use financial services related to Bitcoin can be used to predict the market price direction.

The following tables provide descriptive statistics of the top $2,000$ users based on their bit-index, as well as the users within the first 100 Pareto fronts for the Pareto front clustering based on balance and cumulative normalized gain ($\approx 2,000$ each. We see in Table 14 and for the Pareto front clustering results for balance in Table 15 that they are similar. Where the descriptive statistics of each shows a high balance and number of transactions, as well as a much longer period of activity in comparison to the users identified using the cumulative normalized gain measurement. The high number of transactions and the long period of activity could be indicative that the metrics based on balance have a tendency towards selecting Bitcoin related services, rather than our desired users, that are

network influencers increasing their gain using exchanges. Furthermore, we see in Table 16 that the number of transactions are significantly lower compared to the users found based on balance. With a difference of 11,200–14,344 transactions to just 28 for users identified based on the cumulative normalized gain. The much lower number of transactions for users identified based on gain is indicative that the users selected are likely more reflective of individuals within the Bitcoin network rather than services.

Metric	Summary		
	Mean	**Median**	**Std.**
Balance	8,642	2,323	18,994
Transactions	14,344	14	86,313
Volume	142,212	1,965	533,543
Days Active	341	132	423

Table 14: Bit-index Summary

Metric	Summary		
	Mean	**Median**	**Std.**
Wallets in First Front		46	
Balance	7,442	768	17,464
Transactions	11,200	8	73,099
Volume	103,014	958	517,725
Days Active	286	94	439

Table 15: Pareto Front Balance Summary

Metric	Summary		
	Mean	**Median**	**Std.**
Wallets in First Front		83	
Normalized Gain (USD)	3,613,620	9,143,080	244,680
Balance	1,199	0.000055	7,945
Transactions	28	6	92
Volume	76,954	5,580	343,589
Days Active	195	91	216

Table 16: Pareto Front Gain Summary

For both of the Pareto front clustering methods we did not see any significant change in the number of users for each front as the number of Pareto fronts increased. For both methods we see a similar trend that the number of users within each front increased and decreased respectively according to the current front generated. This outcome differed from our initial assumptions, that the number of users in each front would increase with each consecutive front created. We see the number of users in each front based on normalized gain in Figure 43 increase from an initial 83 users in the first front a maximum of 200 users. For each front based on the normalized gain the average number of users is nearly double that of the Pareto fronts for users based on balance. For a comparison the complete results can be found in Section A.4, where the number of users in each Pareto front is given for both the balance and normalized gain.

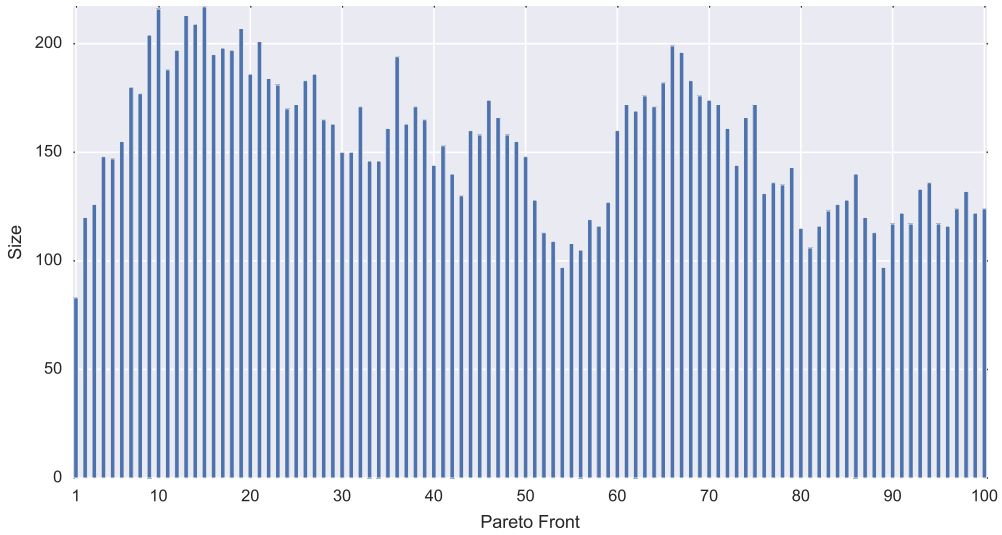


Figure 17: Pareto Fronts by Balance Sizes

Service Type	bit-index	Pareto Front Balance	Pareto Front Gain
Exchanges	24%	30%	76%
Gambling	32%	25%	0%
Services/others	20%	16%	12%
Old/historic	14%	16%	8%
Darknet/market	8%	10%	0%
Pools	2%	3%	4%

Table 17: Percentage of Users Using Services By Metric

Our final decision on the appropriate metric to use for the feature engineering necessary to create the predictive model was made based on which metric most accurately identified individuals rather than services. Based on the results of Table 14, there is a significant discrepancy between the descriptive statistics for the bit-index and Pareto front clustering based on balance, compared to the Pareto front clustering based on normalized gain. For the bit-index and balance results the users on average have a much higher balance than those of gain, with the number of transactions vastly higher ranging from 11,200–14,344 transactions compared to just 28 for gain. A value of 28 for the number of transactions performed by users is much more realistic than a value ranging from 11,200–14,344, which is more indicative of Bitcoin related services that individuals. Lastly, as shown in Table 17 we see a clear breakdown of the services used by the top ranked 2,000 individuals identified using each metric, for obvious reasons all users that are already labelled as Bitcoin services are excluded. For those users identified by the bit-index and Pareto front clustering by balance, we see a fairly even distribution of use amongst the various types of Bitcoin related services. However, for the users identified by the Pareto front clustering using gain, we see a much higher usage (76%) of exchanges used amongst users in this front. Thus, given the realistic balances and number of transactions, as well as the predominant tendency to use exchanges for users identified by the metric, we chose to use the Pareto front clustering by cumulative normalized gain. This metric provides the most accurate identification of network influencers that accumulate wealth through the use of exchanges, and as such are the users whose features we wish to extract given our hypothesis.

3.3 PREDICTIVE MODEL CREATION

The predictive model creation is the culmination of all of the prior steps outlined in the methodology, the pre-processing was essential in order to gather the data and present it in a usable form for predictive modelling. The two metrics for identifying major network influencers, the bit-index and Pareto front clus-

tering are also essential to identifying the wallets owned by users which have significant influence on the network. Where the Pareto front metric based on normalized gain in particular identified those users whom frequently interact with exchange services. Our predictive model creation process involves extracting the pre-processed features from the database aggregated daily, including Bitcoin network and market data, as well as the gain for each wallet within the first 100 Pareto fronts. We then feature engineer the data and use it to evaluate various models based on those researched in Section 2.1.1, and applied by other researchers to predict the market in Section 2.2.3. After creating and evaluating the potential of several models the final model selection is conducted using a set of holdout experiments, where a portion of the data is used for training and the rest is set aside for testing. The final model is selected based on the performance of each when applied to the holdout test data, each model is compared using several metrics which are selected based on our literature review in Section 2.1.2.

3.3.1 *Feature Engineering*

Prior to performing feature engineering the pre-processed data was first extracted using queries on the database, each of the attributes pertaining to the Bitcoin network and the gain calculated for each of the influencers was aggregated daily. In addition to the network features, the market data was aggregated for all of the major exchanges listed in Table 1; for each exchange the price and volume were aggregated by the hour so that features could be engineered based on the market trends over the course of a 24 hour trading period each day.

The extracted data was then transformed from row based to columnar based, row based format is the default used by database systems, whereby the results of the queries returned the features for each wallet as additional rows. Therefore, in order to be properly feature engineer the data it was transformed to columnar based, where each wallet received a unique column and there was only a single row for each date of data. Transforming the data from row based to columnar required performing a transformation on the data, to facilitate this we used

the software library `pandas` [85], which provides columnar data frames that are stored in memory rather than disk for fast access. The process of transforming the data from row based to columnar is demonstrated in an example in Table 18, in row based format there are duplicate rows for each date anytime a wallet has a change in gain on that date. For the predictive models, the data must only have a single row for each date, as such the process of converting to columnar based as shown in Table 18 consisted of creating a column for each unique wallet found in the data. After creating a column for each unique wallet, the data is transformed so that for each date the change in gain for the wallets is set in the column rather than by duplicating rows.

Date	Wallet	Gain
2012-01-01	1	100
2012-01-01	2	10
2012-01-02	3	25
2012-02-03	1	50
2012-02-03	3	15

Row Based

Date	1	2	3
2012-01-01	100	10	0
2012-01-02	0	0	25
2012-01-03	50	0	15
2012-01-04	0	0	0
...			

Columnar Based

Table 18: Row Based to Columnar Based Transformation

Following the transformation of the data from row based to columnar the Bitcoin market data was also feature engineered to extract details based on the daily price trends, rather than just an aggregation for the entire day. Volatility is a valuable metric in understanding the risk in a financial market and was applied by Bukovina, Marticek, et al. [62] to the Bitcoin market specifically. We calculate the market price volatility based on a moving average of the normalized USD price over the course of the 24 hour trading period each the day, starting at midnight and ending at midnight on the start of the next day. The equation used for the average daily price volatility calculation is given in Equation 28, where V_{daily} is calculated based on the average of a rolling standard deviation $\sigma_{i,\,rolling}$. For each hour of the day the rolling-window standard deviation is calculated over the previous 24 hours leading up to that hour. The final average daily price voalitily is then calculated by taking the sum of all rolling-window standard deviations divided by 24, the hours in each trading day.

$$V_{daily} = \frac{1}{24}\sum_{i=1}^{24} \sigma_{i,\,rolling} \cdot \sqrt{24} \tag{28}$$
$$= \frac{1}{24}\sum_{i=1}^{24} \left(\sqrt{\frac{1}{i}\sum_{j=1}^{i} (x_{j-24} - \mu)^2} \right) \cdot \sqrt{24}$$

In addition to the daily volatility, the closing change in price was considered for each day, where the closing change in price is referred to as *price close*, which is calculated as the difference in price at the end of the day compared to the start, $(price_{end} - price_{start})$. The remaining market features *trading volume* and *average price* are market aggregations representing the total trading volume each day across all major exchanges, and the average normalized price of Bitcoin in USD over the 24 hour daily trading period. Once we completed the market data feature engineering, it was then combined with the wallet gain features into a single data frame, for reference the list of all features are shown in Table 19. All of the features were used both for our training data, as well for the test data throughout the creation, optimization, and evaluation of our predictive models.

Feature	**Description**
Volatility	The average daily Bitcoin market price volatility calculated based on hourly price changes.
Trading Volume	The total trading volume normalized to USD for all major exchanges.
Price Close	The change in price between the end and start of each day.
Average Price	The average price normalized to USD over the 24 hour trading period each day.
Wallet Gain	The daily gain metric calculated for each of the wallets within the first 100 pareto fronts.

Table 19: Model Features

After combining all of the features into a single data frame we then assessed the sparsity of the data, which poses a challenge for many predictive models where the data is expected to have a value for every column in each row. The sparsity of the data is due to the fact that on any given day only a small fraction of the major influencers are performing any transaction, thus altering their gain.

We find that the majority of the dates within our data, nearly all of the major influencers had no change in gain as they had performed no transactions. We calculate the sparsity of the data as the sum of the empty rows divided by the total dimensions of the data, we calculated in total $2,171,918$ empty data entries out of a total for the dimensions of the data $rows \times columns = 2,335,144$. Based on our calculations the matrix has a sparsity of approximately 93%, which is high, implying that only 7% of the entire data frame used in our models contains features.

Normalization is a standard step when feature engineering data, however due to the high sparsity of our data we had to use a specialized data normalization method to ensure that the sparsity of the data is preserved after normalizing it. Normalization is required prior to training and evaluating models to ensure that all of the features are scaled appropriately and display properties of a standard normal distribution with a mean, $\mu = 0$ and standard deviation $\sigma = 1$ [7]. Performing normalization is important to ensure that all features have an appropriate influence regardless of the original numerical scale of their values compared to other features. The process of data standardization, often referred to as Z-score normalization, is given in Equation 29, where X represents the data entry, μ the mean of the data, σ the standard deviation, and X' the scaled feature. The Z-score normalization is applied to the data such that the resultant new features are centered around 0 with a standard deviation of 1 [7, 14]. For our data we used a variation of the standardization method known as the *max absolute scaler*, which is desiged to ensure that all zero entries in the sparse matrix remain as zero. The equation for this normalization method is given in Equation 30, where the maximum of the absolute values of the maximum and minimum X_{max}, X_{min} are used to scale the data [7, 14].

$$X' = \frac{X - \mu}{\sigma} \tag{29}$$

$$X' = \frac{X}{Max\left(|X_{max}|,\ |X_{min}|\right)} \tag{30}$$

We applied the *maximum aboslute scaler* to our data in order to ensure that all of the data was standardized, while at the same time preserving the sparsity of the data. We used the implementation `MaxAbsScaler` provided by the `scikit-learn` software library [86] to perform the standardization, resulting in the standardized data still maintaining the original 93% sparsity. Combined the total feature dimensions of the data are $2,072$, with $2,068$ of the feature columns belonging to the gain features representing each of the wallets within the first 100 Pareto fronts.

As part of our feature engineering processing we attempted to reduce the number of features from $2,072$ to only those which describe the variances in the data. The significance of the features are demonstrated in Figure 18, where each of the numerical labels are the wallet identifiers correseponding to the gain metric for that wallet. All of the wallets from the first Pareto front were significant depending on the day as the data was predominanly sparse, although in many cases only a small subset of all wallets performed transactions on any given day. The ranking on the left is out of a maximum score of 100, meaning the feature is significant to the final outcome of the predictive model for 100% of the data. In the case of price close, with a score of 42, it is still highly significant, as it implies that the price trend (upwards or downwards) of the current day has an influence on the price direction (`UP/DOWN`) of the following day.

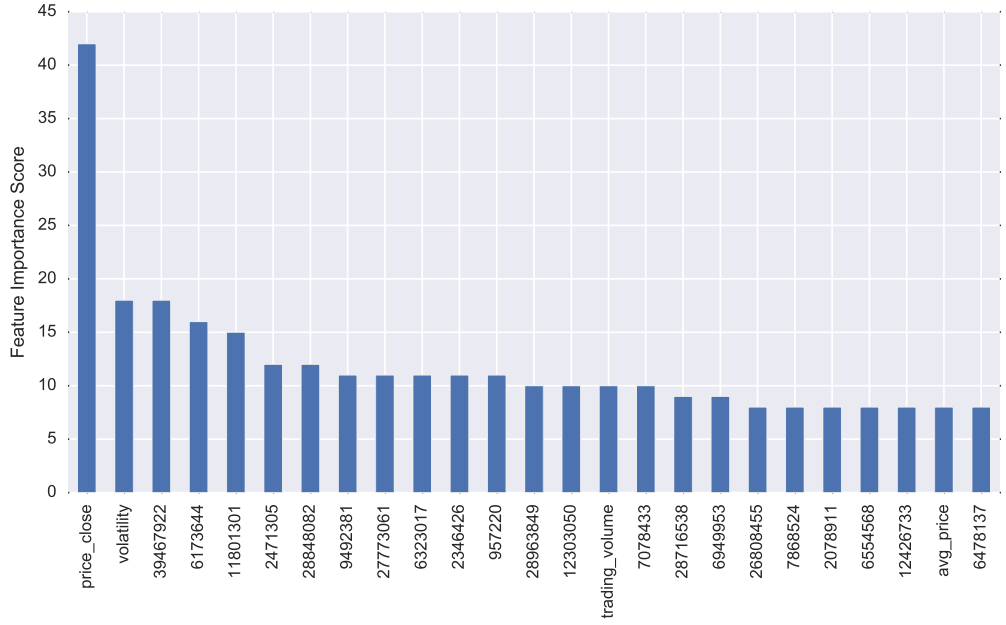


Figure 18: Features Ranked by Importance

While the top wallets repesented in Figure 18 have a significance of appoximately 5%–15%, and are significant to an extent in explaining a portion of the data, they still do not describe the influence of all major network influencers. Based on our evaluation, using the wallets from the first 100 Pareto fronts captures the majority of users within the network that have a possible influence on the network. Any attempt we made to reduce the number of wallets used as features had a negative impact during model evaluation as less of the variances of the data could be explained. All of the features are needed due to the high sparsity of the data, because on any given day only a small fraction of the users that have influence on the network are performing transactions, thus altering their gain within the Bitcoin network.

After finalizing the features and applying the *max absolute scaler* normalization to the data, we established the known predictive outcome to be used for training and testing of the models. As stated previously in Section 1.3, our hypothesis is to create a model that can predict the future daily price direction (`UP/DOWN`) of Bitcoin based on the actions of major network influencers with greater accuracy than random chance. We define the *future price direction* for each day by calculating the average price of the future day over the 24 hour trading period

and comparing it to the average price of the current day. An outcome where $price_{future} > price_{current}$ would result in a binary value of 1 representing direction `UP`, the opposite outcome is encoded as a binary value of 0, representing a direction `DOWN`. Lastly, it is important to consider that we had no instances where the average price did not change, in all instances the average price for the following day either went `UP` or `DOWN` relative to the average price of the current day.

3.3.2 *Model Evaluation*

The model evaluation process involves the creation of several predictive models based on extensive literature review of predictive models used in binary classification problems discussed in Section 2.1.1. We created and evaluated four models: non-linear, SVM, decision trees, and XGBoost; for each model we not only evaluate the performance but also the challenges in hyper-parameter optimization, and the robustness to various train and test splits. After evaluating each model, the final model for our experiments was selected based on its performance and robustness relative to all of the other models created using a combination of numerous metrics.

For each of the models we performed hyper-parameter optimization for the three most significant parameters responsible for achieving the optimal results. In each case the performance of the models greatly improved after selecting the optimal parameter values, the non-linear model did not require hyper-paramater optimization as there are no parameters. The detailed list of each parameter and the relevance to the model are shown in Table 20, the detailed list of applicable values optimized for each model are given in the associated sections for each model. Every model was optimized using a similar range of values for each parameter, giving each a unbiased range of values to be exhaustively evaluated using the Grid Search algorithm.

Model	Parameter	Description
SVM	`C`	Penalty parameter of the error term
	`kernel`	Kernel used: linear, poly, sigmoid, or rbf
	`gamma`	Coefficient used for kernel functions
Decision Tree	`splitter`	Strategy to choose the split at each node
	`max_features`	Number features when choosing best split
	`max_depth`	Maximum tree depth for base learners
XGBoost	`learning_rate`	Learning rate for each boosting round
	`max_depth`	Maximum tree depth for base learners
	`subsample`	Fraction of observations randomly sampled for each tree

Table 20: Model Parameters Optimized

3.3.2.1 *Non-Linear Model*

The non-linear model was developed using the proprietary software solution Eureqa, which is advertised as an automated artifical intelligence powered modelling engine, that uses evolutionary algorithms to automatically create accurate predictive models [87, 88]. The Eureqa software can be described as a "robot scientist", an automated software solution that can create machine learning models using proprietary evolutionary strategies to derive models and equations for describing the hidden mathematical relationships in the data [87]. We used the Eureqa software to automatically build a non-linear predictive model using the first half of the data (50%) to classify the future price direction. In total over a week of CPU time was dedicated to discover the final non-linear model, we had to make numerous adjustments to the data provided to the software in order to produce modest results.

Initially we attempted to use the entire training data of 2,072 features to create the model, but found that the software could not handle the high-dimensionality of the data within a reasonable amount of CPU time. The academic version of the software that we were provided was restricted to 1 CPU core and as such we had to limit our training data to the 100 most significant features so that a model could be discovered within a modest amount of CPU time. Even after reducing

the number of features for the data, the software still took approximately 1 week of CPU time to discover a model with better than 50% accuracy.

The following is the non-linear model discovered using Eureqa given in Equation 31, where X_{wallet} is the gain for a specific labelled wallet. As well, where $SMM(x, n)$ is the simple moving median of the feature x over the past n entries, and $SMA(x, n)$ is the simple moving average of the feature x over the past n entries, and lastly $Delay(x, n)$ is the feature x from n rows prior. Each of the variables in the non-linear equation are multipled by an arbitraty coefficient, with the final parameter as a fraction involving the $price_{close}$, $Delay$, SMA, and the gain of a specific wallet. Given the solution Y to the non-linear equation, the final direction is determined by applying a *Logistic* function, which produces a numeric result between $[0, 1]$. Lastly each of the direction values from the logistic function are rounded to give a final value of 0 or 1, where any direction value > 0.5 is assigned 1 for `UP` and otherwise assigned 0 for `DOWN`.

$$
\begin{aligned}
Y = {} & X_{15090019} + 13.7631919552486 \cdot X_{915804} \\
& + 11.142939926681 \cdot SMM(X_{9025801},\ 15) \\
& + 3.28683604376165 \cdot Delay(X_{16970637},\ 4) \\
& + 2.30360200999047 \cdot X_{13408047} \\
& + 1.7563489850896 \cdot X_{13341271} \\
& + \frac{price_{close} - Delay(X_{25304929},\ 1)}{SMA(volume,\ 18) - X_{7914913}} \\
Direction = {} & Logistic\,(Y)
\end{aligned}
\tag{31}
$$

While the final model created with Eureqa performed well on the training data we used (50%), we noticed over-fitting when we performed simple tests with the remaining 50% holdout test data. Due to the limitations of the academic version of the software to only 1 CPU core it was not possible to easily create and experiment with new models. The limitations of the academic version of the Eureqa software made it impractical to experiment with and generate more robust mod-

els. Furthermore, the time required to create moderately accurate models took an order of magnitude longer than any of the other models.

3.3.2.2 *Support Vector Machines*

The second model we evaluated was SVM, based on our literature review in Section 2.1.1.1. With a SVM the data is represented as a series of points within a higher dimensional space, the points are then separated into categories by defining a hyperplane which attempts to maximize the gap between the two classes of points to be as wide as possible. We performed hyper-parameter optimization for the SVM parameters using exhaustive Grid Search, which evaluates all possible combinations of parameter values with cross-validation to select the optimal values. The detailed list of values optimized using Grid Search are provided in Table 21, where each of the parameters and values to test were selected for their primary role in influencing the predictive performance of the model.

Parameter	**Significance**	**Values**
`C`	Penalty parameter of the error term	[0.001, 0.01, 0.1, 1, 10, 100, 1000]
`kernel`	Kernel function for mapping the data	[`linear`, `poly`, `rbf`, `sigmoid`]
`gamma`	Kernel function coefficient	[0.1, 0.01, 0.001, 0.0001, 0.00001

Table 21: SVM Parameter Values

We used exhaustive Grid Search to test all possible combinations of parameter values, giving a total of 140 possible combinations, for each evaluation cross-validation was performed using 5-fold validation. While we found that Grid Search succeeded in finding the optimal parameters when using the training data, we often found during our initial evaluations that the SVM would suffer from over-fitting and perform poorly when applied to the test data that was withheld (holdout). Furthermore, if the amount of available training data was increased or decreased the model would require completely being re-calibrated, each time with vastly different `C` and `kernel` parameter values, demonstrating that the model inherently is not robust. We performed Grid Search hyper-parameter

optimization several times, reducing the amount of training data in each experiment until we established a model that consistently performed better than random.

Overall, the choice of SVM posed numerous issues during our model evaluation, we also theorize that the sparsity of the data could have a negative impact on the performance of the model. During our literature review in Section 2.1.1.1 we were unable to find examples of SVM being applied to highly sparse data, however it has been consistently used as a predictive model for binary classification tasks.

3.3.2.3 *Decision Trees*

Following our evaluation of SVM as a potential predictive model for the experiments, we proceeded to evaluate the effectiveness of decision trees. Based on our literature review in Section 2.1.1.2, we found that decision trees are consistently used as predictive models in both binary and multi-class classifier problems. Furthermore, they have the added benefit of providing a visual representation of the decision tree that is used to classify the data into each category. We found this aspect of decision trees extremely helpful, as it would demonstrate the logic behind the decision process. In addition, decision trees are also *non-parametric*, whereby the parameters and rules used in the model would be discovered based on the statistical properties of the training data provided [14]. The complete list of parameter values optimized are provided in Table 22, each of which was selected based on their importance in affecting the performance of the model.

Parameter	**Significance**	**Values**
`splitter`	Strategy when splitting each node	[`best`, `random`]
`max_features`	The number of features when choosing best split	[2, 4, 6, 8, 10, `auto`, `sqrt`, `None`]
`max_depth`	The maximum depth of the tree for learners	[2, 4, 6, 8, 10, 12, 14, `None`]

Table 22: Decision Tree Parameter Values

Similarly to with SVM, we performed exhaustive Grid Search and tested all possible combinations of parameter values, in total evaluating 128 possible combinations. For each evaluation cross-validation was used with 5-folds for the validation, while we noticed that the decision trees were more stable in the final choice of parameter values, the greatest difficulty was that the resultant decision trees often were extremely complex and large. While the decision trees captured the majority of features, they often suffered from over-fitting when evaluated on the test data. In each evaluation we determined that the resultant `max_features` parameter value optimized using Grid Search was always *None*, which puts no restrictions on the decision tree, allowing it to select all of the available features when generating the tree.

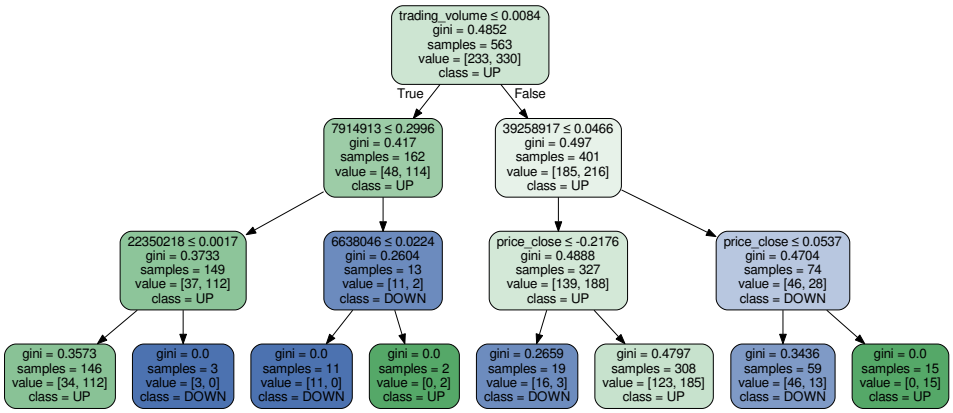


Figure 19: Decision Tree Model

We were able to reduce the negative impact of over-fitting with the decision trees by consequently applying our own choice of parameter value for `max_depth`. The default value often found by Grid Search when applied to the training data was *None*, making no restrictions on the height of the tree. While allowing for the creation of large trees produced exceptional results on the training data, the model was often over-fitted when evaluated against the test data that was withheld (holdout). We manually optimized the list of parameter values for `max_depth` and found the optimal tree height to be 4, allowing for good accuracy when applied to the training data, as well as reducing the effect of over-fitting when evaluated on the test data.

Despite applying an optimal tree height parameter, we still had issues in the non-deterministic tree generation used by decision trees, the resultant tree shown in Figure 19 represents one of the best performing models. However, in many future evaluations when using the same parameters different decision trees would be produced with inferior performance. The non-determininstic aspect of decision trees is inherent to their very nature, for each execution a random number generator is used as part of the algorithm, which produces different resultant trees. Thus, in order to make our results for the final model selection consistent we specified a default seed value of 0 for the random number generator, which resulted in a deterministic tree being generated each time.

3.3.2.4 *XGBoost*

The final model we evaluated was XGBoost, based on our literature review in Section 2.1.1.3, XGBoost has consistently been one of the strongest performing machine learning models. It has been used to win several competitions in machine learning, winning against many other competing models [23–25]. Naturally, we decided that it would be beneficial to our own research to evaluate the effectiveness of XGBoost as a predictive model for binary classification tasks. Similarly to all other models, we chose the most important parameters and appropriate values as shown in Table 23 to optimize for XGBoost. In all of our experiments conducted we trained the model using only 50% of the data, with the remainder being the holdout that was used for our preliminary evaluation of the model.

Parameter	**Significance**	**Values**
`learning_rate`	Learning rate for each boosting round	[0.1, 0.2…1]
`max_depth`	Maximum tree depth for base learners	[4, 6, 8, 12]
`subsample`	Fraction of observations randomly sampled for each tree	[0.25, 0.5, 0.8, 1]

Table 23: XGBoost Parameter Values

Similarly to the previous models, we performed exhaustive Grid Search and tested all of the possible combinations of parameter values. In total there were 144 combinations of parameters evaluated, similar to the number of possible com-

bination of parameters evaluated for the previous models. We used the same method of cross-validation with 5-folds used for validation, immediately we noticed that Grid Search produced a similar set of optimal parameter values when executed with varying sizes of training data. Furthermore, during our initial evaluations the issues of over-fitting that affected decision trees appeared to be mitigated as a result of the ensemble method applied in XGBoost. By default XGBoost uses a combination of many estimators, providing more robustness than a single model, while ensuring that any single tree does not become overly complex. For our implementation we used the standard XGBoost Python package provided by the creator Chen [22], which used a total of 100 estimators for the ensemble when generating the model.

Overall, XGBoost during our preliminary evalulations proved to be a more robust solution and was not as susceptible to overfitting as the other models we evaluated. Furthermore, it provided similar predictive performance when applied to holdout test data, which confirmed that our model was not over-fitted to the training data. While XGBoost performed well in comparison to our preliminary evaluation, the final model selection process will combine the results of several metrics in order to determine the most accurate and robust predictive model for the final set of experiments.

3.3.3 *Model Selection*

After creating and evaluating all of the proposed models we selected the final model to use for our experiments using a combination of widely used metrics for evaluating binary classifiers. As part of our literature review in Section 2.1.2 we assessed each of the metrics and found that each had strengths and weaknesses, thus in order to provide the best selection possible we evaluated and compared each of the models against all of the metrics. The following are the metrics used to assess the predictive performance of each model: *precision*, *recall*, *f1-score*, and *ROC*. Each of these metrics provides a value between $[0, 1]$, where 1 represents a perfect score and 0 a completely incorrect score. For the final model selection we

also assess the robustness of each model, where we are evaluating the predictive performance of the model as the amount of training data increases and the test data (holdout) is decreased. During our model evaluation we established that over-fitting was a common issue for nearly every model, and setup the experiments in order to assess the robustness of each model to over-fitting.

Every model was optimized using Grid Search, and in many cases the parameters were further optimized by hand during our model creation and evaluation to ensure that the optimal parameters were chosen. For each model, all of the paramater values seleced were chosen such that they give the model strong predictive performance, while not also succumbing to over-fitting. The optimal parameters for each model can be found in Table 24.

Model	**Parameter**	**Optimized Value**
SVM	`C`	1.0
	`kernel`	`linear`
	`gamma`	0.005
Decision Tree	`splitter`	`random`
	`max_features`	`None`
	`max_depth`	4.0
XGBoost	`learning_rate`	0.1
	`max_depth`	5.0
	`subsample`	1.0

Table 24: Optimal Model Parameters

A decreasing amount of test data was set aside as the holdout to evaluate the performance and robustness of each model; we compared the performance of each model using 50%, 30%, and 10% holdouts. For each holdout experiment all four model were evaluated and compared against each other using the four metrics. The performance of each model in predicting either the `UP` or `DOWN` future price direction was averaged, except for ROC AUC, which is a single numeric valute representing the performance of the model in predicting both outcomes. The range of the entire data used for experiments spans the period of 2013-01-01 – 2016-02-01 with a total of 2,072 features as described in Section 3.3.1, in each of

the experiments a percentage of the data was removed and withheld (holdout) from training and used for evaluating the models.

For brevity, only the results of the 50% and 10% holdout experiments are listed below, the complete details for all experiments used to assess each model can be found in Section A.5. The results of the 50% holdout experiments are given in Table 25, the training period for the models was from 2013-01-01 – 2014-07-17, with the remaining data as holdout to be used for testing spanning from 2014-07-18 – 2016-02-01. The 50% holdout test data was used to evaluate the performance of each of the models, we see that nearly all of the models evaluated performed better than random, with the exception of SVM, which performed approximately equivalent to random. As noted previously during our model evaluation we had numerous challenges with the SVM model due to its sensitivity to parameter changes when evaluated on varying amounts of training data. These same challenges with SVM are evident in the peformance for the experiments being equivalent to a uniform random outcome. As well, it is significant to mention that all of the other models performed better than random, with the non-linear model created using Eureqa achieving a performance similar to that of other Bitcoin price direction predictive models [72, 73]. While the Eureqa model achieved better than random, both the decision trees and XGBoost models achieved a significant margin better than random, with the decision trees model receiving a score for each metric between 0.58–0.60, and XGBoost receiving between 0.64–0.74. These results are consistent with our initial evaluations, where after limiting the maximum depth of the decision trees there was less over-fitting, resulting in a lower score during training but a higher overall score during testing. XGBoost received the highest score in every metric for the 50% holdout tests, demonstrating the same level of performance and robustness as during our initial evaluations, lending more credibility to it's strong performance as a predictive model [23–25].

The 10% holdout experiments assessed not only the performance of each model but also the robustness of each model to over-fitting. For the 10% holdout experiments, 90% of the available data was used for training, spanning the period of 2013-01-01 – 2015-10-11, with the remaining 10% holdout data used for test-

Model	Prediction	Precision	Recall	F1-Score	AUC	Support
Eureqa	0	0.56	0.40	0.47		295
	1	0.50	0.65	0.56		269
	Avg / Total	**0.53**	**0.52**	**0.51**	**0.53**	**564**
SVM	0	0.52	0.43	0.47		295
	1	0.47	0.56	0.51		269
	Avg / Total	**0.50**	**0.49**	**0.49**	**0.50**	**564**
Decision Tree	0	0.63	0.49	0.55		295
	1	0.55	0.68	0.61		269
	Avg / Total	**0.59**	**0.58**	**0.58**	**0.60**	**564**
XGBoost	0	0.70	0.54	0.61		295
	1	0.60	0.75	0.66		269
	Avg / Total	**0.65**	**0.64**	**0.64**	**0.74**	**564**

Table 25: 50% Holdout Model Evaluation Results

ing, spanning from 2015-10-12 – 2016-02-01. In comparison to the 50% holdout experiments the 10% holdout experiments had significantly less days available for testing. For the 50% holdout experiments there were 564 days to evaluate the predictive performance of each model, whereas in the 10% holdout experiments there were only 113. With less available data for testing and more data used to train the models the negative effects of over-fitting become more evident, we see in Table 26 the results of the 10% holdout experiments. The Eureqa model achieves varying performance of 0.47–0.62, however the two key metrics which evaluate the overall performance, F1-score and AUC give it a score of 0.47 and 0.54, slightly better than random. The SVM achieved approximately the same score across all holdout experiments, being almost equivalent to that of pure random; the decision trees performed the worst out of all models with a score between 0.44–0.53, affirming our challenges with over-fitting during the model evaluation. Lastly, the performance of the XGBoost model was significantly better than all of the other models evaluated, receiving a score for each metric between 0.72–0.78, demonstrating an even greater level of performance and robustness than in the 50% holdout experiments.

Model	Prediction	Precision	Recall	F1-Score	AUC	Support
Eureqa	0	0.70	0.13	0.22		54
	1	0.54	0.95	0.69		59
	Avg / Total	**0.62**	**0.56**	**0.47**	**0.54**	**113**
SVM	0	0.49	0.44	0.47		54
	1	0.53	0.58	0.55		59
	Avg / Total	**0.51**	**0.51**	**0.51**	**0.48**	**113**
Decision Tree	0	0.47	0.15	0.23		54
	1	0.52	0.85	0.65		59
	Avg / Total	**0.50**	**0.51**	**0.44**	**0.53**	**113**
XGBoost	0	0.73	0.65	0.69		54
	1	0.71	0.78	0.74		59
	Avg / Total	**0.72**	**0.72**	**0.72**	**0.78**	**113**

Table 26: 10% Holdout Model Evaluation Results

The ROC plots and their associated AUC provide a comprehendable visualization of the performance of binary classifier predictive models, and are frequently used to compare the performance of models [28, 35]. We use the ROC plot to provide a final visual assessment of the performance of each model in addition to each of the metrics used in Table 25 and Table 26. In each of the plots there is a dashed diagonal line, known as the *line of no discrimination*, dividing the lower and upper portion of the plot, where any points that lie above the line are considered better than perfectly random [35]. The ROC plot for the results of the 50% holdout experiment are shown in Figure 20, here we clearly see the XGBoost and decision trees above the *line of no discrimination* with the Eureqa model only marginally above it. Furthermore, we can see that the SVM model which performed approximately equivalent to random lies almost directly along the *line of no discrimination*, reinforcing the previous observations we made about its performance. The widening gap in the performance and robustness of each model becomes even more evident in the ROC plot for the final 10% holdout experiment in Figure 21, in this visualization we see performance of the XGBoost model increase even further, whereas the remaining models including decision trees perform only marginally better than the *line of no discrimination*. The strong

performance of the XGBoost model in both of the ROC plots is a testament to its robustness as a predictive model in addition to it's overall strong performance.

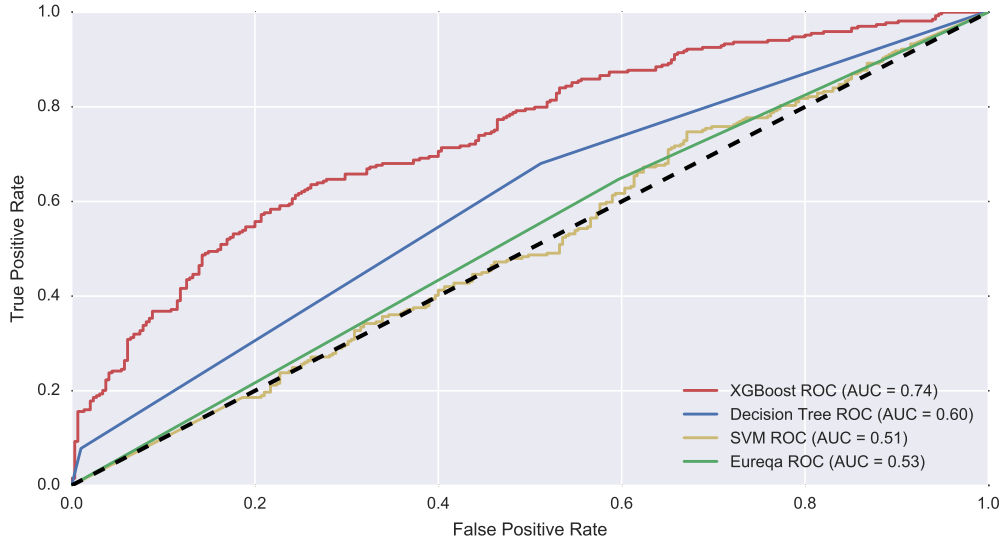


Figure 20: 50% Holdout Model Evaluation ROC Results

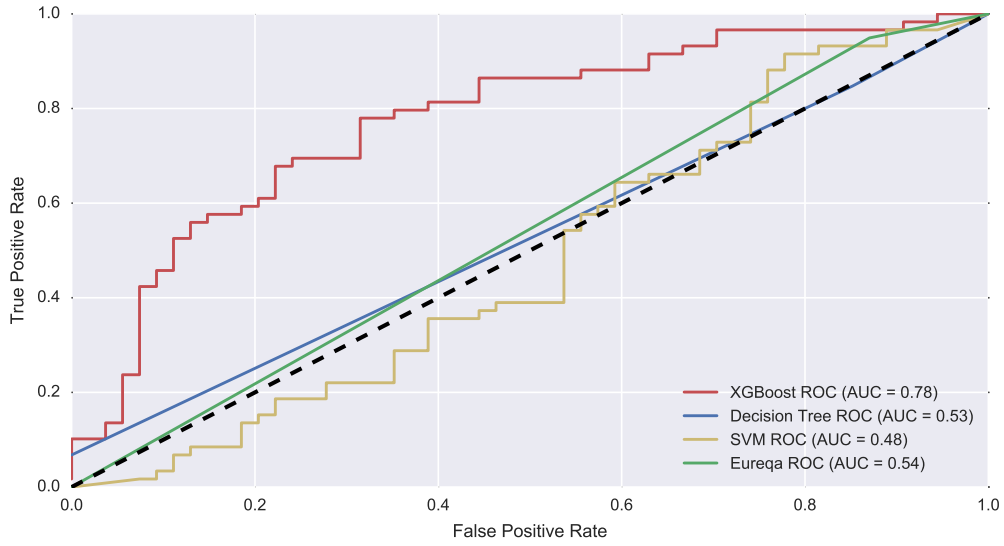


Figure 21: 10% Holdout Model Evaluation ROC Results

We made our final model selection for the experiments in Chapter 4 based on the performance results of each model across the 50%, 30%, and 10% holdout experiments. In each of the experiments the XGBoost model consistently performed much better than all other models, and demonstrated a robustness against over-fitting when the amount of training data was inceased. Over-fitting was a challenge that plagued all of the other models, and even after using Grid Search to

find the optimal parameters we still could not completely avoid it. The detailed experimental results for all of the holdout experiments conducted can be found in Section A.5, in each of the experiments XGBoost performed consistently much better than all of the other models. The exceptional performance of XGBoost as a predictive model is consistent with our initial model evaluation in Section 3.3.2.4; where the ensemble technique used by the model provides a much better predictive outcome than a single model alone can provide.

Chapter IV

EXPERIMENTS AND RESULTS ANALYSIS

4

EXPERIMENTS AND RESULTS ANALYSIS

For the experiments we evaluated the robustness of the predictive model using a combination of holdout experiments, which were used to select the final model in Section 3.3.3, as well as streching and sliding window experiments, which test the daily predictive accuracy of the model as training data is continuously added. Similarly to the model evaulation criteria used to select the final model, the outcome of each experiment is evaluated using *precision*, *recall*, *f1-score*, and *ROC*. Each of these evaluation criterion provide a reliable measurement of the predictive performance of the model as outlined in Section 2.1.2 and are frequently used to corroborate the performance of binary classification models as discussed in the prior research Section 2.1.1.

4.1 EXPERIMENTAL SETUP

The experimental setup was designed to assess the robustness of the predictive model using a combination of holdout experiments, which test against overfitting, as well as strectching and sliding window experiments which evaluate the capabilities of the model when given data daily after each prediction.

The holdout experiments are designed to assess the precitive capabilities of the model with an increasing amount of training data and a decreasing amount of available data for testing. Holdout experiments are frequently used to evaluate the robustness of a model as demonstrated in Section 2.1.2. In each holdout experiment a percentage of the data is used for training the model, while the remaining data is completely withheld referred to as the *holdout*, having no influence on the model. The model is then evaluated against the testing data that was withheld, in each of the experiments an increasing amount of the available data

is used for training, while the remaining data is withheld as the holdout for testing the model. In each scenario the training data is provided as a single whole segment to train the model, with the prediction made on all of the remaining holdout data as one evaluation, this is different from the stretching and sliding window experiments where the model predicts the test data one day at a time. The time periods for the data used in training and testing for each experiment are given in Table 27.

While the holdout experiments evaluate the robustness of the model to overfitting, the stretching and sliding window experiments assess the predictive capabilites of the model in a realistic test environment where new information becomes available daily. With the stretching and sliding window experiments the model is provided new information with each additional day, the current available data is then used to make a prediction for the outcome of the following day. These experimental setups are closely modelled after the concept of *backtesting*, which is frequently used evaluate Bitcoin market predictive models as discussed in Section 2.2.3, and test against historical data with each new amount of additional information. In both experiments a minimum training period of six months is used, following the six month period the price direction of each following day is predicted based on the data added to the training set daily. The training and testing time periods for the stretching and sliding window experiments are given in Table 28, for the sliding window experiments only the previous six months of training data is used, whereas all of the available data is continually added for the stretching window experiments.

Experiment	**Training Period**	**Test Period**
50% Holdout	2013-01-01 – 2014-07-17	2014-07-18 – 2016-02-01
40% Holdout	2013-01-01 – 2014-11-07	2014-11-08 – 2016-02-01
30% Holdout	2013-01-01 – 2015-02-27	2015-02-28 – 2016-02-01
20% Holdout	2013-01-01 – 2015-06-20	2015-06-21 – 2016-02-01
10% Holdout	2013-01-01 – 2015-10-11	2015-10-12 – 2016-02-01

Table 27: Holdout Experimental Setup

Parameter	Stretching Window	Sliding Window
Training Period	2013-01-01 – 2013-06-29	2013-01-01 – 2013-06-29
Test Period	2013-06-30 – 2016-02-01	2013-06-30 – 2016-02-01
Training Data	Continually Added	Previous 6 Months

Table 28: Stretching and Sliding Window Experimental Setup

4.1.1 *Monte Carlo Model*

In each experiment the performance of the predictive model is evaluated in comparison to the outcome of a Monte Carlo method random choice (`UP/DOWN`) in predicting the future price direction. As clearly demonstrated in "Do Not Trust All Simulation Studies Of Telecommunication Networks," by Pawlikowski [89] the selection of a good Pseudo-Random Number Generator (PRNG) is critical to the accurate and reproducable results of any experiment that requires randomness.

For each experiment the PRNG used is *Mersenne Twister*, which is frequently recommended for experiments requiring randomness, and provides uniformly distributed pseudo-random numbers [89–91]. The implementation used was that provided by the `NumPy` and `SciPy` open-source scientific libraries [92, 93]. The PRNG was seeded before each trial using the randomness provided by the Linux kernel through `/dev/urandom`, as it provides adequate entropy based on system events [94]. Lastly, each experiment is conducted with 30 trials using the predicted outcome provided by the Monte Carlo model, the results from each trial are then combined to provide an averaged result for each experiment.

4.1.2 *Experiment Evaluation*

We evaluate the outcome of each experiment using a combination of four metrics, each of which has a benefit in assessing the performance and robustness of a binary predictive model. For each experiment the model is provided the training data and then a prediction of the future price direction (`UP/DOWN`) is made on the

test data which is completely withheld (holdout) from the model during training. The predictive performance of the model in accurately determining if the price will move `UP` or `DOWN` for each future day relative to the price of the previous day is then compared to the known outcome.

A combination of four metrics are used to assess both the performance and robustness of the binary prediction: *precision*, *recall*, *f1-score*, and *ROC*. For each of the metrics the result is a value between $[0, 1]$, a value of 1 represents a perfect result and a value of 0 represents a completely incorrect result. The detailed explanation of each metric in addition to the application for evaluating binary predictive models is described in Section 2.1.2, Table 29 provides a brief explanation of the purpose of each metric used to assess the outcome of the experiments.

Metric	**Summary**
Precision	The ratio of the true positives to the true and false positives and describes the ability of the classifier to not mis-label a positive sample that is negative.
Recall	The ratio of the true positives to the true positives and false negatives and describes the ability of the classifier to find all positive samples.
F1-Score	The weighted average of both the precision and recall and accounts for the contribution of both precision and recall.
ROC	A plot showing the fraction of true and false positives at various thresholds. The area under the ROC curve is calculated providing a single value summary of the performance.

Table 29: Summary of Metrics Used for Experiments

Every experiment is conducted to further support or refute the hypothesis defined in Section 1.3, in each experiment the predictive model is tested against the outcome of the Monte Carlo model using the test data. The performance of each is compared to that of the known actual outcome of the test data using the four metrics. Lastly, the performance of each model is compared for the outcome of all experiments to test the hypothesis and draw conclusions from the results.

4.2 MODEL OPTIMIZATION

The final model parameters for each experiment are configured using a two step hyper-parameter optimization process based on exhaustive Grid Search with varying K-folds for cross-validation of the parameters dependent on the amount of training data available. Grid Search was chosen for the hyper-parameter optimization of the final model in comparison to other methods as it provides an exhaustive search of all parameter values with cross-validation. Furthermore, it is also the standard hyper-parameter optimization method provided by the `scikit-learn` [86] software library and Application Programming Interface (API) used to create the model.

The first step of the hyper-parameter optimization determines the set of parameters for the model using exhaustive Grid Search with cross-validation. After the parameters for the model are chosen the second step determines the optimal number of estimators for the XGBoost model based on the default cross-validation provided by the XGBoost library. The number of estimators has a significant impact on the outcome of the model [23] and as such is optimized each time any of the parameters of the model are modified. Following the estimator optimization, the model is then verified using cross-validation provided by Grid Search, the two step process is repeated for every set of parameter combinations exhaustively tested by Grid Search.

The complete list of parameters optimized for the final model used in each experiment are shown in Table 30. In comparison to the initial model evaluation and selection in Section 3.3.2 where appropriate defaults were used for the majority of parameters a much more detailed list of parameters were chosen for the hyper-parameter optimization of the final model. The list of parameters and possible values for each to evaluate were chosen based on recommendations provided by the XGBoost authors and independent researchers [23, 26, 95], in order to maximize the performance of the final model.

The two step hyper-parameter optimization for the final model used in each experiment resulted in identical parameter values except for the `learning_rate`

Parameter	Significance	Values
`learning_rate`	The learning rate for each boosting round	$[0.01, 0.1, 0.2 \dots 1]$
`max_depth`	Maximum tree depth for base learners	$[2, 4, 6, 8, 12]$
`min_child_weight`	Minimum sum of weight in a child, reduces over-fitting	$[0.25, 0.5, 0.8, 1]$
`gamma`	Minimum loss reduction required to make a split	$[0, 0.1, 0.2, 0.3]$
`subsample`	Fraction of observations randomly sampled for each tree	$[0.25, 0.5, 0.8, 1]$
`colsample_bytree`	Fraction of columns randomly sampled for each tree	$[0.25, 0.5, 0.8, 1]$
`reg_alpha`	L1 regularization term on weights	$[0.001, 0.1, 1, 100]$
`scale_pos_weight`	Faster convergance when there is class imbalance	$[0.25, 0.5, 0.8, 1]$

Table 30: XGBoost Model Hyper-Parameters

and number of estimators. The stability of the optimal parameters for each experiment demonstrates the robustness of the model without the need for computationally expensive hyper-parameter optimization, in comparison to SVM, which required vastly different parameter values for each experiment as highlighted in Section 3.3.2.

4.3 HOLDOUT EXPERIMENTS

The holdout experiments were designed to test the robustness of the predictive model by ensuring that it does not succumb to over-fitting when provided with an increasing amount of training data. Prior to each experiment the parmaters of the model were optimized using Grid Search to ensure that the best possible parameters for each experiment were chosen, in all cases the initial optimized model required very little tuning to achieve optimal results.

Prior to executing each experiment the model parameters were exhaustively optimized, in all cases the only parameters that varied from those shown in Table 30 were the learning rate (η) and the optimal number of estimators. Given that

the amount of training data increased with each holdout experiment the number of K-folds used for the cross-validation were also increased. The optimal learning rate and number of estimators for the model used in each experiment are shown in Table 31, furthermore the number of K-folds for cross-validation for each holdout experiment are also provided.

Experiment	Learning Rate (η)	# Estimators	K-folds
50% Holdout	0.3	6	5
40% Holdout	0.3	7	7
30% Holdout	0.25	17	7
20% Holdout	0.2	13	7
10% Holdout	0.1	12	9

Table 31: Holdout Experiment Parameters

In each of the holdout experiments the model is compared to the performance of the Monte Carlo model, which is based on randomness for the binary predictions. As outlined in Section 4.1.1, the outcome of 30 trials are used as an average for the performance of the Monte Carlo model, whereas with the holdout experiments only one trial is conducted as the resultant outcome of the model is deterministic in each experiment.

For brevity, the outcome of the two experiments in which the Monte Carlo model performed best are provided in Table 32 and Table 33, rather than all holdout experiments. The complete results for each of the holdout experiments are provided in Section A.6. In all of the holdout experiments conducted the XGBoost model received a score for each metric between 0.65–0.69, with the exception of the final 10% holdout experiment, where it received a score for each metric between 0.81–0.84. By contrast, the Monte Carlo model received a consistent score below 0.50, ranging from 0.41–0.48 for all metrics in each holdout experiment conducted, performing significantly worse. An exact 50/50 coin toss model would receive a score of 0.50 out of a range of $[0, 1]$, for each of the metrics, however the Monte Carlo model, even in the best performing experiment, only received a score of 0.48. The reason for it performing less than 0.50 is due to the unbal-

anced support for each actual predictive outcome, in addition to the statistical variances of the uniform distribution.

Model	Prediction	Precision	Recall	F1-Score	AUC	Support
Monte Carlo	0	0.51	0.57	0.53		295
	1	0.45	0.39	0.42		269
	Avg / Total	**0.48**	**0.48**	**0.48**	**0.48**	**564**
XGBoost	0	0.72	0.66	0.69		295
	1	0.66	0.72	0.69		269
	Avg / Total	**0.69**	**0.69**	**0.69**	**0.73**	**564**

Table 32: 50% Holdout Experiment Results

Model	Prediction	Precision	Recall	F1-Score	AUC	Support
Monte Carlo	0	0.45	0.54	0.49		54
	1	0.48	0.39	0.43		59
	Avg / Total	**0.46**	**0.46**	**0.46**	**0.45**	**113**
XGBoost	0	0.79	0.83	0.81		54
	1	0.84	0.80	0.82		59
	Avg / Total	**0.82**	**0.81**	**0.81**	**0.84**	**113**

Table 33: 10% Holdout Experiment Results

The holdout experiments are critical to ensuring that the model is robust and does not suffer from over-fitting as an increasing about of training data is made available as time progresses. The holdout experiments provide a method to evaluate the robustness in addition to the predictive capabilities of the mode as an increasing about of training data becomes available and is then used to predict a decreasing amount of test data (holdout). In each of the holdout experiments the strong performance of the XGBoost model compared to the Monte Carlo model demonstrates its capabilities over prediction using pure randomness. For all holdout experiments the XGBoost model achieved a minimum of 0.20 higher score for all metrics than the Monte Carlo model, with a maximum performance of 0.35 higher score in the case of the 10% holdout experiment.

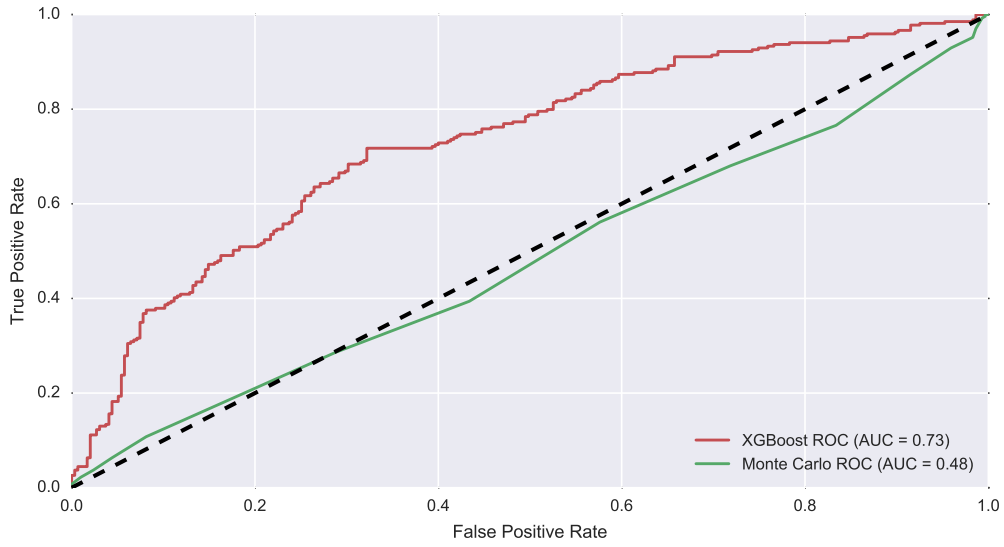


Figure 22: 50% Holdout Experiment ROC Results

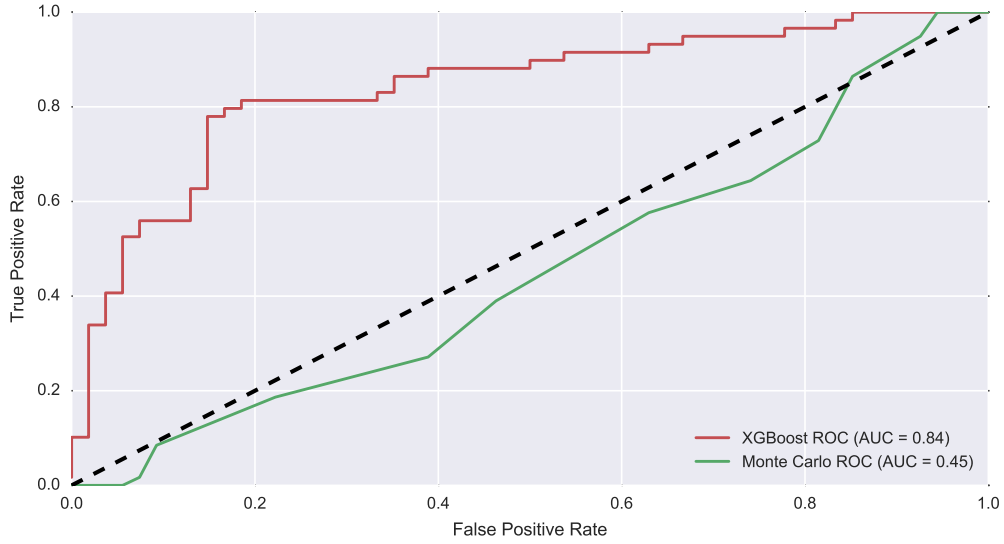


Figure 23: 10% Holdout Experiment ROC Results

For the AUC metric all of the holdout experiments conducted were also plotted to display the ROC and associated AUC. For brevity the associated Figure 22 and Figure 23 for the 50% and 10% holdout experiments described previously are provided, the remaineder can be found in the appendix Section A.6. Again in each of the experiments the AUC was consistently a minumum of 0.20 higher than that of the Monte Carlo model and a maximum of 0.39 higher in the case of the 10% holdout experiment. The ROC plot and associated AUC are described in detail in Section 2.1.2, and are metrics frequently used to evaluate the perfor-

mance of binary predictive models. The benefit that these metrics provide is an intuitive visualization of the fraction of true and false positives at various thresholds.

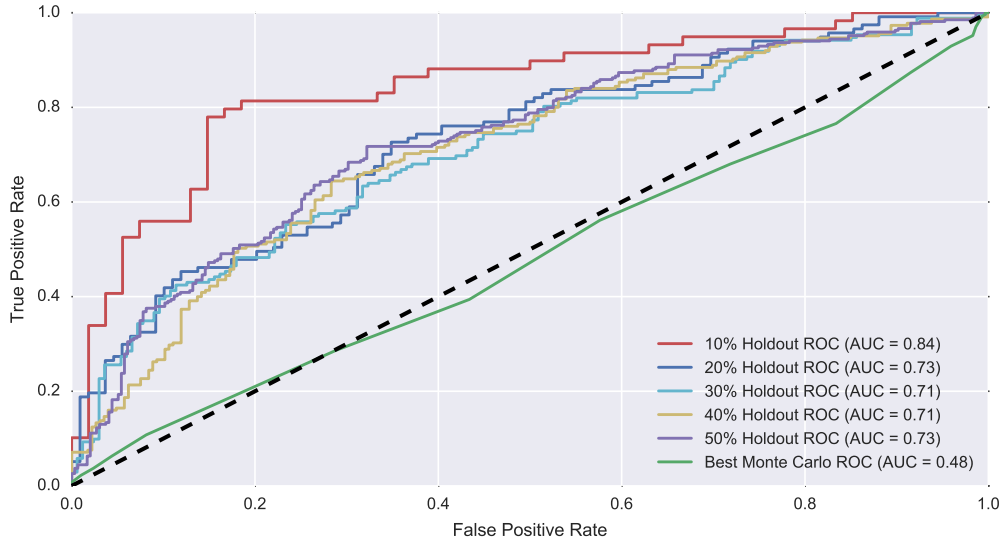


Figure 24: All XGBoost Holdout Experiments ROC Results

We can see in Figure 24 the ROC performance of the XGBoost model in comparison to that of the best performing Monte Carlo model (from the 50% holdout experiment). The dashed black line dividing the upper-left and lower-right parts of the figure is known as the *line of no discrimination*, and represents the performance of a hypothetical model achieving a perfectly random predictive outcome [35]. It is clearly visible that in all holdout experiments the XGBoost models remain far above the *line of no discrimination*, whereas the Monte Carlo model remains slightly worse than an ideal perfectly random predictive outcome. The deviation of our Monte Carlo model from the ideal random predictive outcome represented by the *line of no discrimination* is due to the variances of the uniform randomness of the model, in addition to the actual predictive outcome not being perfectly balanced.

4.4 DAILY PREDICTION EXPERIMENTS

The daily prediction window experiments were designed to demonstrate the robustness of the model in making daily predictions, while simultaneously using the actual outcome of each additional day to re-train the model for predicting the following day. This is a realistic representation of the performance of the model in a real-world scenario, as the model is continually being re-trained based on the additional data added for each new day.

We created two experimental scenarios to evaluate the daily prediction performance of each model, the first being the application of a continually expanding *stretching window* for training of the model, and the second approach a fixed *sliding window*. For the *stetching window* experiments the results of each additional day are added to the model and used for training, such that all of the data is utilized when training the model. Whereas, for the *sliding window* experiments a fixed window of historical data that can be used for training is used; each additional day is still added to train the model, however at most only the previous 6 months worth of data can be used for training. The intention behind the *sliding window* is to reduce the possible effects of over-fitting by limiting the amount of training data to a maximum of 6 months; this removes possible side-effects of much older historical training data on the predictive model.

The experimental parameters for the daily prediction experiments are given in Table 28, in both cases the stretching and sliding window models used the same learning rate $\eta = 0.01$. The learning rate used for the daily prediction experiments is significantly smaller than the minimum value 0.10 used in the holdout experiments, likely due to the large amount of training data being used relative to only predicting the following day. For each model the stabilized parameters used for the holdout experiments in Section 4.3 were used, only the learning rate and number of estimators were optimized using the two step hyper-parameter Grid Search optimization. Due to the dynamically changing set of available training data, both models were re-optimized using Grid Search at the start of each month to ensure that the learning rate and number of estimators were adjusted

to reflect the new training data available. Over the course of the entire test period the optimized learning rate for both models was always $\eta = 0.01$, with an average number of estimators as 176 for the stretching window model and 17 for the sliding window model.

Parameter	**Stretching Window**	**Sliding Window**
Training Data	Continually Added	Previous 6 months
Model Re-Calibration	Monthly	Monthly
Learning Rate (η)	0.01	0.01
Average # Estimators	176	17
K-folds	5	3

Table 34: Daily Prediction Experiment Parameters

The realistic experimental setup for daily prediction demonstrated exceptional results, the following Table 35 lists the results for the daily prediction *stretching window* experiments. In each case the model was used to predict the price direction (`UP/DOWN`) for the following day, entire test period spanned from 2013-06-30 – 2016-02-01, with the initial first six months used as training data from 2013-01-01 – 2013-06-29. The actual results of each day were then added as additional training data, with the model re-trained before predicting the following day. Across all of the metrics used to evaluate the performance of the predictive model for the *stretching window* experiments we see that the XGBoost model receives a score between 0.69–0.73, nearly 0.20 higher than that of the Monte Carlo model, which recevied a score of 0.50 for all of the metrics. We see that the Monte Carlo model is approximates a perfect random prediction much better in comparison to the previous holdout experiments due to the increased amount of test data (947 samples), allowing the random variances to become more perfectly uniform random.

The experimental results for the *sliding window* model are given in Table 36, we see that in the case of the sliding six month window that the performance of the XGBoost model is nearly identical to that of the stretching window experiments. For each of the metrics the XGBoost model receives a score between 0.69–0.72, nearly identical to the score received for the sliding window model. Again for each metric the XGBoost model receives a score approximately 0.20 higher than

that of the Monte Carlo model, demonstrating that the model still performs well even with a smaller window of available historical training data.

Model	Prediction	Precision	Recall	F1-Score	AUC	Support
Monte Carlo	0	0.49	0.56	0.52		464
	1	0.51	0.44	0.47		483
	Avg / Total	**0.50**	**0.50**	**0.50**	**0.49**	**947**
XGBoost	0	0.65	0.77	0.70		464
	1	0.73	0.60	0.66		483
	Avg / Total	**0.69**	**0.68**	**0.68**	**0.73**	**947**

Table 35: Stretching Window Experimental Results

Model	Prediction	Precision	Recall	F1-Score	AUC	Support
Monte Carlo	0	0.51	0.58	0.54		464
	1	0.53	0.45	0.49		483
	Avg / Total	**0.52**	**0.52**	**0.51**	**0.52**	**947**
XGBoost	0	0.65	0.80	0.72		464
	1	0.75	0.58	0.66		483
	Avg / Total	**0.70**	**0.69**	**0.69**	**0.72**	**947**

Table 36: Sliding Window Experimental Results

Similarly to the holdout experiments, we conducted a final comparison of the performance of the stretching and sliding window models to the Monte Carlo model using an ROC plot and the associated AUC. The ROC plot and associated AUC are described in detail in Section 2.1.2, and are metrics frequently used to evaluate the performance of binary predictive models. The benefit that these metrics provide is an intuitive visualization of the fraction of true and false positives at various thresholds, as well they provide a simple way to visually compare the performance of both the stretching and sliding windows to the Monte Carlo model in one figure. The results of the stretching window experiments are shown in Figure 25, where we see that the Monte Carlo model lies approximately along the *line of no discrimination*, close to the performance of a hypothetical perfect random prediction. Furthermore, we see that the XGBoost model lies well above the line, with an AUC of 0.73, that is 0.24 greater than that of the Monte Carlo Model.

The results of the sliding window experiments are nearly identical to that of the stretching window, in Figure 26 we see that again the XGBoost model lies well above the *line of no discrimination* with the Monte Carlo model just marginally above it due to the random variances of the uniform random distribution. Again we see that the XGBoost model has an AUC of 0.72, nearly the same as that of the stretching experiments, with a result that is 0.20 greater than that of the Monte Carlo model. Lastly, we compare the performance of both the stretching and sliding window models to the best Monte Carlo model, we see in Figure 27 that the performance of both XGBoost models is nearly identical. In each model the ROC curve is either slightly higher or lower at certain points, demonstrating the visible trade-offs in each model. However the combined area under each curve is nearly identical with the stretching window having an area of 0.73 and the sliding 6 month window having an area of 0.72.

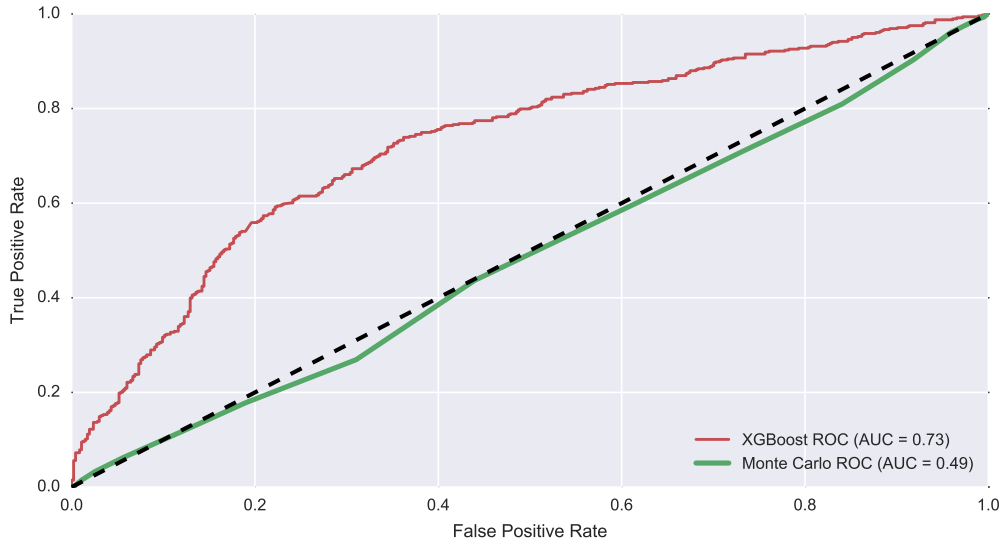


Figure 25: Stretching Window Experiment ROC Results

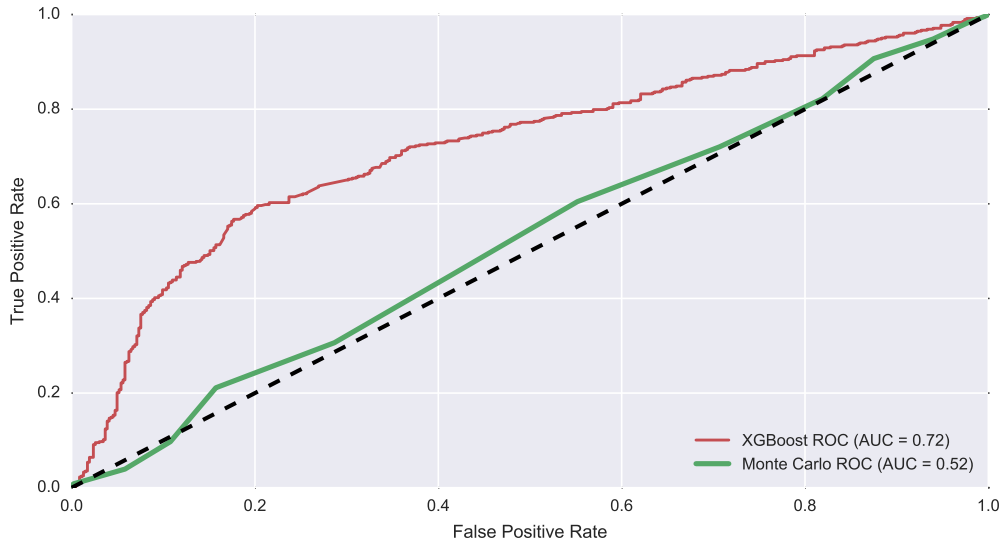


Figure 26: Sliding Window Experiment ROC Results

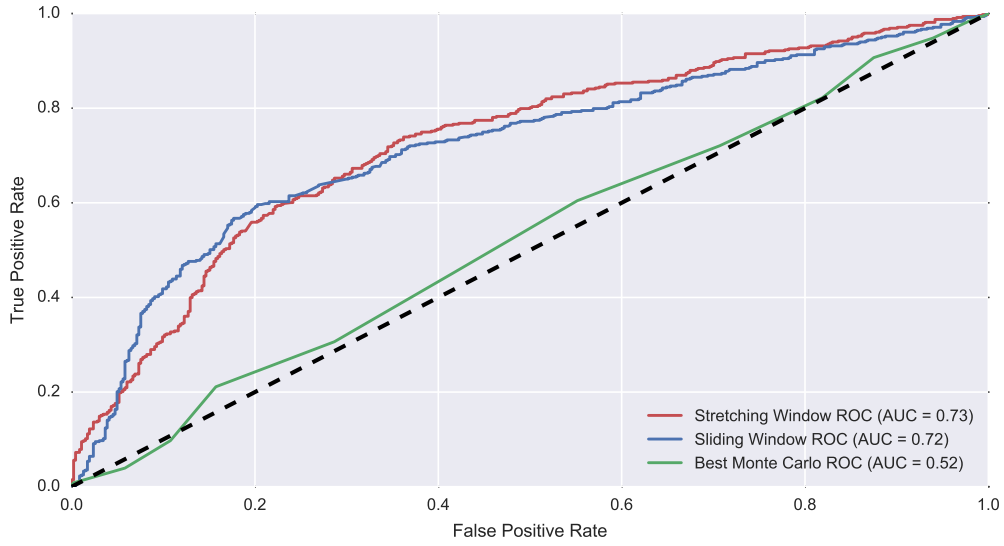


Figure 27: Sliding and Stretching Window Experiments ROC Results

Given the performance of the XGBoost model in the stretching and sliding window experiments, we then demonstrated the actual price direction (`UP/DOWN`) prediction accuracy of the model by plotting the prediction results overlaid on the historical price of Bitcoin in USD. In Figure 28 the historical price of Bitcoin is plotted as a faded dashed line for the final three months of the daily prediction experiments from 2013-06-30 - 2016-02-01. All of the daily price direction predictions where the model was correct are plotted as a blue point, the false predictions are plotted as a red point. The results are self-evident, over the course of the

final three months of the test period the XGBoost model correctly predicted the future price direction for 73 out of a total of 93 days, only making a false prediction for 20 out of the total of 93 days. The predictive performance of the model is demonstratably much better than that of pure randomness, as the model makes a correct prediction for the price direction 73 out of the total of 93 days, an accuracy of 79%.

The same visualization was repeated for the sliding window model, in Figure 29 we again see the historical price of Bitcoin in USD represented by a dashed line and the false and correct predictions overlaid on top of the price. For the sliding window model there were a total of 65 days where the price direction was correctly predicted out of a total of 93 days, compared to 28 false predictions. While the accuracy for the final three months is slightly lower (70%) than that of the stretching window, the results are still much higher than that of pure randomness. Furthermore as the time period is increased from three months to six months the accuracy of the two models converge and are approximately the same, the results can be found in Section A.7 for the final three and six months of the experiments.

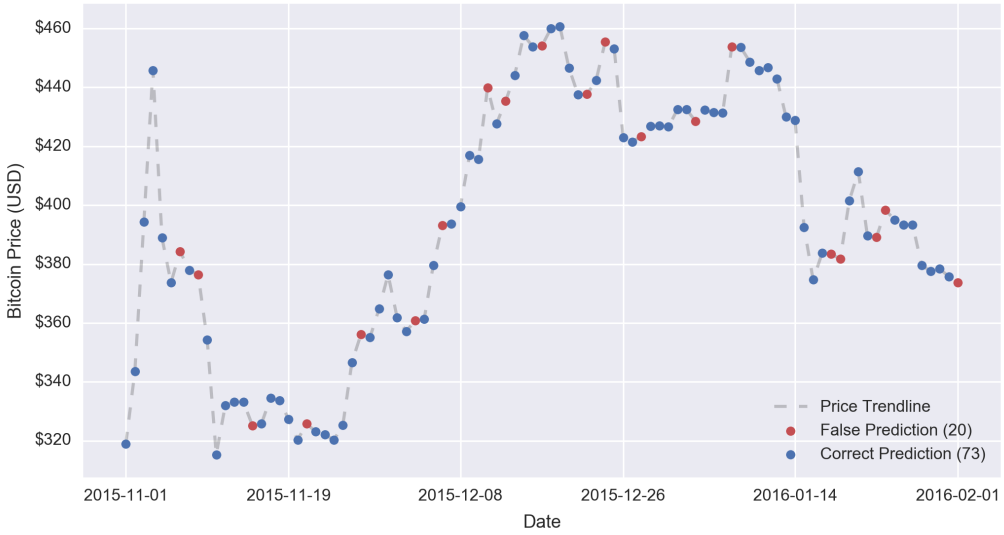


Figure 28: Stretching Window Prediction, Final 3 Months

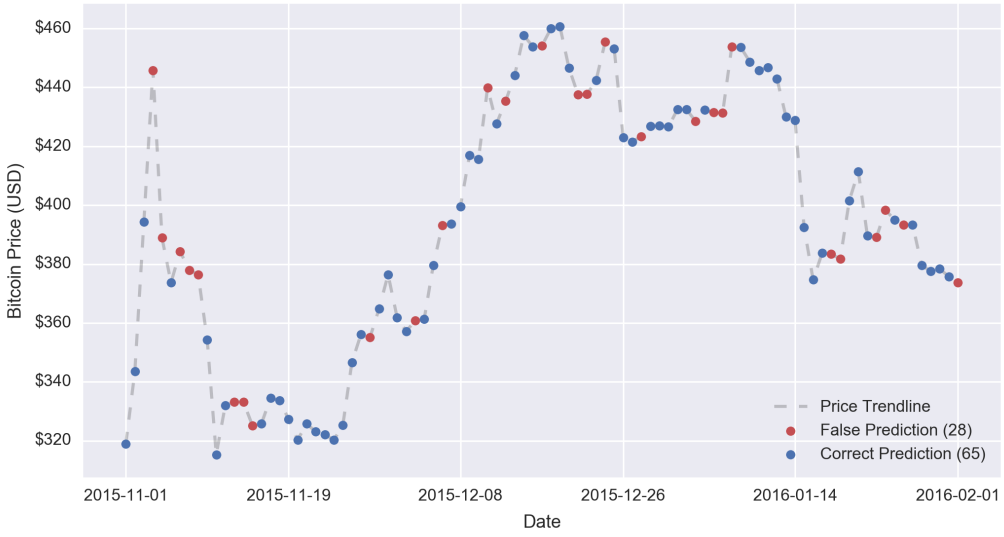


Figure 29: Sliding Window Prediction, Final 3 Months

The XGBoost model consistently performed strongly for both the stretching and sliding window experiments, demonstrating a resilience to over-fitting even when the predictive model was continually being provided additional training data after each additional day. The strong performance of the XGBoost model is a testament to it's consistent strong performance demonstrated in other research applications [23–25]. When applied to a realistic scenario performing daily predictions the model demonstrates similar performance to our holdout experiments. Lastly, the visualizations shown in Figure 28 and Figure 29 give an intuitive assessment of the actual predictive accuracy of the model relative to the high volatility of the Bitcoin market during the final three months of the experiments.

4.5 CONCLUSIONS

Through a multitude of experiments we tested our original hypothesis extensively by comparing the predictive performance of a uniform random Monte Carlo model against our own predictive XGBoost model based on the features of network influencers. We evaluated each of the models using a combination of multiple evaluation metrics: *precision*, *recall*, *F1-score*, and *ROC*, each designed for evaluating the performance of binary classifiers. We established our selection of

evaluation metrics based on a wide body of literature review and the evaluation metrics used by other researchers to compare predictive models [7, 28, 30, 35]. For each of these metrics the range of possible values is between $[0, 1]$, with a value of 0.50 representing a perfectly random prediction, 0 a complete incorrect prediction, and 1 a perfect prediction.

We evaluated each of the predictive models using a number of experiments designed to compare both the performance and robustness of each model. We designed a set of five holdout experiments (50%, 40%, 30%, 20%, 10%), each with a decreasing amount of the data set aside to test the model, the remaining used for training. The holdout experiments were created to evaluate not only the predictive performance of each model, but the robustness of our XGBoost model to over-fitting, when the amount of training data is increased relative to a decreasing amount of test data.

In each of the holdout experiments the XGBoost model exceeded the results of the Monte Carlo model, achieving consistently a minumum of 0.20 higher than that of the Monte Carlo model for all experiments and all metrics used. In the case of the 10% holdout experiments the XGBoost model had a score 0.39 higher than that of the Monte Carlo model, with a score of 0.84. The Monte Carlo model received a score of approximately 0.50 over all the experiments and metrics, close to that of a perfect random prediction. The Monte Carlo model in some cases received a score slightly below or above 0.50 due to the imbalance in the number of preditive outcomes (`UP/DOWN`) for the test data. Compared to the Monte Carlo model, the XGBoost model received a score for each metric between 0.65–0.69, with the exception of the final 10% holdout experiment, where it received a score for each metric between 0.81–0.84. The XGBoost model demonstrated a consistent high level of accuracy and robustness in all cases, achieving a score significantly higher than that of the purely random Monte Carlo model.

Following the results of the holdout experiments, we evaluated a more realistic application of each model, where the model was used to predict the price direction for each following day based on the previous historical training data available. After each prediction, the accuracy of the model was recorded, and

the outcome of the following day was then used to train the model to predict another future outcome. Consistent with our holdout experiments, the XGBoost model achieved a minimum score of approximately 0.20 higher than the Monte Carlo model, with the values for each metric ranging from 0.68–0.73. Lastly we demonstrated what the actual outcome of the predictive model would look like when plotted against the historical price of Bitcoin for the final three months of the test data. Both the stretching and sliding window models achieved an accuracy exceeding 50%, with the stretching window predicting 73 out of a total of 93 days correctly (79% accuracy), and the sliding window model predicting 65 days correctly (70% accuracy).

The consistent strong performance of our XGBoost model supports our hypothesis of a predictive model that can be used to predict the price direction (`UP/DOWN`) of the Bitcoin market based on the action of network influencers with better accuracy than that of random chance. The results of each experiment were evaluated using the four leading metrics used for evaluting binary classifers based on our literature review. In all of the experiments conducted our XGBoost model achived apprximately 0.20 higher than that of the Monte Carlo model, which achieved a score approximately equivalent to that of random chance. The consistent performance of the XGBoost model in all experiments and the robustness to over-fitting supports our hypothesis that a predictive model for the Bitcoin market can be made based on the features extracted from the actions of major network influencers.

Chapter V

CONCLUDING REMARKS, CONTRIBUTIONS, AND FUTURE WORK

5

CONCLUDING REMARKS, CONTRIBUTIONS, AND FUTURE WORK

5.1 CONCLUDING REMARKS

Throughout the entire research process we have closely modelled our methodology and experiments around the basis of our hypothesis and objectives outlined in Section 1.3 and Section 1.4. Where our hypothesis is that a predictive model can be created to predict the daily price direction of the Bitcoin market with better accuracy than pure randomness based on the market data, network features, and activities of users that accumulate a disproporionate amount of wealth using exchanges. Given the hypothesis we defined three main objectives for our research: defining major network influencers clearly as users that accumulate a disproportionate amount of wealth by using exchanges, creating metrics to identify said users, and creating an accurate predictive model based on their actions.

We conducted a thorough background and literature review of the state of the art in machine learning models that were used as a basis for our own predictive model. As well, we also reviewed the approaches by other researchers to have better insight into the Bitcoin network and predict the future outcome of the markets based on analyzing the patterns of transactions on the blockchain and the positive and negative sentiment of Bitcoin on social media. We then reviewed specifically the state of the art in existing predictive models used to predict the future price direction of the Bitcoin market in Section 2.2.3, and used the predictive models created by researchers as a basis for our own models. Furethermore, we also created our own unique model using XGBoost, a recently published machine learning model that has demonstrated exceptionally good results for numerous other researchers in prediction and classification problems [23–25]

For our methodology we set out to meet all of the objectives defined based on our hypothesis, we began first by performing the necessary pre-processing to extract the information contained within the Bitcoin blockchain and store it in a relational database. Following the extraction of the blockchain we then outlined the creation of our own unique parallel address clustering method, based on the *Union-Find* variants created by other researchers [44, 47–49], which allowed us to identify all of the addresses belonging to each user. We then met our objective of identifying the major network influencer by creating and evaluating several metrics we created in Section 3.2. Following the creation of our metrics, we created several predictive models based on the features of the market, network transactions, and actions of major network influencers identified by our metrics. We created our predictive models based on the existing state of the art in addition to creating a completely unique predictive model based on XGBoost. Lastly, we evaluated each model, we found that our models based on the existing state of the art achieved similar performance, only slightly better than randomness with 50%–55% accuracy, but that our new XGBoost model performed much better than all the other models with a score of 0.10–0.20 higher for each metric.

In our final experiments we tested our hypothesis using the XGBoost model from our initial model evaluation, which had exceptional performance compared to all of the other models evalulated. We assessed our XGBoost predictive model compared to that of a purely random Monte Carlo model using four different metrics with a score between $[0, 1]$, each widely used in machine learning applications: *precision*, *recall*, *F1-score*, and *ROC*. With the metrics we then compared the performance of our predictive model to the Monte Carlo model using a number of experiments designed to compare both the performance and robustness of each model. We designed a set of five holdout experiments (50%, 40%, 30%, 20%, 10%), each with a decreasing amount of the data set aside to test the model, the remaining used for training. The holdout experiments evaluate not only the predictive performance but also the robustness of the model to over-fitting as the training data is increased and testing data decreased. In each of the holdout expriments the XGBoost model greatly exceeded the accuracy of the Monte

Carlo model achieving consistently a minumum of 0.20 higher than that of the Monte Carlo model for all holdout experiments and all metrics used. For the last 10% holdout experiment the XGBoost scored significantly higher than that of the Monte Carlo model, which was approximately 0.46, with a score of 0.81–0.84.

Following the holdout experiments we then evaluated each model with a more realistic application. Where the XGBoost model was used to predict the price direction for each following day based on the previous historical training data, with the actual outcome of each day then used to re-train the model for predicting the next following day. Two experiments were created in this manner, the first continually adding the entire history of each day to be used for training the model, the second using only a maximum of the past six months for training. Consistent with our holdout experiments, the XGBoost model achieved a minimum score of approximately 0.20 higher than the Monte Carlo model, with the values for each metric between 0.68–0.73 compared to the Monte Carlo model with approximately 0.49–0.52. For a final evaluation we plotted the outcome of the predictive model for the final three months of market data. In terms of the number of predictions correct versus incorrect, the XGBoost model with the entire history used for training, predicted 73 out of a total of 93 days correctly (79% accuracy), and when using the past six months for training predicted 65 days correctly (70% accuracy).

The consistent strong performance of our predictive model using XGBoost and based on the feature engineering of the market history, blockchain activity, and actions of major network influencers is supportive of our hypothesis. Throughout our research we met all of our objectives and created a predictive model which uses the actions of major network influencers as features, when compared to a purely random Monte Carlo model our model consistently performs much better. We demonstrated in each of our holdout experiments, designed to test the robustness of the model to overfitting, a consistent performance of 0.20 higher than the Montel Carlo model, and in the 10% holdout experiment 0.46 higher with a score of 0.81–0.84 out of a maximum of 1.0 for a perfect score. For our experiments designed to test a more realistic application of our predictive model, we

again see that our model achieves a minimum score of approximately 0.20 higher than the Monte Carlo model with the values for each metric between 0.68–0.73. As well, we also demonstrated for the final three months of training data that our model predicted 73 out of a total of 93 days correctly (79% accuracy), which is approximately 25% higher than the accuracy of all other existing predictive models which achieve an accuracy between 50%–55% [67, 72, 73]. The consistent strong performance of our predictive model, which performed significantly better than the Monte Carlo model in all experiments conducted, strongly supports our hypothesis. Furthermore, our predictive model emphasizes the importance of our contribution as our experiments demonstrated an accuracy 25% higher than that of all other existing predictive models for the Bitcoin market.

5.2 CONTRIBUTIONS

During the course of our research we made numerous contributions to the existing body of available research pertaining to Bitcoin. Much of our own research was based on our extensive literature review in Chapter 2, where we extended much of the prior research as necessary in order to meet our research objectives and to test our hypothesis; we clearly outline each of our contributions as follows.

RELATIONAL BLOCKCHAIN DATABASE: While many other researchers have mentioned vaguely their methodology of extracting the Bitcoin blockchain and storing the information in a relational database [44, 48, 57, 72], with the exception of the simple database schema published by Spagnuolo, Maggi, and Zanero [49] no detailed RDBMS schema has been published for the Bitcoin blockchain. We provide the complete and detailed schema diagram of our RDBMS in Section 3.1.1, this schema is much more detailed than the simplified schemas available and provides not only a relational representation of every possible facet of information stored in the blockchain, but also the consideration of aggregation results. Our schema is a valuable contribution to any future researchers that need to process the Bitcoin blockchain and perform analyses of it, as the blockchain

is not in a format that is easy to analyze, it must be first processed and the information extracted into another medium of storage such as an RDBMS. Future researchers can use our detailed shema as a reference to design their own RDBMS, which allows for the support of SQL to perform OLAP and extract information for analyses easily on the entire blockchain data.

PARALLEL ADDRESS CLUSTERING: The process of clustering all of the addresses owned by a single user is known as *address clustering*. While many researchers apply the same heuristics and clustering technique known as *Union-Find* [44, 47–49], a computationally expensive process, we have proposed a novel version that can be executed in parallel. We were required to create a parallel address clustering algorithm as the size of the Bitcoin blockchain and number of transactions had increased significantly since the application of the algorithms by researchers several years prior. We describe our parallel algorithm in Section 3.1.2, which greatly expedited the process of clustering all of the addresses, and will be even more beneficial to future research work as the blockchain continues to increase in size.

METRICS TO IDENTIFY NETWORK INFLUENCERS: As part of the fulfillment of our objectives and to evaluate our hypothesis we created metrics that can be used to identify the influencers within the Bitcoin network. Based on our hypothesis our primary purpose was to identify users that had accumulated a disproportionate amount of wealth, however each of our metrics in Section 3.2 could be used to identify influencers based on any other criteria. Our metrics are beneficial as they extend widely-used methods of ranking individuals (h-index), as well as optimizing multiple objectives (Pareto optimization), and can be applied by researchers to identify influencers based on any criteria.

PREDICTIVE MODEL FOR THE BITCOIN MARKET: Our most notable contribution, and the purpose of our hypothesis, was the creation of a predictive model for the Bitcoin market. We have demonstrated that our model not only has a high accuracy in comparison to a Monte Carlo model, but also is significantly more

accurate by at least 25% than all other existing models. Furthermore, we selected XGBoost, a very new and highly successful machine learning model, for the creation of our final model which is significanly different than all other previous predictive models for the Bitcoin market. Our application of XGBoost and our unique approach of using the actions of major network influencers as features for our predictive model is a unique contribution not even considered or discussed by any other researchers.

5.3 FUTURE WORK

While our final predictive model demonstrated significantly higher accuracy compared to a Monte Carlo model as well as the results of other researchers, there are still several areas we would like to consider for future research. Based on our own literature review as well as some of the limitations of our own research we have identified the following areas of future research.

ADDING BITCOIN SENTIMENT AS FEATURES: During our literature review we discussed the application of social media sentiment analysis related to Bitcoin by other researchers. While the researchers could not conclusively conclude wether or not the sentiment on social media could predict the direction of the Bitcoin market, it would have also been beneficial to have considered social media sentiment as one of our features. We attempted to gain access to the historical tweets pertaining to Bitcoin provided by a subsidiary of Twitter, GNIP[1], but found the costs of gaining access to the data prohibitively expensive[2].

INCLUDING MORE MARKET FEATURES: While we included the historical market features of market volatility, trading volume, price close, and average price there are other features that we could have considered which may have further improved the accuracy of our model. For future research we would like to consider more market features such as the intraday-spread and intraday-return that

1 GNIP unleash the power of social data. https://www.gnip.com
2 We were invoiced at a cost of $50,000 USD for tweets with `#bitcoin` from 2013-01-01 – 2016-01-01

were used by other researchers [57], as well as making the features specific to the individual USD, CNY, and EUR markets rather than aggregated them as a whole.

DEEP LEARNING: While we had great success using XGBoost for our predictive model we would also like to evaluate and consider the posibility of deep learning models based on the concepts of neural networks. Initially we considered using deep learning after the results of Greaves and Au [73], which had the best predicitve performance out of all other researchers using a simple neural network. However, we were unable to effectively evaluate deep learning due to the sheer magnitude and dimensions of our training data. During our research we had only limited access to GPUs and any attempts to create even simple deep learning model using CPUs resulted in extremely long training and testing times. For future research, with access to powerful GPUs, we would like to re-evalaute the model and compare the performance of XGBoost to that of a deep learning predictive model.

Chapter VI

APPENDIX

APPENDIX

a.1 MARKET OPPORTUNITY

Figure 30: Bitcoin Price in USD Markets

Figure 31: Bitcoin Price in CNY Markets

Figure 32: Bitcoin Price in EUR Markets

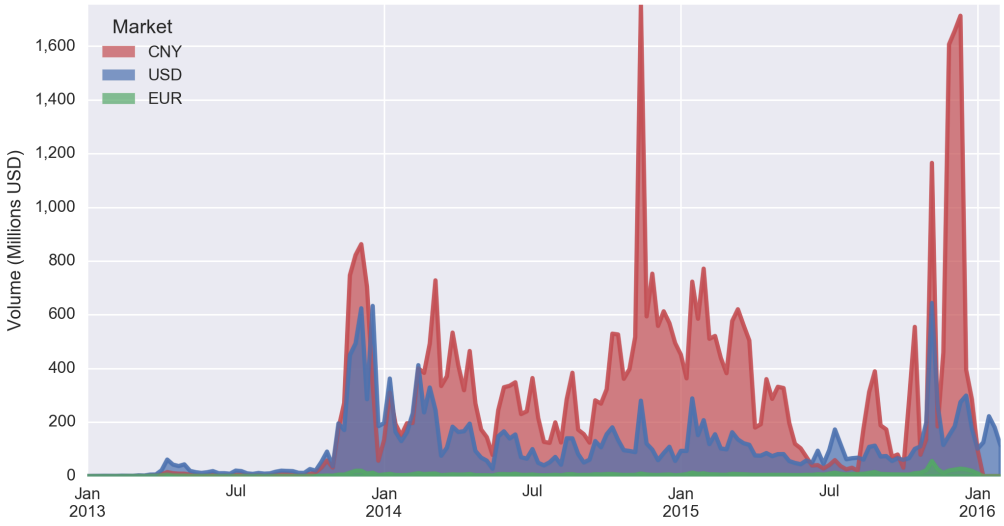


Figure 33: Bitcoin Weekly Volume for USD, CNY, EUR Markets

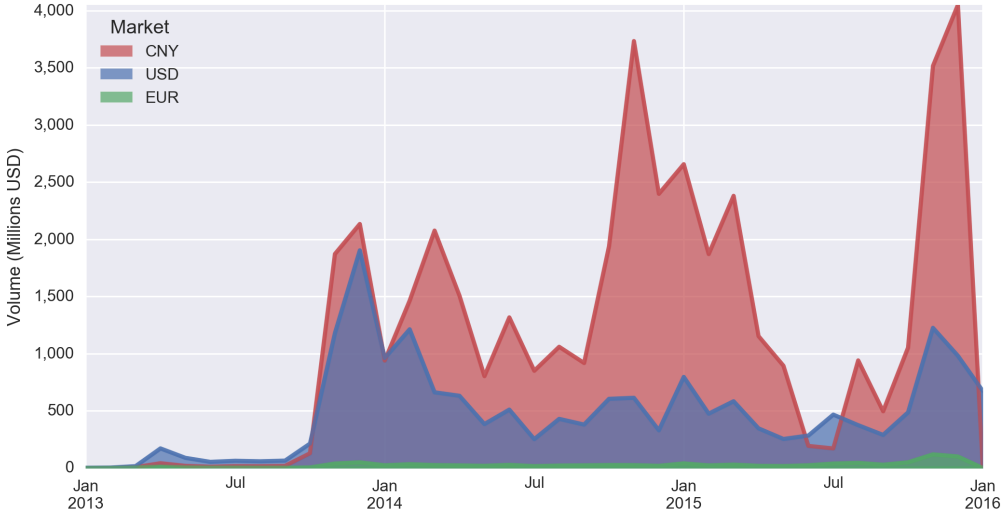


Figure 34: Bitcoin Monthly Volume for USD, CNY, EUR Markets

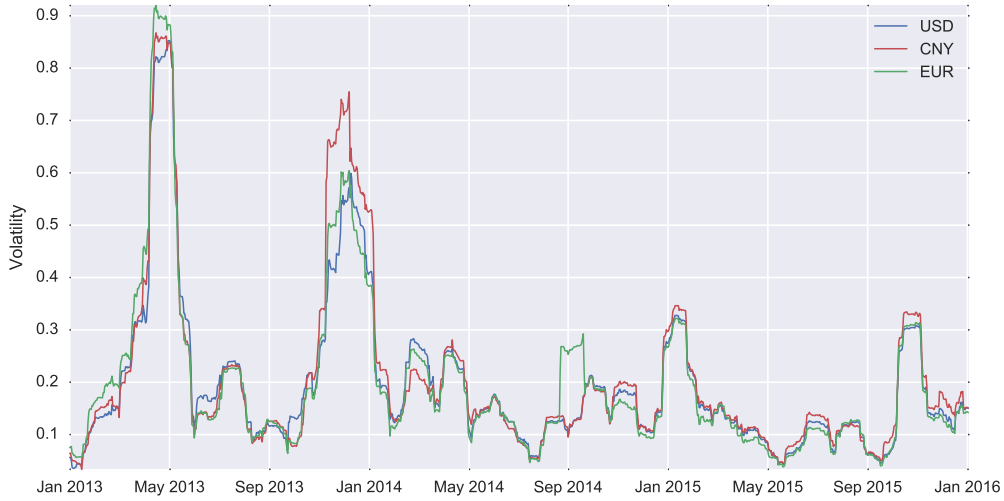


Figure 35: Bitcoin Volatility for USD, CNY, EUR Markets

a.2 BIT-INDEX CALCULATION EXAMPLES

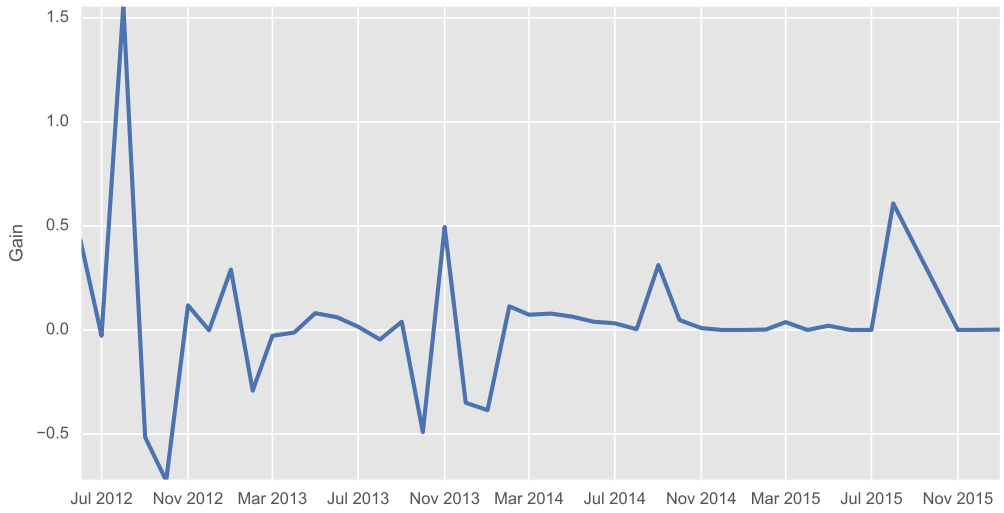


Figure 36: Silk Road Historical Monthly Gain

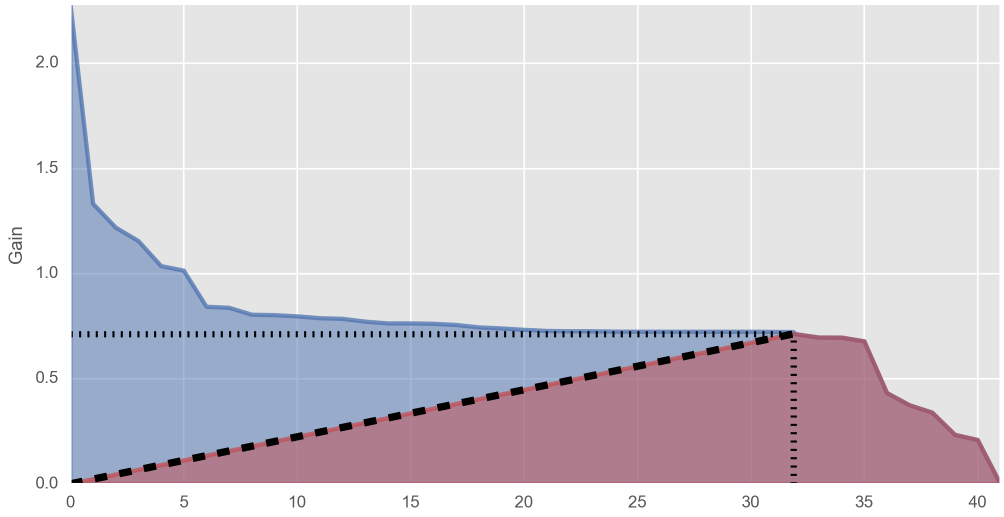


Figure 37: Silk Road Historical Monthly Gain

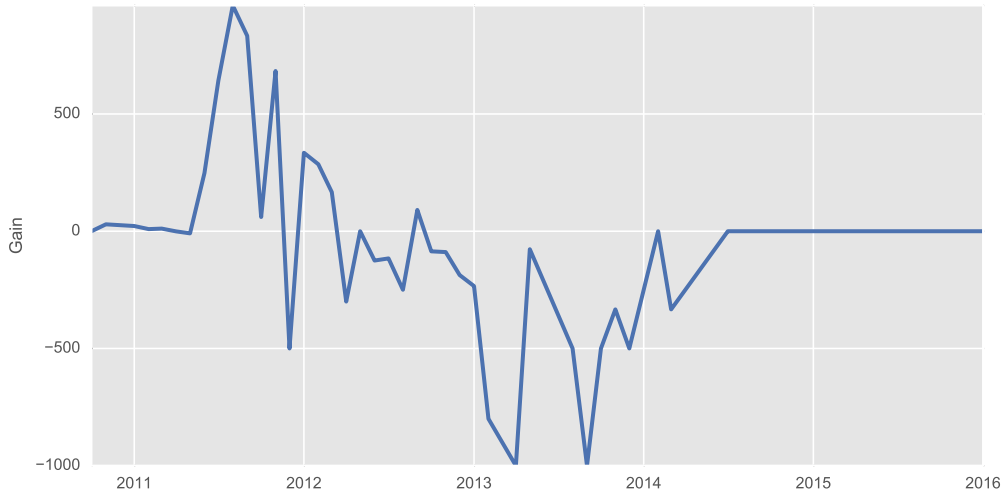


Figure 38: User with 8,700 BTC Historical Monthly Gain

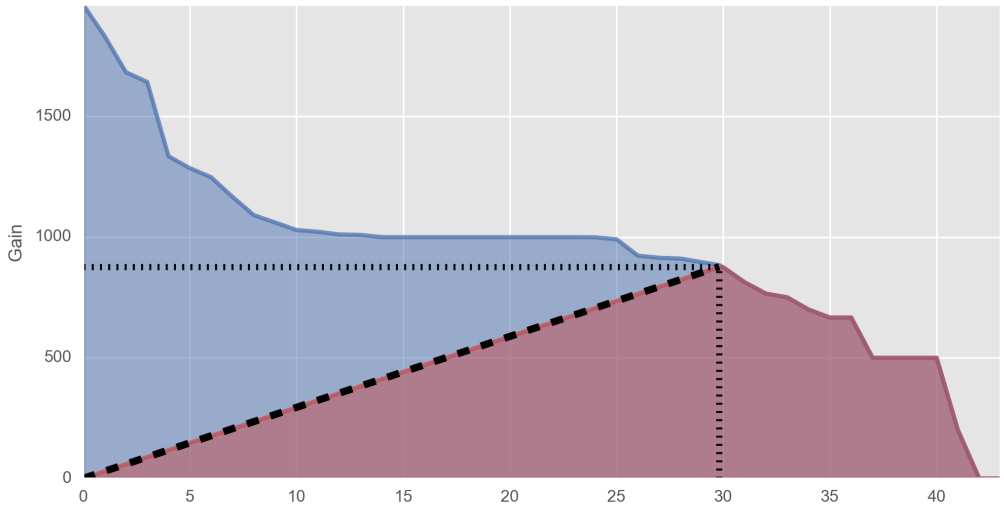


Figure 39: User with 8,700 BTC Historical Monthly Gain

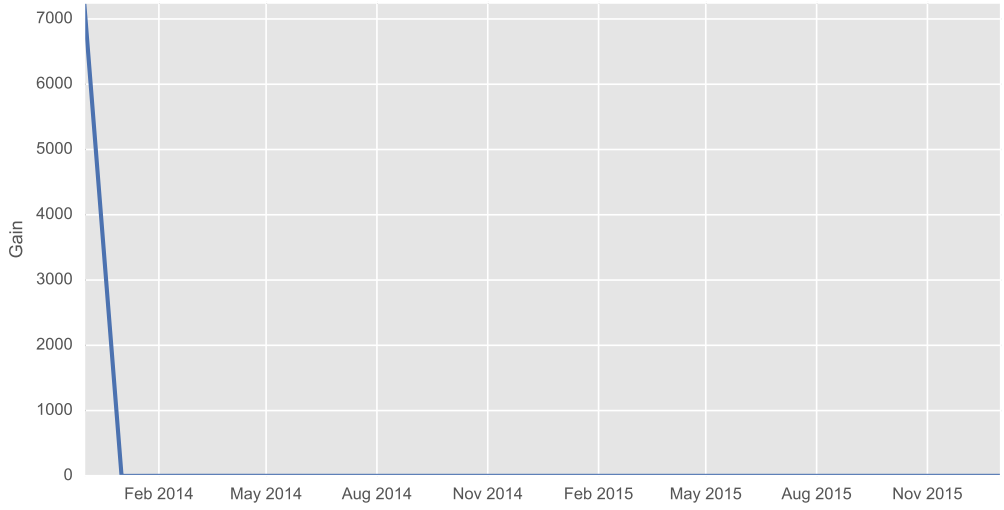


Figure 40: User with 21,744 BTC Historical Monthly Gain

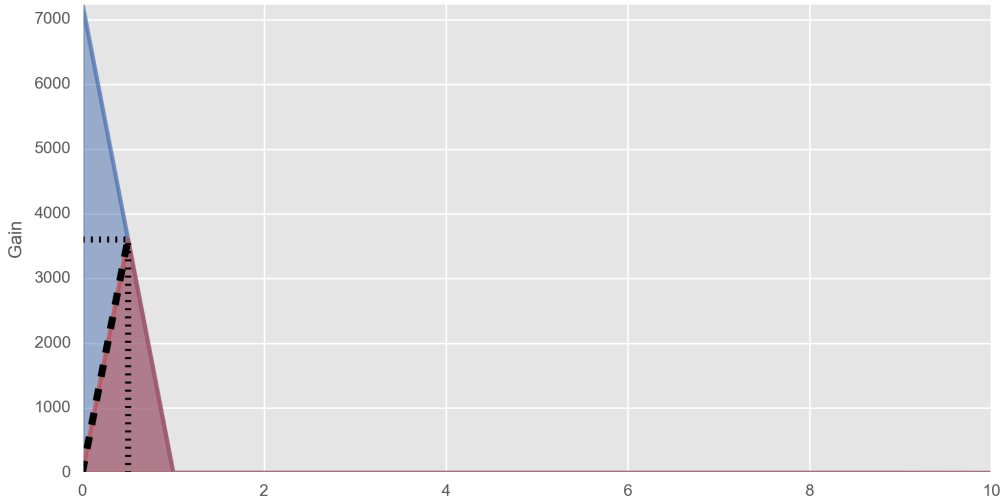


Figure 41: User with 21,744 BTC Historical Monthly Gain

a.3 PARETO FRONT CLUSTERING

Wallet ID	Size	Gain	Balance	Transactions	Volume	Active Period
967094	35	−10,010,510	87,112	272	698,050	2011-06-13 – 2016-01-21
29091650	1	15,954,569	69,370	63	208,312	2013-04-09 – 2016-01-21
36816796	98,614	16,731,573	39,606	501,423	3,503,300	2013-09-09 – 2016-02-03
64546	7	−42	31,000	58	75,000	2010-05-04 – 2016-01-21

continued ...

continued ...

Wallet ID	Size	Gain	Balance	Transactions	Volume	Active Period
26418861	8	9,785,598	28,050	456	73,252	2015-02-16 – 2016-02-03
12910452	1	15,186,848	21,744	17	21,746	2013-12-19 – 2016-01-21
35715895	1	3,314,492	12,277	170	21,102	2015-08-06 – 2016-01-27
28160350	1	2,298,647	9,112	27	9,114	2015-04-02 – 2016-01-21
37610306	1	1,833,219	5,421	15	5,433	2015-08-12 – 2016-01-30
35592877	109	1,077,230	5,361	12,161	18,204	2015-08-10 – 2015-12-25
10479526	1	2,344,015	5,297	4	30,063	2013-10-24 – 2016-01-21
38299190	5	1,604,640	4,352	22	5,288	2015-10-04 – 2016-02-03
16475247	1	1,612,456	4,343	3	19,975	2014-05-07 – 2016-02-01
18936475	1	1,595,988	2,720	8	2,728	2014-08-04 – 2015-04-02
36268175	1	507,228	2,282	16	20,717	2015-08-26 – 2016-01-14
34249409	1	387,395	2,001	6	4,061	2015-06-04 – 2015-11-18
29369438	1	453,508	1,740	6	2,122	2015-03-08 – 2015-04-29
35283989	1	205,020	1,650	8	1,751	2013-04-13 – 2015-08-11
38549999	1	327,146	1,410	46	2,547	2015-09-02 – 2015-10-14
36682013	1	809,448	1,173	7	2,353	2014-08-11 – 2015-09-06
27018864	1	302,603	1,112	7	1,154	2015-03-06 – 2015-04-02
33047611	1	242,685	900	11	902	2015-07-07 – 2015-07-08
36331101	1	225,141	800	3	830	2015-08-03 – 2015-08-31
38964574	1	212,947	736	9	936	2014-08-21 – 2015-10-16
39144453	1	111,471	400	8	2,136	2015-03-16 – 2015-12-17
38362929	1	72,911	303	10	403	2015-10-01 – 2015-10-08
36800682	1	71,360	293	3	979	2015-09-08 – 2015-09-08
37322290	1	78,913	285	9	492	2015-06-29 – 2015-09-18

continued ...

continued ...

Wallet ID	Size	Gain	Balance	Transactions	Volume	Active Period
37096007	2	58,377	253	5	760	2015-09-13 – 2015-09-13
39066167	1	66,936	187	3	593	2015-10-08 – 2016-01-17
37486849	1	41,338	150	7	352	2015-08-06 – 2015-10-24
38604839	5	29,025	133	247	635	2015-09-20 – 2015-10-09
37118579	1	23,024	100	4	100	2015-09-14 – 2015-09-14
39007630	1	42,754	97	3	267	2015-10-15 – 2015-12-20
39060070	1	34,670	80	7	278	2015-10-16 – 2016-01-10
39553907	1	15,720	77	7	690	2015-10-07 – 2016-01-22
39123781	1	14,476	66	5	358	2015-09-25 – 2015-11-05
38327901	1	14,439	60	3	191	2015-10-04 – 2015-10-04
39462182	3	14,920	54	9	104	2015-10-19 – 2015-10-23
39545076	1	13,873	50	3	310	2015-10-23 – 2015-10-23
39575795	2	19,995	50	5	162	2015-10-14 – 2015-12-29
39635330	1	14,734	41	61	525	2015-10-24 – 2015-11-11
39548377	1	11,098	40	5	240	2015-10-23 – 2015-10-23
39610940	1	5,356	29	9	154	2015-08-06 – 2015-10-27
39654530	1	947	1	3	5	2014-03-08 – 2015-10-25
39654407	1	39	0	5	0	2015-10-23 – 2015-11-17

Table 37: Wallets Within First Pareto Front by Balance

Wallet ID	Size	Gain	Balance	Transactions	Volume	Active Period
12632046	2	57,860,572	0	193	891,088	2013-12-04 – 2014-12-05
23321817	16	34,274,045	105	756	2,997,118	2014-11-06 – 2015-11-17
21793989	30	34,061,705	0	218	239,948	2014-02-05 – 2014-12-09

continued ...

continued ...

Wallet ID	Size	Gain	Balance	Transactions	Volume	Active Period
23251293	1	31,879,280	0	63	288,683	2014-06-12 – 2014-12-25
29091650	1	15,954,569	69,370	63	208,312	2013-04-09 – 2016-01-21
12910452	1	15,186,848	21,744	17	21,746	2013-12-19 – 2016-01-21
12632045	2	12,789,658	0	50	199,757	2013-12-06 – 2014-11-28
14051017	13	11,450,484	0	76	96,870	2013-11-13 – 2014-04-11
14292981	2	9,891,130	0	18	115,472	2014-01-16 – 2014-11-28
14273647	1	9,504,826	0	94	81,327	2013-11-22 – 2014-02-25
28105417	4	7,505,229	0	49	71,120	2014-06-09 – 2015-04-02
12611704	1	6,486,948	0	24	281,043	2013-12-11 – 2014-08-01
13871328	8	6,074,934	0	35	141,311	2014-01-24 – 2014-08-25
32686848	1	5,347,658	0	9	29,350	2014-07-21 – 2015-07-03
11854505	6	5,226,839	0	26	33,391	2013-11-30 – 2013-12-06
14299632	5	4,409,110	0	14	98,377	2014-02-04 – 2014-02-26
27121637	1	3,621,029	0	4	100,001	2014-12-08 – 2015-03-09
38771589	1	3,360,968	1	9	15,999	2013-12-24 – 2015-10-11
27122935	1	3,211,184	0	5	88,682	2014-12-08 – 2015-03-09
15512173	1	3,030,484	0	3	47,758	2014-03-22 – 2014-04-02
16002246	1	1,795,050	0	3	30,000	2014-02-18 – 2014-04-19
21601885	1	1,693,860	0	3	14,000	2014-08-09 – 2014-10-25
30553538	1	1,554,253	0	6	8,994	2014-08-05 – 2015-05-25
30776201	1	1,198,340	0	8	13,000	2014-09-18 – 2015-05-29
15772561	1	1,036,649	0	2	9,706	2014-03-15 – 2014-04-11
35757394	1	972,109	0	6	2,400	2013-12-05 – 2015-08-21
21876719	1	862,998	0	4	11,387	2014-09-02 – 2014-11-02

continued ...

continued ...

Wallet ID	Size	Gain	Balance	Transactions	Volume	Active Period
36682013	1	809,448	1,173	7	2,353	2014-08-11 – 2015-09-06
25061929	1	752,271	0	2	13,775	2014-12-31 – 2015-01-20
26022846	1	707,250	0	2	10,000	2014-12-08 – 2015-02-11
25494950	1	672,200	0	2	10,000	2014-12-08 – 2015-01-30
37955477	1	608,000	0	8	10,000	2014-12-08 – 2015-11-13
28455135	1	566,390	0	5	8,464	2014-11-30 – 2015-04-09
29369438	1	453,508	1,740	6	2,122	2015-03-08 – 2015-04-29
25204653	1	433,370	0	2	14,000	2015-01-07 – 2015-01-23
32873502	1	406,140	0	7	4,000	2014-09-09 – 2015-07-06
34249409	1	387,395	2,001	6	4,061	2015-06-04 – 2015-11-18
27839289	1	355,894	0	2	15,400	2015-03-12 – 2015-03-26
27967890	1	319,203	0	2	11,864	2015-03-11 – 2015-03-29
35938206	2	310,660	0	8	35,178	2015-06-03 – 2015-08-24
35956759	1	291,545	0	2	9,772	2015-08-05 – 2015-08-25
37609883	1	244,680	0	2	8,000	2015-07-11 – 2015-09-22
36331101	1	225,141	800	3	830	2015-08-03 – 2015-08-31
38964574	1	212,947	736	9	936	2014-08-21 – 2015-10-16
39049032	1	200,330	0	4	1,000	2014-03-04 – 2015-10-16
38502045	1	153,943	0	4	2,834	2014-10-28 – 2015-10-07
36879198	1	135,383	0	2	9,900	2015-08-12 – 2015-09-09
38276819	1	128,550	0	6	5,000	2015-03-10 – 2015-10-13
39144453	1	111,471	400	8	2,136	2015-03-16 – 2015-12-17
39250814	2	95,606	0	6	600	2014-08-05 – 2015-10-19
39413088	2	94,734	0	8	600	2014-08-05 – 2015-11-02

continued ...

continued ...

Wallet ID	Size	Gain	Balance	Transactions	Volume	Active Period
39541895	2	91,694	0	8	600	2014-08-05 – 2015-11-02
38362929	1	72,911	303	10	403	2015-10-01 – 2015-10-08
38318303	1	72,212	0	2	6,071	2015-06-30 – 2015-10-04
39066167	1	66,936	187	3	593	2015-10-08 – 2016-01-17
36987668	1	61,654	0	2	3,375	2015-07-22 – 2015-09-11
37096007	2	58,377	253	5	760	2015-09-13 – 2015-09-13
37102703	2	56,009	0	6	15,545	2015-09-10 – 2015-09-18
37369725	4	55,869	0	8	2,604	2015-07-23 – 2015-09-18
39007630	1	42,754	97	3	267	2015-10-15 – 2015-12-20
39469331	1	40,574	0	2	402	2014-09-11 – 2015-10-22
39513577	1	39,550	0	3	2,000	2014-12-19 – 2015-10-23
37890563	1	38,070	0	2	2,000	2015-08-11 – 2015-09-27
37129940	1	37,132	0	2	5,580	2015-09-08 – 2015-09-14
39060070	1	34,670	80	7	278	2015-10-16 – 2016-01-10
38597192	1	33,712	0	4	3,040	2015-10-08 – 2015-11-25
38502046	1	32,547	0	2	20,665	2015-10-06 – 2015-10-07
38604839	5	29,025	133	247	635	2015-09-20 – 2015-10-09
39566937	1	25,956	0	4	68	2013-12-02 – 2015-10-23
39575795	2	19,995	50	5	162	2015-10-14 – 2015-12-29
38835410	1	16,031	50	12	354	2015-10-12 – 2015-10-30
39176363	1	15,659	50	8	138	2015-10-18 – 2015-12-08
39462182	3	14,920	54	9	104	2015-10-19 – 2015-10-23
39635330	1	14,734	41	61	525	2015-10-24 – 2015-11-11
39545076	1	13,873	50	3	310	2015-10-23 – 2015-10-23

continued ...

continued ...

Wallet ID	Size	Gain	Balance	Transactions	Volume	Active Period
39548377	1	11,098	40	5	240	2015-10-23 – 2015-10-23
39610940	1	5,356	29	9	154	2015-08-06 – 2015-10-27
39651066	1	4,271	0	2	40	2014-08-15 – 2015-10-25
39613779	1	2,216	7	8	7	2015-10-10 – 2015-12-09
39559183	1	1,699	5	6	14	2015-10-17 – 2015-11-03
39650786	1	1,502	0	7	40	2014-10-17 – 2015-10-25
39629354	1	1,502	0	2	549	2015-07-25 – 2015-10-24
39651071	1	1,152	0	2	7	2013-12-22 – 2015-10-25

Table 38: Wallets Within First Pareto Front by Gain

a.4 MAJOR INFLUENCERS SELECTION

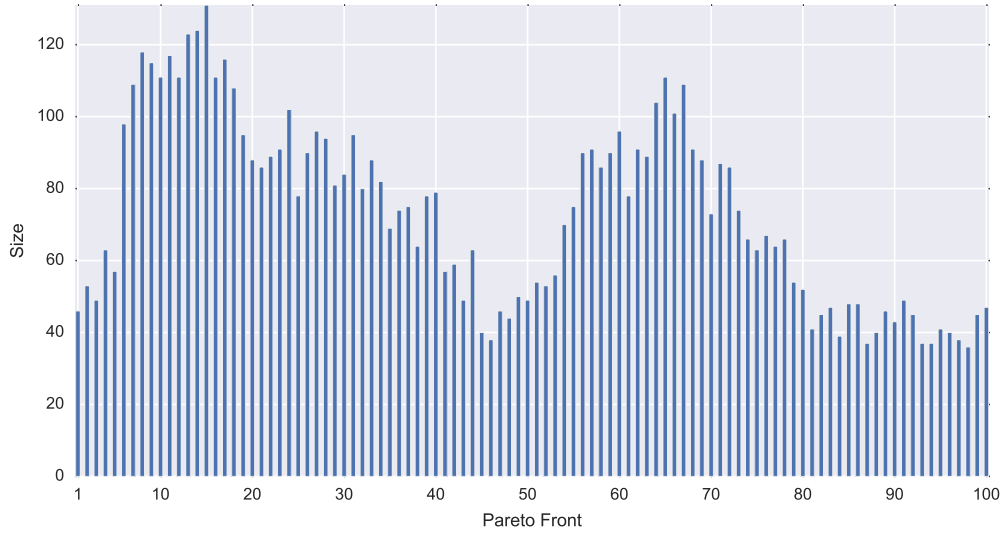


Figure 42: Pareto Fronts by Balance Sizes

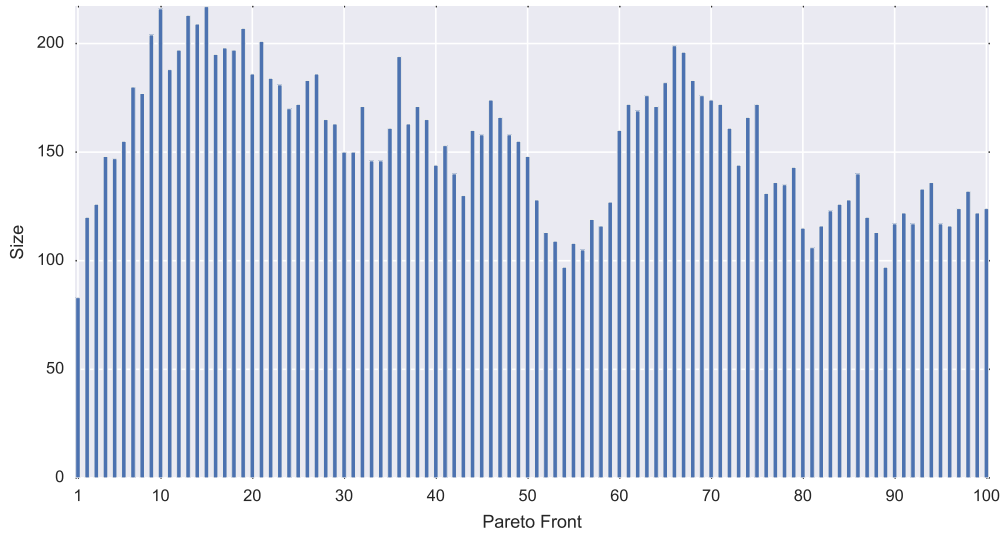


Figure 43: Pareto Fronts by Gain Sizes

a.5 PREDICTIVE MODEL SELECTION

Model	Prediction	Precision	Recall	F1-Score	AUC	Support
Eureqa	0	0.56	0.40	0.47		295
	1	0.50	0.65	0.56		269
	Avg / Total	**0.53**	**0.52**	**0.51**	**0.53**	**564**
SVM	0	0.52	0.43	0.47		295
	1	0.47	0.56	0.51		269
	Avg / Total	**0.50**	**0.49**	**0.49**	**0.50**	**564**
Decision Tree	0	0.63	0.49	0.55		295
	1	0.55	0.68	0.61		269
	Avg / Total	**0.59**	**0.58**	**0.58**	**0.60**	**564**
XGBoost	0	0.70	0.54	0.61		295
	1	0.60	0.75	0.66		269
	Avg / Total	**0.65**	**0.64**	**0.64**	**0.74**	**564**

Table 39: 50% Holdout Model Evaluation Results

Model	Prediction	Precision	Recall	F1-Score	AUC	Support
Eureqa	0	0.51	0.32	0.40		167
	1	0.52	0.70	0.59		172
	Avg / Total	**0.51**	**0.51**	**0.50**	**0.51**	**339**
SVM	0	0.54	0.29	0.38		167
	1	0.52	0.76	0.62		172
	Avg / Total	**0.53**	**0.53**	**0.50**	**0.53**	**339**
Decision Tree	0	0.48	0.61	0.54		167
	1	0.48	0.35	0.41		172
	Avg / Total	**0.48**	**0.48**	**0.47**	**0.49**	**339**
XGBoost	0	0.66	0.57	0.61		167
	1	0.63	0.71	0.67		172
	Avg / Total	**0.64**	**0.64**	**0.64**	**0.69**	**339**

Table 40: 30% Holdout Model Evaluation Results

Model	Prediction	Precision	Recall	F1-Score	AUC	Support
Eureqa	0	0.70	0.13	0.22		54
	1	0.54	0.95	0.69		59
	Avg / Total	**0.62**	**0.56**	**0.47**	**0.54**	**113**
SVM	0	0.49	0.44	0.47		54
	1	0.53	0.58	0.55		59
	Avg / Total	**0.51**	**0.51**	**0.51**	**0.48**	**113**
Decision Tree	0	0.47	0.15	0.23		54
	1	0.52	0.85	0.65		59
	Avg / Total	**0.50**	**0.51**	**0.44**	**0.53**	**113**
XGBoost	0	0.73	0.65	0.69		54
	1	0.71	0.78	0.74		59
	Avg / Total	**0.72**	**0.72**	**0.72**	**0.78**	**113**

Table 41: 10% Holdout Model Evaluation Results

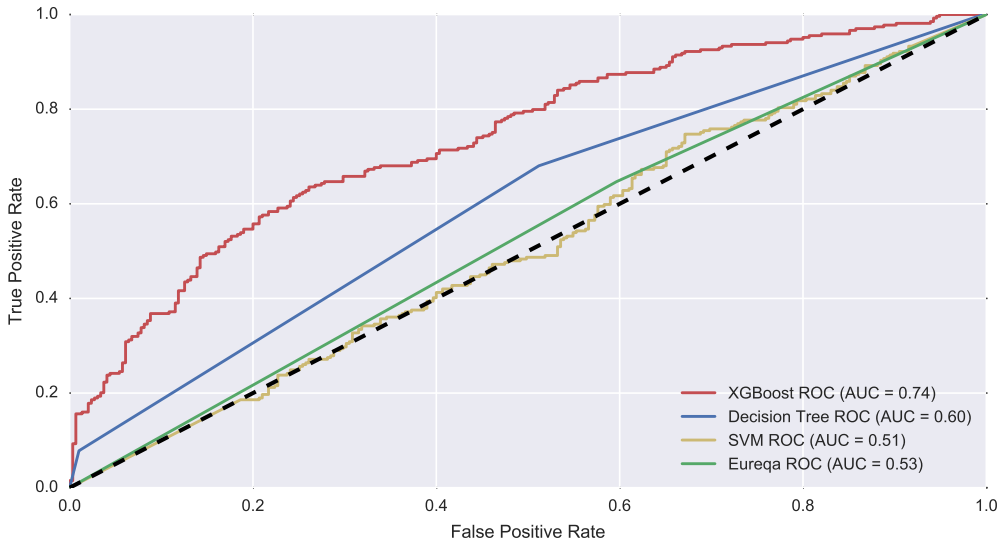


Figure 44: 50% Holdout Model Evaluation ROC Results

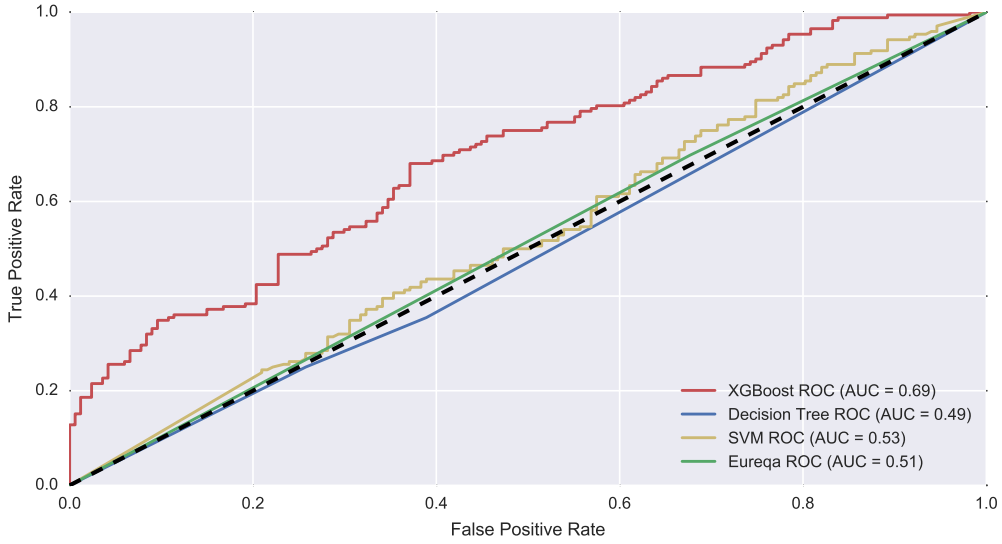


Figure 45: 30% Holdout Model Evaluation ROC Results

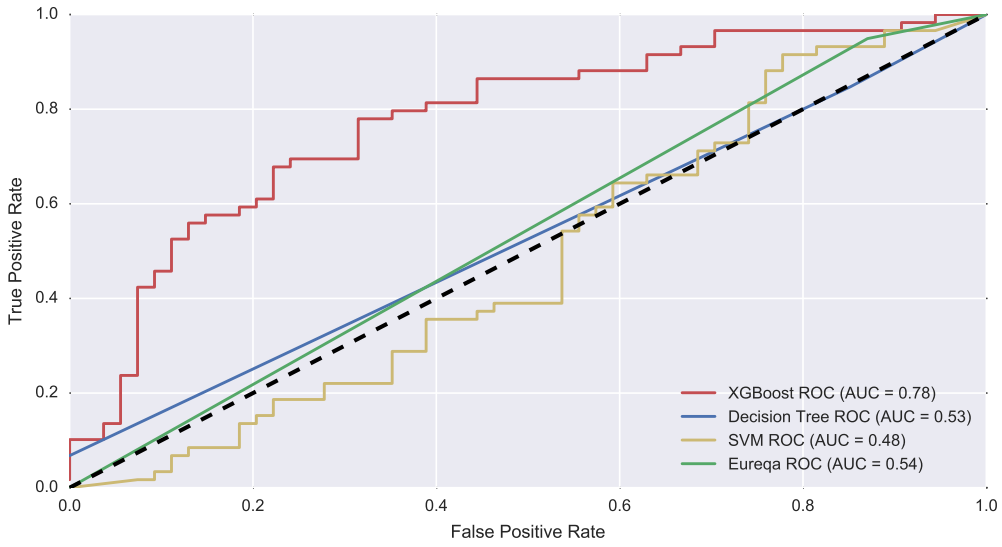


Figure 46: 10% Holdout Model Evaluation ROC Results

a.6 HOLDOUT EXPERIMENTS

Model	Prediction	Precision	Recall	F1-Score	AUC	Support
Monte Carlo	0	0.51	0.57	0.53		295
	1	0.45	0.39	0.42		269
	Avg / Total	**0.48**	**0.48**	**0.48**	**0.48**	**564**
XGBoost	0	0.72	0.66	0.69		295
	1	0.66	0.72	0.69		269
	Avg / Total	**0.69**	**0.69**	**0.69**	**0.73**	**564**

Table 42: 50% Holdout Experiment Results

Model	Prediction	Precision	Recall	F1-Score	AUC	Support
Monte Carlo	0	0.47	0.57	0.52		226
	1	0.45	0.36	0.40		225
	Avg / Total	**0.46**	**0.46**	**0.46**	**0.45**	**451**
XGBoost	0	0.66	0.72	0.69		226
	1	0.69	0.64	0.66		225
	Avg / Total	**0.68**	**0.68**	**0.68**	**0.71**	**451**

Table 43: 40% Holdout Experiment Results

Model	Prediction	Precision	Recall	F1-Score	AUC	Support
Monte Carlo	0	0.46	0.50	0.48		167
	1	0.46	0.42	0.44		172
	Avg / Total	**0.46**	**0.46**	**0.46**	**0.46**	**339**
XGBoost	0	0.62	0.71	0.66		167
	1	0.67	0.58	0.62		172
	Avg / Total	**0.65**	**0.64**	**0.64**	**0.71**	**339**

Table 44: 30% Holdout Experiment Results

Model	Prediction	Precision	Recall	F1-Score	AUC	Support
Monte Carlo	0	0.41	0.50	0.45		109
	1	0.41	0.32	0.36		117
	Avg / Total	**0.41**	**0.41**	**0.41**	**0.43**	**226**
XGBoost	0	0.65	0.69	0.67		109
	1	0.69	0.66	0.68		117
	Avg / Total	**0.67**	**0.67**	**0.67**	**0.73**	**226**

Table 45: 20% Holdout Experiment Results

Model	Prediction	Precision	Recall	F1-Score	AUC	Support
Monte Carlo	0	0.45	0.54	0.49		54
	1	0.48	0.39	0.43		59
	Avg / Total	**0.46**	**0.46**	**0.46**	**0.45**	**113**
XGBoost	0	0.79	0.83	0.81		54
	1	0.84	0.80	0.82		59
	Avg / Total	**0.82**	**0.81**	**0.81**	**0.84**	**113**

Table 46: 10% Holdout Experiment Results

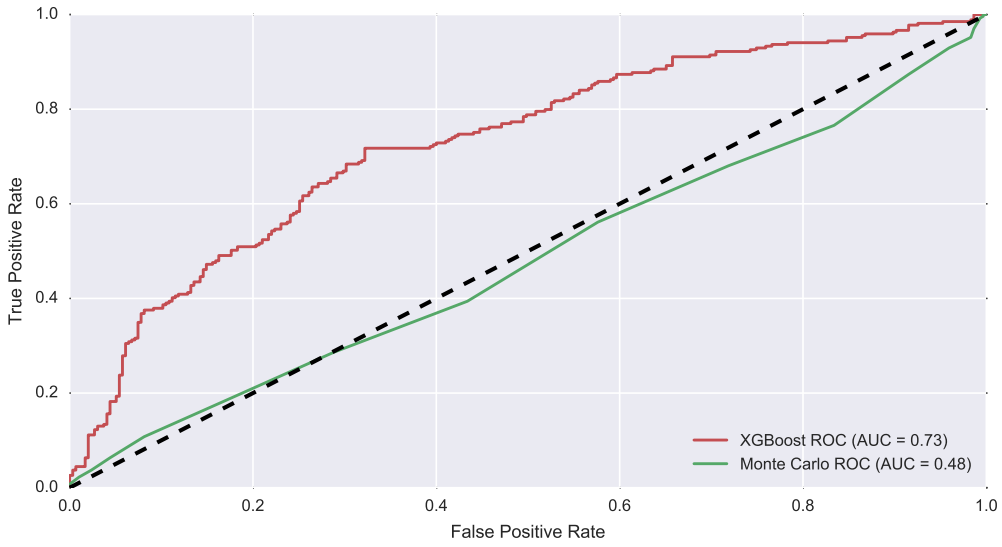


Figure 47: 50% Holdout Experiment ROC Results

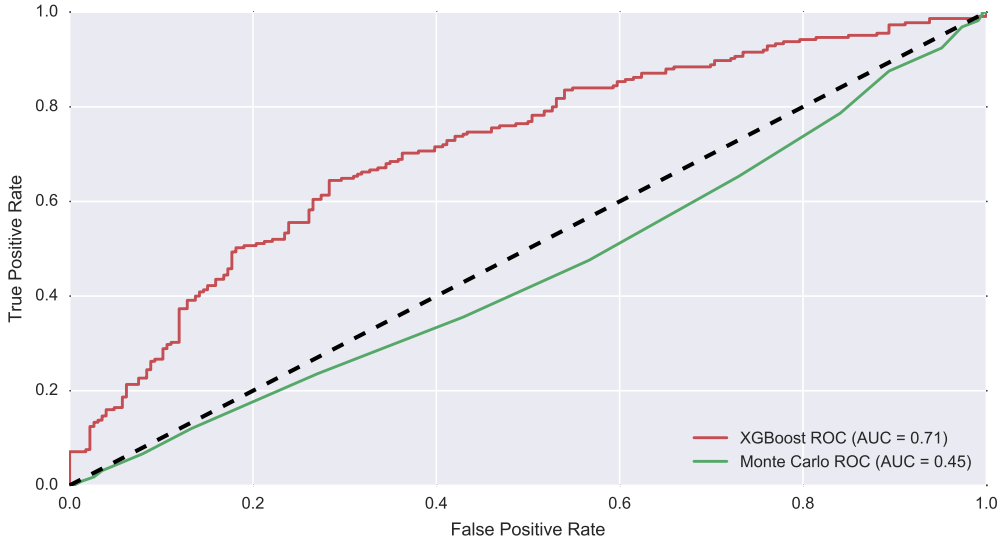


Figure 48: 40% Holdout Experiment ROC Results

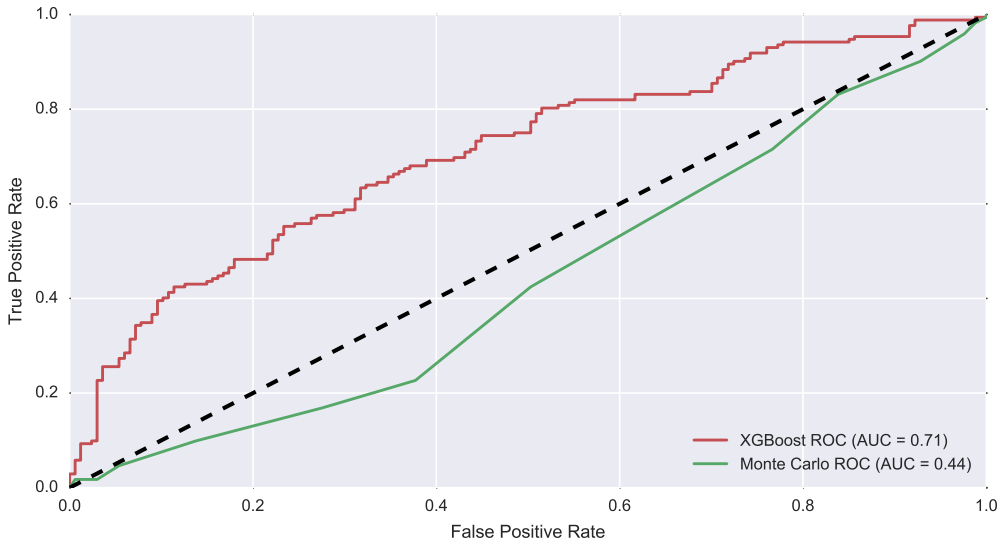


Figure 49: 30% Holdout Experiment ROC Results

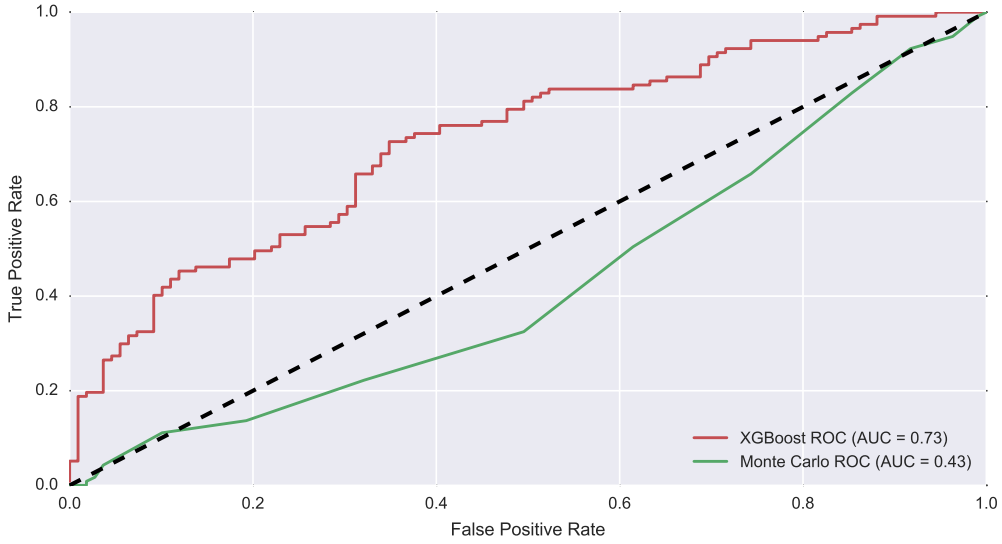


Figure 50: 20% Holdout Experiment ROC Results

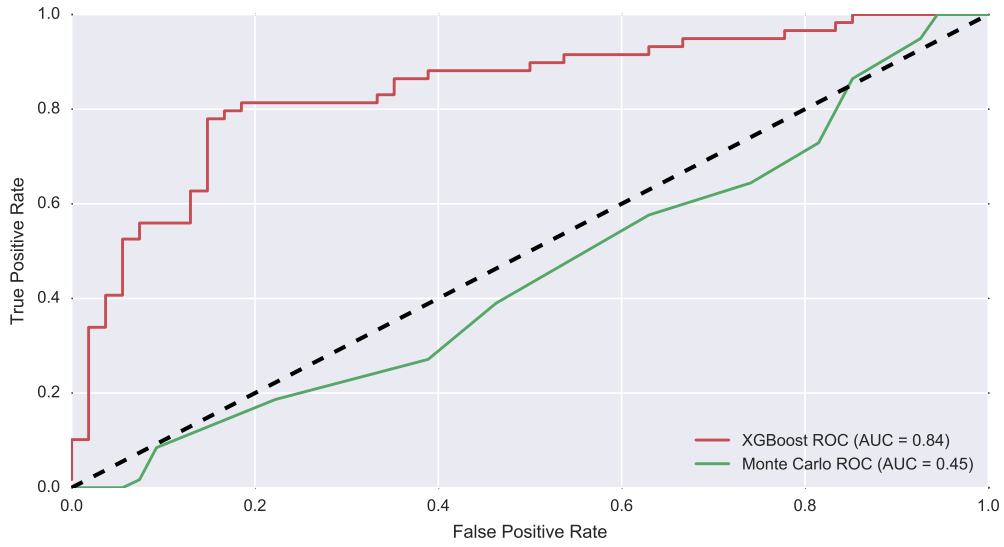


Figure 51: 10% Holdout Experiment ROC Results

a.7 DAILY PREDICTION EXPERIMENTS

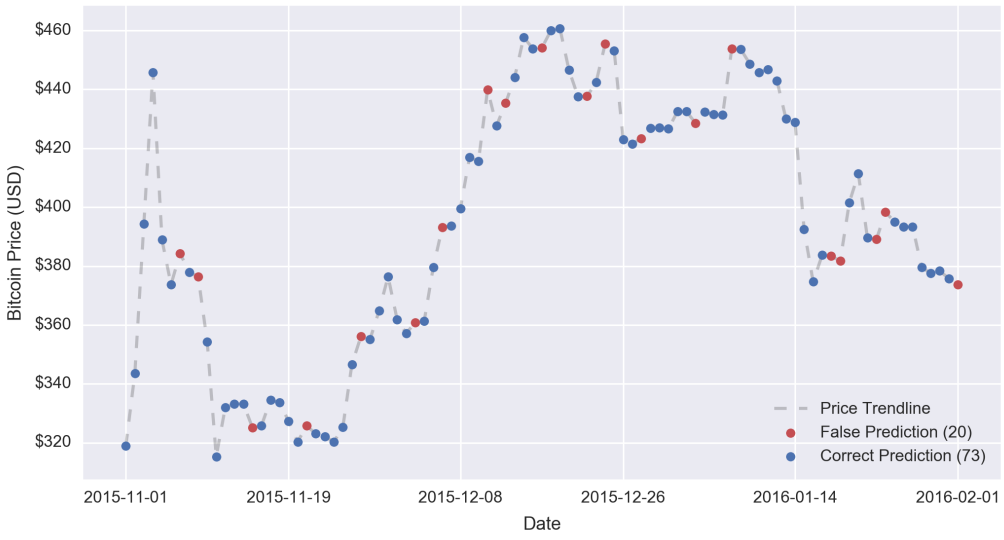


Figure 52: Stretching Window Prediction, Final 3 Months

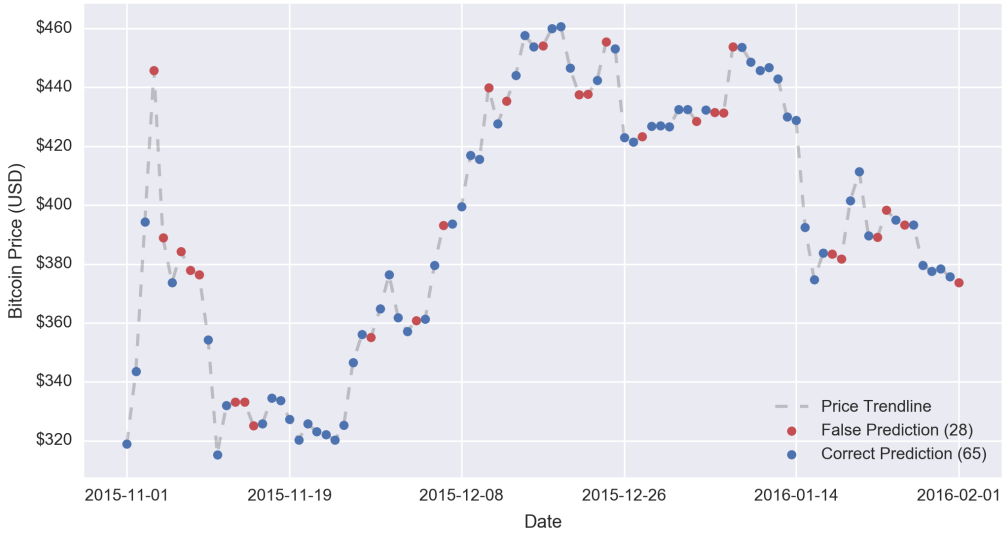


Figure 53: Sliding Window Prediction, Final 3 Months

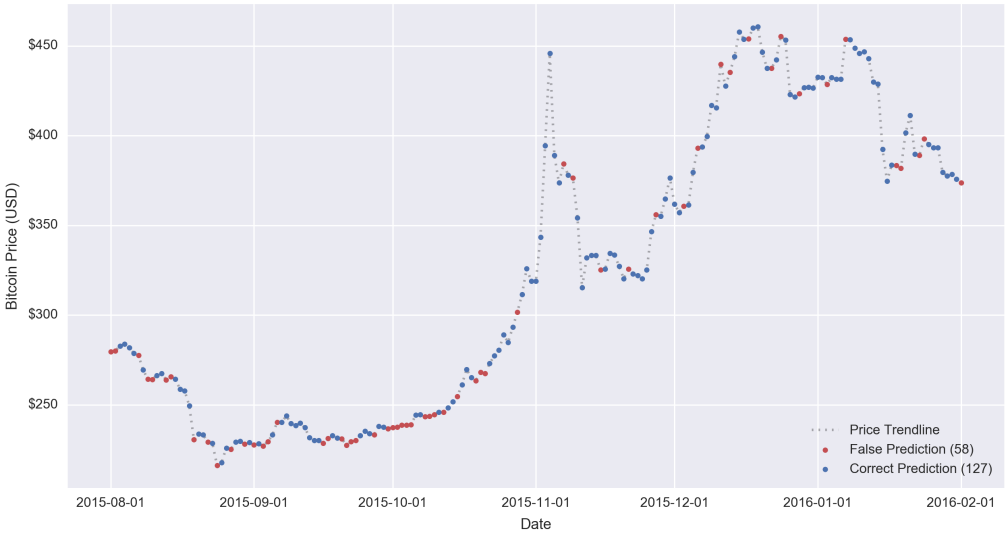


Figure 54: Stretching Window Prediction, Final 6 Months

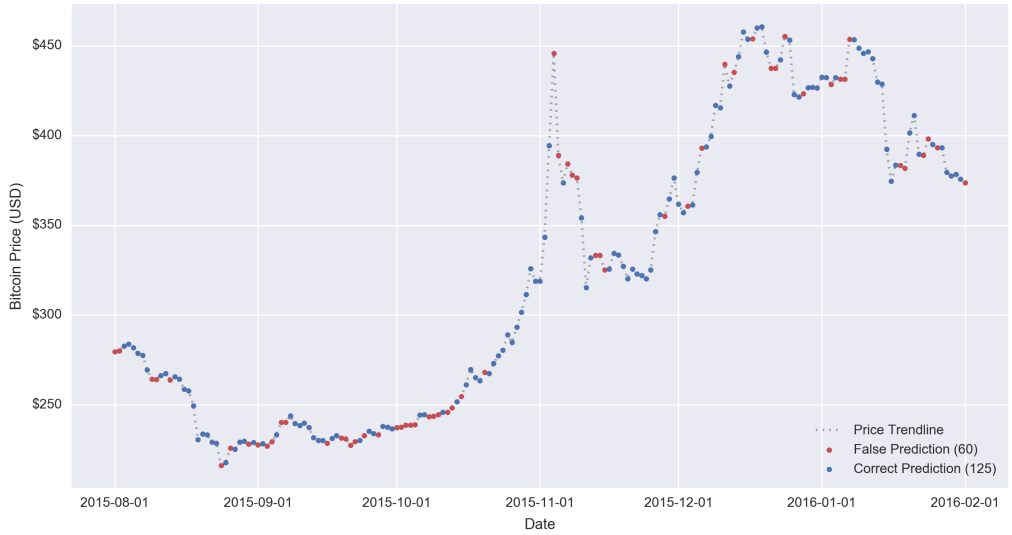


Figure 55: Sliding Window Prediction, Final 6 Months

BIBLIOGRAPHY

[1] Donald E. Knuth. "Computer Programming as an Art." In: *Communications of the ACM* 17.12 (1974), pp. 667–673.

[2] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. http://nakamotoinstitute.org/bitcoin. [Online; accessed Mar. 1, 2016]. Oct. 2008.

[3] Ghassan O Karame, Elli Androulaki, and Srdjan Capkun. "Double-Spending Fast Payments in Bitcoin." In: *Proceedings of the 2012 ACM Conference on Computer and Communications Security*. ACM. 2012, pp. 906–917.

[4] Ghassan Karame, Elli Androulaki, and Srdjan Capkun. "Two Bitcoins at the Price of One? Double-Spending Attacks on Fast Payments in Bitcoin." In: *IACR Cryptology ePrint Archive* 2012 (2012), p. 248.

[5] Jamie Redman. *Bitcoin Pizza Day's Sixth Anniversary!* https://news.bitcoin.com/bitcoin-pizza-day-anniversary. [Online; accessed Aug. 16, 2016]. May 2016.

[6] OANDA. *Historical Currency Converter – OANDA Solutions for Business*. https://www.oanda.com/solutions-for-business/historical-rates-beta/hcc.html. [Online; accessed Aug. 3, 2016]. Aug. 2016.

[7] Claude Sammut and Geoffrey I Webb. *Encyclopedia of Machine Learning*. New York, NY, USA: Springer Science & Business Media, 2011.

[8] Corinna Cortes and Vladimir Vapnik. "Support-vector networks." In: *Machine learning* 20.3 (1995), pp. 273–297.

[9] A Aizerman, Emmanuel M Braverman, and LI Rozoner. "Theoretical foundations of the potential function method in pattern recognition learning." In: *Automation and remote control* 25 (1964), pp. 821–837.

[10] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. "A training algorithm for optimal margin classifiers." In: *Proceedings of the fifth annual workshop on Computational learning theory*. ACM. 1992, pp. 144–152.

[11] William H Press. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge university press, 2007.

[12] Hyeran Byun and Seong-Whan Lee. "Applications of Support Vector Machines for Pattern Recognition: A Survey." In: *Pattern recognition with support vector machines*. Springer, 2002, pp. 213–236.

[13] Alex J Smola, Bernhard Schölkopf, and Klaus-Robert Müller. "The connection between regularization operators and support vector kernels." In: *Neural networks* 11.4 (1998), pp. 637–649.

[14] Kevin P Murphy. *Machine learning: a probabilistic perspective*. Boston, MA, USA: MIT press, 2012.

[15] Max Bramer. *Principles of Data Mining*. Vol. 180. Springer, 2007.

[16] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[17] Wei-Yin Loh. "Classification and regression trees." In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 1.1 (2011), pp. 14–23.

[18] Sreerama K Murthy. "Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey." In: *Data mining and knowledge discovery* 2.4 (1998), pp. 345–389.

[19] James C Bezdek, James M Keller, Raghu Krishnapuram, Ludmila I Kuncheva, and Nikhil R Pal. "Will the real iris data please stand up?" In: *IEEE Transactions on Fuzzy Systems* 7.3 (1999), pp. 368–369.

[20] M. Lichman. *UCI Machine Learning Repository*. http://archive.ics.uci.edu/ml/datasets/Iris. [Online; accessed Jun. 12, 2016]. 2013.

[21] Carolin Strobl, James Malley, and Gerhard Tutz. "An Introduction to Recursive Partitioning: Rationale, Application and Characteristics of Classifica-

tion and Regression Trees, Bagging and Random Forests." In: *Psychological methods* 14.4 (2009), p. 323.

[22] Tianqi Chen. *Story and Lessons Behind the Evolution of XGBoost*. http://homes.cs.washington.edu/~tqchen/2016/03/10/story-and-lessons-behind-the-evolution-of-xgboost.html. [Online; accessed Aug. 24, 2016]. Mar. 2016.

[23] Tianqi Chen and Tong He. "Higgs Boson Discovery with Boosted Trees." In: *Cowan et al., editor, JMLR: Workshop and Conference Proceedings*. 42. 2015, pp. 69–80.

[24] Kaggle. *Caterpillar Winners' Interview: 1st place*. http://blog.kaggle.com/2015/09/22/caterpillar-winners-interview-1st-place-gilberto-josef-leustagos-mario. [Online; accessed Sep. 4, 2016]. Sept. 2015.

[25] Vlad Sandulescu and Mihai Chiru. "Predicting the future relevance of research institutions-The winning solution of the KDD Cup 2016." In: *arXiv preprint arXiv:1609.02728* (2016).

[26] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System." In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2016, pp. 1–10.

[27] Jerome H Friedman. "Greedy function approximation: a gradient boosting machine." In: *Annals of statistics* (2001), pp. 1189–1232.

[28] David Martin Powers. "Evaluation: From Precision, Recall and F-measure to ROC, Informedness, Markedness and Correlation." In: (2011).

[29] Allen Kent, Madeline M Berry, Fred U Luehrs, and James W Perry. "Machine Literature Searching VIII. Operational Criteria for Designing Information Retrieval Systems." In: *American documentation* 6.2 (1955), pp. 93–101.

[30] Asela Gunawardana and Guy Shani. "A Survey of Accuracy Evaluation Metrics of Recommendation Tasks." In: *Journal of Machine Learning Research* 10.Dec (2009), pp. 2935–2962.

[31] Andreas Hotho, Andreas Nürnberger, and Gerhard Paaß. "A Brief Survey of Text Mining." In: *Ldv Forum*. Vol. 20. 1. 2005, pp. 19–62.

[32] David L Olson and Dursun Delen. *Advanced Data Mining Techniques*. Berlin, BE, DE: Springer Science & Business Media, 2008. ISBN: 9783540769170.

[33] CJ van Rijsbergen. "Information Retrieval." In: *The Information Retrieval Group* (1979).

[34] Charles E Metz. "Basic Principles of ROC Analysis." In: *Seminars in nuclear medicine*. Vol. 8. 4. Elsevier. 1978, pp. 283–298.

[35] Tom Fawcett. "An Introduction to ROC Analysis." In: *Pattern recognition letters* 27.8 (2006), pp. 861–874.

[36] Peter A Flach. "The Geometry of ROC Space: Understanding Machine Learning Metrics Through ROC Isometrics." In: *ICML*. 2003, pp. 194–201.

[37] James A Hanley and Barbara J McNeil. "The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve." In: *Radiology* 143.1 (1982), pp. 29–36.

[38] Simon J Mason and Nicholas E Graham. "Areas beneath the relative operating characteristics (ROC) and relative operating levels (ROL) curves: Statistical significance and interpretation." In: *Quarterly Journal of the Royal Meteorological Society* 128.584 (2002), pp. 2145–2166.

[39] Simon Barber, Xavier Boyen, Elaine Shi, and Ersin Uzun. "Bitter to Better—How to Make Bitcoin a Better Currency." In: *Financial Cryptography and Data Security*. Vol. 7397. Springer, 2012, pp. 399–414.

[40] David Chaum, Amos Fiat, and Moni Naor. "Untraceable electronic cash." In: *Proceedings on Advances in cryptology*. Springer-Verlag New York, Inc. 1990, pp. 319–327.

[41] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. "Compact e-cash." In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2005, pp. 302–321.

[42] Jeremiah Bohr and Masooda Bashir. "Who Uses Bitcoin? An exploration of the Bitcoin community." In: *Privacy, Security and Trust (PST), 2014 Twelfth Annual International Conference on*. IEEE. 2014, pp. 94–101.

[43] Bitcoin Talk. *Bitcoin Forum*. https://bitcointalk.org. [Online; accessed Aug. 28, 2016]. 2009.

[44] Dorit Ron and Adi Shamir. "Quantitative Analysis of the Full Bitcoin Transaction Graph." In: *Financial Cryptography and Data Security*. Springer, 2013, pp. 6–24.

[45] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. Boston, MA, USA: MIT Press, 2009. ISBN: 9780262258104.

[46] Fergal Reid and Martin Harrigan. "An Analysis of Anonymity in the Bitcoin System." In: *Security and Privacy in Social Networks*. Springer, 2013, pp. 197–223.

[47] Elli Androulaki, Ghassan O Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. "Evaluating User Privacy in Bitcoin." In: *Financial Cryptography and Data Security*. Springer, 2013, pp. 34–51.

[48] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M Voelker, and Stefan Savage. "A Fistful of Bitcoins: Characterizing Payments Among Men with No Names." In: *Proceedings of the 2013 conference on Internet measurement conference*. ACM. 2013, pp. 127–140.

[49] Michele Spagnuolo, Federico Maggi, and Stefano Zanero. "BitIodine: Extracting Intelligence from the Bitcoin Network." In: *International Conference on Financial Cryptography and Data Security*. Springer. 2014, pp. 457–468.

[50] Dániel Kondor, Márton Pósfai, István Csabai, and Gábor Vattay. "Do the Rich Get Richer? An Empirical Analysis of the Bitcoin Transaction Network." In: *PloS one* 9.2 (2014), e86197.

[51] Robert Dorfman. "A formula for the Gini coefficient." In: *The Review of Economics and Statistics* (1979), pp. 146–149.

[52] Robert I Lerman and Shlomo Yitzhaki. "A note on the calculation and interpretation of the Gini index." In: *Economics Letters* 15.3 (1984), pp. 363–368.

[53] Sorin Solomon and Peter Richmond. "Power Laws of Wealth, Market Order Volumes and Market Returns." In: *Physica A: Statistical Mechanics and its Applications* 299.1 (2001), pp. 188–197.

[54] Oren S Klass, Ofer Biham, Moshe Levy, Ofer Malcai, and Sorin Solomon. "The Forbes 400, the Pareto power-law and efficient markets." In: *The European Physical Journal B* 55.2 (2007), pp. 143–147.

[55] Mark EJ Newman. "Clustering and preferential attachment in growing networks." In: *Physical review E* 64.2 (2001), p. 025102.

[56] Hawoong Jeong, Zoltan Néda, and Albert-László Barabási. "Measuring preferential attachment in evolving networks." In: *EPL (Europhysics Letters)* 61.4 (2003), p. 567.

[57] Jermain Kaminski and Peter Gloor. "Nowcasting the Bitcoin Market with Twitter Signals." In: *arXiv preprint arXiv:1406.7577* (2014).

[58] Johan Bollen, Huina Mao, and Xiaojun Zeng. "Twitter mood predicts the stock market." In: *Journal of Computational Science* 2.1 (2011), pp. 1–8.

[59] Xue Zhang, Hauke Fuehres, and Peter A Gloor. "Predicting stock market indicators through Twitter 'I hope it is not as bad as I fear'." In: *Procedia-Social and Behavioral Sciences* 26 (2011), pp. 55–62.

[60] Martina Matta, Ilaria Lunesu, and Michele Marchesi. "Bitcoin Spread Prediction Using Social And Web Search Media." In: *Proceedings of DeCAT* (2015).

[61] Ifigeneia Georgoula, Demitrios Pournarakis, Christos Bilanakos, Dionisios N Sotiropoulos, and George M Giaglis. "Using Time-Series and Sentiment Analysis to Detect the Determinants of Bitcoin Prices." In: *Social Science Research Network* (2015).

[62] Jaroslav Bukovina, Matus Marticek, et al. *Sentiment and Bitcoin Volatility*. Tech. rep. Mendel University in Brno, Faculty of Business and Economics, 2016.

[63] Malcolm Baker and Jeffrey Wurgler. "Investor Sentiment in the Stock Market." In: *The Journal of Economic Perspectives* 21.2 (2007), pp. 129–151.

[64] Sentdex. *How Sentdex and Sentiment Analysis works*. http://sentdex.com. [Online; accessed Sep. 2, 2016]. 2016.

[65] Clive WJ Granger. "Investigating causal relations by econometric models and cross-spectral methods." In: *Econometrica: Journal of the Econometric Society* (1969), pp. 424–438.

[66] Clive WJ Granger. "Time series analysis, cointegration, and applications." In: *American Economic Review* (2004), pp. 421–425.

[67] Devavrat Shah and Kang Zhang. "Bayesian regression and Bitcoin." In: *Communication, Control, and Computing (Allerton), 2014 52nd Annual Allerton Conference on*. IEEE. 2014, pp. 409–414.

[68] George H Chen, Stanislav Nikolov, and Devavrat Shah. "A latent source model for nonparametric time series classification." In: *Advances in Neural Information Processing Systems*. 2013, pp. 1088–1096.

[69] L. Wasserman. *All of Nonparametric Statistics*. Springer Texts in Statistics. New York, NY, USA: Springer New York, 2006. ISBN: 9780387306230.

[70] Andrew W Lo and A Craig MacKinlay. "Stock market prices do not follow random walks: Evidence from a simple specification test." In: *Review of financial studies* 1.1 (1988), pp. 41–66.

[71] William F Sharpe. "The sharpe ratio." In: *The journal of portfolio management* 21.1 (1994), pp. 49–58.

[72] Isaac Madan, Shaurya Saluja, and Aojia Zhao. *Automated Bitcoin Trading via Machine Learning Algorithms*. 2015.

[73] Alex Greaves and Benjamin Au. "Using the Bitcoin Transaction Graph to Predict the Price of Bitcoin." In: (2015).

[74] Bitcoin Charts. *Bitcoin Charts*. https://bitcoincharts.com/. [Online; accessed Sep. 2, 2016]. 2016.

[75] Michael Fleder, Michael S Kester, and Sudeep Pillai. "Bitcoin Transaction Graph Analysis." In: *arXiv preprint arXiv:1502.01657* (2015).

[76] Mark EJ Newman. "Power laws, Pareto distributions and Zipf's law." In: *Contemporary physics* 46.5 (2005), pp. 323–351.

[77] Reuters. *U.S. holds final auction of bitcoins from Silk Road case*. http://www.reuters.com/article/us-usa-bitcoin-auction-idUSKCN0SU28Q20151105. [Online; accessed May. 2, 2016]. Nov. 2015.

[78] Jorge E Hirsch. "An index to quantify an individual's scientific research output." In: *Proceedings of the National Academy of Sciences of the United States of America* 102.46 (2005), pp. 16569–16572.

[79] Jorge E Hirsch. "Does the h-index have predictive power?" In: *Proceedings of the National Academy of Sciences* 104.49 (2007), pp. 19193–19198.

[80] R.L. Burden and J.D. Faires. *Numerical Analysis*. 9th. Boston, MA, USA: Cengage Learning, 2010. ISBN: 9780538733519.

[81] J.D. Faires and R.L. Burden. *Numerical Methods, 4th*. 4th. Boston, MA, USA: Cengage Learning, 2012. ISBN: 9781285402468.

[82] Coindesk. *From Worst to First: Bitcoin's Price Ends 2015 on Top*. http://www.coindesk.com/bitcoin-price-in-2015-doom-and-gloom-give-way-to-positive-years-end/. [Online; accessed Aug. 28, 2016]. Dec. 2015.

[83] Markus Hartikainen, Kaisa Miettinen, and Margaret M Wiecek. "Constructing a Pareto front approximation for decision making." In: *Mathematical Methods of Operations Research* 73.2 (2011), pp. 209–234.

[84] Coindesk. *Coindesk Bitcoin Price Index Chart*. http://www.coindesk.com/price. [Online; accessed May. 28, 2016]. Feb. 2016.

[85] Wes McKinney. "Data Structures for Statistical Computing in Python." In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman. 2010, pp. 51 –56.

[86] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python.” In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[87] Renáta Dubčáková. “Eureqa: Software Review.” In: *Genetic programming and evolvable machines* 12.2 (2011), pp. 173–178.

[88] M Schmidt and H Lipson. *Eureqa Formulize (Version 1.24)[Computer software]. Nutonian Inc., Cambridge, MA*. 2016.

[89] Krzysztof Pawlikowski. “Do Not Trust All Simulation Studies Of Telecommunication Networks.” In: *International Conference on Information Networking*. Springer. 2003, pp. 899–908.

[90] Makoto Matsumoto and Takuji Nishimura. “Mersenne Twister: a 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator.” In: *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 8.1 (1998), pp. 3–30.

[91] Makoto Matsumoto and Takuji Nishimura. “Dynamic Creation of Pseudorandom Number Generators.” In: *Monte Carlo and Quasi-Monte Carlo Methods* 2000 (1998), pp. 56–69.

[92] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. “The NumPy array: a structure for efficient numerical computation.” In: *Computing in Science & Engineering* 13.2 (2011), pp. 22–30.

[93] Eric Jones, Travis Oliphant, Pearu Peterson, et al. *SciPy: Open source scientific tools for Python*. http://www.scipy.org/. [Online; accessed Sep. 3, 2016]. 2001–.

[94] Zvi Gutterman, Benny Pinkas, and Tzachy Reinman. “Analysis of the Linux Random Number Generator.” In: *2006 IEEE Symposium on Security and Privacy (S&P'06)*. IEEE. 2006, 15–pp.

[95] Aarshay Jain. *Complete Guide to Parameter Tuning in XGBoost*. https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python. [Online; accessed Aug. 14, 2016]. Mar. 2016.