Bachelor thesis
Computer Science

Radboud University

Trading Bitcoin Using Artificial Neural Networks

Author:
Aaron Visschedijk
s4620666

First supervisor/assessor:
Prof. F.W. Vaandrager
F.Vaandrager@cs.ru.nl

Second supervisor/assessor:
G. Schoenmacker
Gido.Schoenmacker@radboudumc.nl

June 25, 2018

Abstract

Bitcoin is a cryptocurrency based on blockchain technology that has attracted investors because of its big price increases. In this thesis we explore the design of algorithms that can make a profit by trading Bitcoin. These trading algorithms often use models to predict the future price. Based on these predictions a decision is made about buying or selling. We used ARIMA and Artificial Neural Network as our models to make these predictions. These models need training data to learn to predict a variable, in this case the price of Bitcoin. We have collected 6 weeks of Bitcoin to US Dollar transactions to use as training data for the models. We also use a small part of this data as test data to see how well our models perform. The performance of our algorithms is measured by their trading return rate and the prediction accuracy of the model that is used. To get a more realistic measure we also take into account transaction costs that come with Bitcoin transactions. By using these measures we find that we can predict the sign of the next price movement with almost 60% accuracy, but we do not achieve any profitable return rates using both ARIMA and Artificial Neural Networks due to transaction costs. We conclude that by using these models one is unlikely to make a profit, but that does not mean the models have no predictive value at all.

Contents

Chapter 1

Introduction

In this thesis we try to find trading algorithms that can make a profit by trading Bitcoin. We use Machine Learning methods to predict the price of Bitcoin and then use these predictions to make decisions about Bitcoin trades. Cryptocurrency trading with Machine Learning is a relatively new and upcoming field of research, but it has many similarities to Stock Market trading using Machine learning which has been researched more.

At a stock market stocks are exchanged. These stocks fluctuate in value. These fluctuations enable stock traders to try to buy stocks for a low price and sell them for a higher price.

Bitcoin is a cryptocurrency. Cryptocurrencies also fluctuate a in value which means that we can also trade in Bitcoin and try to make a profit. In Chapter 2 we will elaborate more on what a cryptocurrency is exactly.

A lot of stock trading happens through algorithms (about 75% in 2016 according to [17]). Because the principle to make profit on both markets is the same, and algorithmic trading is used heavily on the stock market, it would make sense to use algorithmic trading on the cryptocurrency market. A trading algorithm generally tries to predict the future price of the good that is to be traded. Then, based on these predictions, decisions about buying or selling the good are made to try to make a profit. In the simplest form, if the prediction indicates a higher future price, the good should be bought and if the prediction indicates a lower future price the good should be sold.

The profit made by this method of trading is obviously highly dependant on how accurate the predictions are. Another aspect that is important are the trading complications associated with certain goods (e.g. transactions costs, availability, delivery time). Because of these complications we often need more complex decision making rules than buying when the price goes up and selling when the price goes down to still make a profit.

In this thesis we take a look at trading Bitcoin using algorithms. The models we will use to predict the Bitcoin price are ARIMA and Artifi-

cial Neural Networks. Furthermore, we consider transaction costs, Bitcoin's main complication, when making decisions based on our predictions and when reflecting on our trading performance.

In Chapter 2 we will first explain some fundamental ideas that are necessary to understand the method we use. In Chapter 3, we will then discuss the exact method that is used to produce the results. These results will be analyzed in Chapter 4. We compare our results to similar works in Chapter 5 and finally we summarize our conclusions in Chapter 6.

Chapter 2

Preliminaries

2.1 Stock market

At a stock market, stocks are exchanged freely between buyers and sellers for the market price of the stock. Stocks are an instrument to divide ownership of a company. For example, if a person owns 1% of the stocks of a company, he gets 1% of the profit the company makes. As a consequence, if a company makes a large profit, owners of the company's stock are less likely to sell the stock. On the other side, buyers will be more likely to buy the stock. This change in demand and supply will cause the price to go up. Similarly, if the company's profit declines, the price of its stock will also likely decline.

These price changes have attracted stock traders. Traders are not necessarily interested in getting a margin of the profit a company makes. They speculate what the future price of a stock will be, and try to buy low and sell high to make a profit.

2.2 Cryptocurrency

Cryptocurrencies are digital currencies based on cryptographic proofs [13]. They provide an alternative for traditional money. Cryptocurrencies are different from traditional currencies in multiple ways. These differences are identified by [13] as:

- Cryptocurrencies do not need a central authority like a bank or state.
- Cryptocurrencies keep track of the number of units and who owns them.
- Cryptocurrencies define how new units of the currency are created, and who owns them when they are created.
- One can only prove cryptographically who is the owner of a unit of cryptocurrency.

Figure 2.1: The price of Bitcoin in USD (United States Dollar) over time.

- Cryptocurrency transactions can only be initiated by the owner of the units that are to be transacted.
- If two transactions concerning the same units of the cryptocurrency are initiated at the same time, at most one transaction will take place.

Bitcoin was the first form of cryptocurrency [18]. When Bitcoin was released in 2009 it was revolutionary to use blockchain technology to achieve these six characteristics. The blockchain is a large set of data that contains every Bitcoin transaction ever [18]. By 2018 there were hundreds of cryptocurrencies based on blockchain technology [7].

The price of these cryptocurrencies tends to be very volatile when compared to traditional currencies. For example, if we take a look at the price of Bitcoin in US Dollar (Figure 2.1) we can see that in 2017 and early 2018 the price either increases or decreases very fast multiple times [4, 7]. For example, between November 2017 and January 2018 the price went up by 200%. During this period we also observe an increase in the trading volume (Figure 2.2) [4]. This indicates that people use cryptocurrency not only as a currency, but also as an investment. This enables trading in cryptocurrencies similar to trading stocks, where one buys with the intentions to sell at a higher price later on.

2.3 Prediction models

In 2016, about 75% of stock trading was done by algorithms [17]. These algorithms try to predict the future price of the stock and make decisions

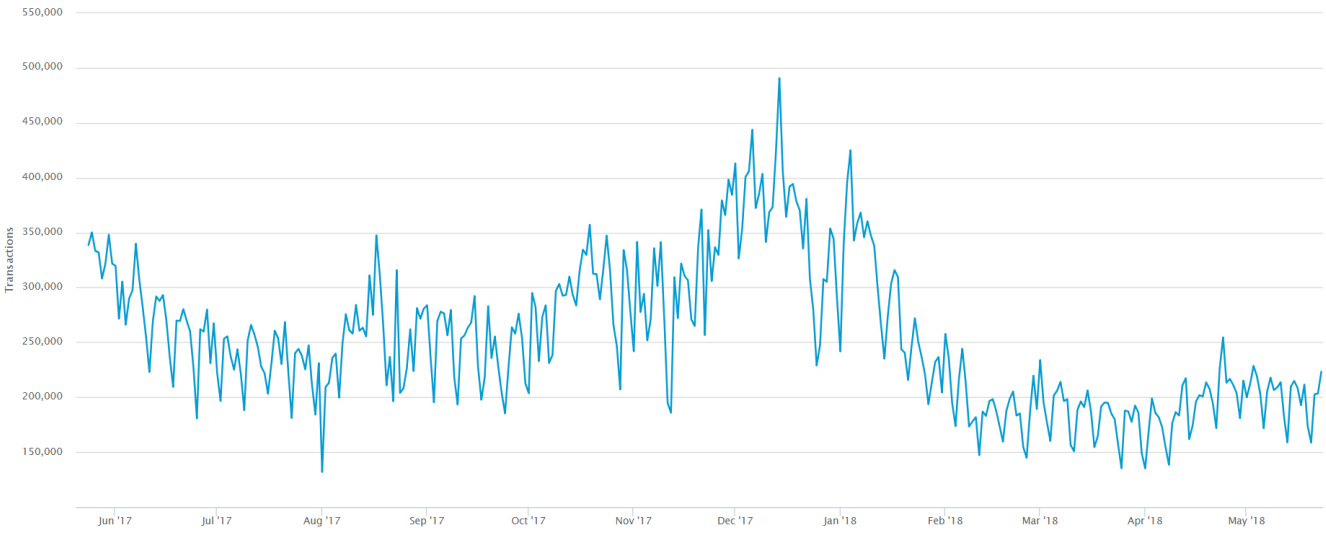


Figure 2.2: The number of Bitcoin transactions per day over time.

based on this prediction. In this thesis we try to use such algorithms to trade Bitcoin. More specifically, we will be using ARIMA and Artificial Neural Network models to predict Bitcoin's future price.

2.3.1 ARIMA

The content about ARIMA in this subsection is based on [2].

ARIMA is a model for time series data prediction. The model has three parameters that each correspond to a different part of the model. In an ARIMA(p, d, q) model, p corresponds to an AR(p) model, d corresponds to the level of integration needed to make the data series stationary and finally, q corresponds to an MA(q) model. We will describe how all these models work exactly later on in this section.

What is meant by stationary data is that the mean, variance and covariance are constant over time. Financial data often shows a trend over time which causes the mean to change over time. If the mean changes over time the data is not stationary. However, we can make it stationary by using the difference between each data point. The d in ARIMA(p, d, q) is the level of differencing between data points needed to make the data stationary.

This process works as follows, instead of using the collected data we use the difference between the data points Y at times t:

$$\Delta Y_t = Y_t - Y_{t-1}$$

If a data series requires a higher level of differencing to become stationary one can just apply the same equation to the resulting ΔY. For example if

$d = 2$:

$$\Delta\Delta Y = \Delta Y_t - \Delta Y_{t-1}$$

An AR(p) model is an auto regressive model that considers the last p values of the variable that is to be predicted. The prediction of a variable at time t with an AR(p) model is given by:

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + ... + \phi_p Y_{t-p} + u_t$$

where Y_t is the value at time t, and ϕ is a value between -1 and 1 that is optimized to get the best fit for the training data. This fit is then used for predictions using the test data.

An MA(q) model is a moving average model that considers the last q error terms of the variable to be predicted. The prediction of a variable at time t with a MA(q) model is given by:

$$Y_t = u_t + \theta_1 u_{t-1} + \theta_2 u_{t-2} + ... + \theta_q u_{t-q}$$

where Y_t is the value at time t, u_t is the error term at time t, and finally θ is a value between -1 and 1 that needs to be fitted to get the best prediction.

We can put the AR(p) and MA(q) models together to get an ARMA(p, q) model. The prediction equation is then simply given by adding the predictions of the two models together:

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + ... + \phi_p Y_{t-p} + \theta_1 u_{t-1} + \theta_2 u_{t-2} + ... + \theta_q u_{t-q} + u_t$$

If we then take the difference between the Y_t variables we get an ARIMA model.

2.3.2 Neural networks

The explanation of neural networks in this section will be based on [8]. Furthermore, we do not aim to explain neural networks in great detail but we will give a brief explanation. For a more detailed explanation, see [8].

An artificial neural network is a useful model for self learning computer programs. It is a network that consists of several layers; an input layer, hidden layer(s) and an output layer. These layers are made out of artificial neurons. These neurons take in inputs and produce a single output. In Figure 2.3 you can see an example of a neural network, in this case the network has an input layer with three neurons, a hidden layer with four neurons and an output layer with three neurons.

Each artificial neuron has weights for all its inputs. When it gets input it multiplies each input by their corresponding weight and takes the sum of all these multiplications and adds a bias. After that, a function is often used

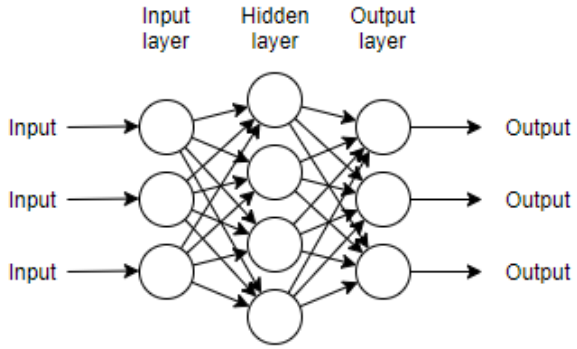


Figure 2.3: A neural network with a single input layer, output layer and hidden layer. The circles represent neurons. This network has 3 input neurons that each take a number as input. The neurons calculate their output, which is also a number, and send it to the neurons that they are connected with. These connections are indicated by the arrows. After all the neurons have processed their input the network produces 3 output numbers through the 3 output neurons.

on this sum to get a result between some desired bounds (usually 0 and 1). For example the *sigmoid function*, defined as:

$$S(x) = \frac{1}{1 + e^{-x}}$$

The result is then the output of the neuron. Formally:

$$output = function(input_0 * weight_0 + ... + input_n * weight_n + bias)$$

If this neuron is in the output layer, the output is an output of the neural network. Otherwise it is the input of all connected neurons in the next layer.

If we want to make the network do something useful we need to train it. To do this we give the network inputs while we know the desired output. We then compare the output of the network to the desired output using a cost function. This is also known as supervised learning [15]. The cost function should return a low value if the output is very similar to the desired output and a high value if the output is very different from the desired output. The neural network aims to minimize the cost function but it can only modify the weight and bias values.

To find this minimum we could take the derivative of the cost function with respect to all the weights and biases in the network. However, the number of weights grows too fast to be able to calculate this derivative. This is why we use a different approach. We calculate the partial derivative of the cost function with respect to each weight or bias. We then change each weight or bias in the direction of its partial derivative in small steps until we find a local minimum.

This way the neural network minimizes the cost function to at least a local minimum. By minimizing the cost function the neural network gets better at giving the desired output for a certain set of inputs.

Chapter 3

Research

In this chapter we will first introduce the main problem. Then we will formalize this problem, making it easier to solve. This will also make analyzing the performance of our solutions easier. Then we describe how to collect the data we need. Lastly, we go over the models that use the collected data to try to give a solution to our problem.

3.1 Problem

The goal of this research is to design algorithms that consistently make a profit by trading Bitcoin. To do this we need to be able to predict the price of Bitcoin. After we have made our predictions we will feed them into a program that will make decisions.

Predicting the exact price is very hard. This is why we simplify the problem; we only try to predict whether the price will increase, decrease or stay the same within certain thresholds. Formally:

$$m[t] = \begin{cases} \textit{Down}, \text{ if } p[t]/p[t-1] < T^- \\ \textit{Stay}, \text{ if } p[t]/[t-1] \geq T^- \text{ and } p[t]/p[t-1] \leq T^+ \\ \textit{Up}, \text{ if } p[t]/p[t-1] > T^+ \end{cases}$$

where t is the chronological index of the trade. This means that the first trade to be recorded has $t = 0$, the second $t = 1$ and so on. We will refer to t as time but note that absolute time between successive transactions may vary. Furthermore, $m[t]$ is the movement of the price of Bitcoin at time t and $p[t]$ is the price of Bitcoin in USD (United States Dollar) at time t. T^- and T^+ are the lower and upper threshold respectively.

It may seem obvious to use absolute thresholds instead of fractional thresholds because our data does not show any significant relation between the price of Bitcoin and the change in Bitcoin price as can be seen in Figure 3.1. However, when choosing the thresholds we want to choose them such

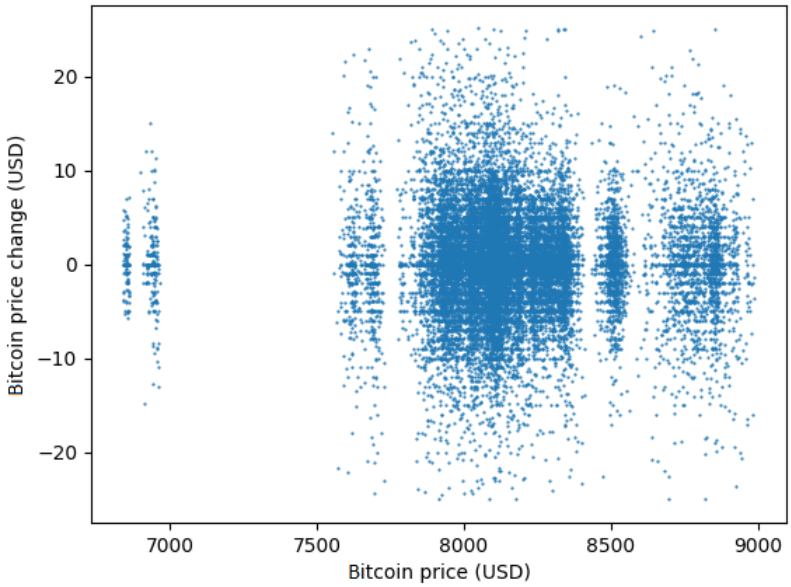


Figure 3.1: Scatter plot of the absolute Bitcoin price in USD (United States Dollar) change and the Bitcoin price in USD.

that we can make the most profitable decisions. Binance [3], the cryptocurrency exchange used in this project takes 0.1% of the amount of currency exchanged as a transaction fee. Consequently, if we want to make a profit the price should go up by at least 0.1% to make a profit. This is why we use fractional thresholds.

We put the upper threshold at 1.001. We use this value because if our prediction is right, we will make a profit in the next time-step if we invest. We put the lower threshold at 0.999. We choose this value because if we already have invested and we assume our prediction is right, we know that if our prediction is *Down*, we are better off selling.

We will also consider the classification problem using only two classes, *Up* and *Down*, while ignoring the transaction costs. This will give us some insight on the impact these fees have on our trading return rates. Formally:

$$m[t] = \begin{cases} \textit{Down}, \text{ if } p[t]/p[t-1] < 1 \\ \textit{Up}, \text{ if } p[t]/p[t-1] > 1 \end{cases}$$

Note that no change at all ($p[t]/p[t-1] = 1$) is not included in this classification problem. This choice was made because if there is no change at all it does not matter whether we decide to buy or sell because we do not consider transaction costs.

Furthermore, we will also try to predict the exact future price of Bitcoin. This will give us more insight in what the models predict exactly and how this compares to the real data.

3.2 Data Collection

To solve the classification problem, we will need to collect data about Bitcoin. During this research, we use Binance [3] to collect this data.

The data collected and used consists of the highest price, lowest price and last price (all in USD) in a 30 second interval from 12th of April, 2018 until the 22nd of April, 2018. After the 22nd of April, every Bitcoin to USD transaction (and thus also every USD to Bitcoin transaction) was recorded in terms of its price, quantity and time of trading until the 1st of June, 2018. The collected data will be used to train and test various models for our classification problem.

This data is not available from the blockchain because Binance does not put every transaction on the blockchain. Some of the transactions are offset in the Binance exchange to save costs. Another issue is that the blockchain does not record that what was used to buy the Bitcoin.

We have also been provided with a data set containing almost 6 million transactions between Ethereum and Bitcoin from the 26th of March until the 5th of April collected from Binance. We will use this data set only for the two-class prediction problem because the price does not change fast enough to get any data outside of the same class in the three-class prediction problem.

3.3 ARIMA

The first model we will use to try to predict the price of Bitcoin is ARIMA. Because ARIMA is using previous values to predict a value the output does not tend to be very interesting when we use very small time-steps. The output then just stays really close to the previous values so we cannot really derive any conclusions from the predictions. Thus, we should increase the time-step to get more variance in the previous values and make our output more interesting.

We also want to find the parameters that give us the best model for predicting Bitcoin. To find these parameters we use the box-jenkins approach [2].

To do this we first plot the autocorrelation function (ACF) and the partial autocorrelation (PACF) function of the data. We can now see if the data is stationary or not. If this is not the case, we take the logarithm of the data and then the first difference. After that we check again if the ACF and PACF show that the data is stationary.

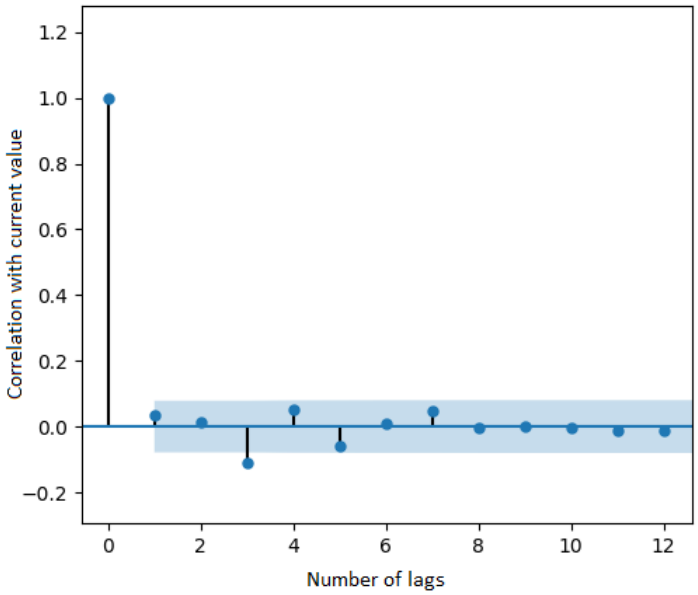


Figure 3.2: Autocorrelation plot of the first level differenced data. The x-axis indicates how many many lags are considered when determining the correlation with the current value. The y-axis shows the correlation with the current value.

If the data is stationary we use the ACF and PACF to estimate what would be good p and q parameters for our model. We can do this because the ACF and PACF shows how much a previous result correlates with newer results.

We find that the highest significant lag is 7 by looking at Figure 3.2, so p and q are both ≤ 7. After testing we find the optimal ARIMA model is ARIMA$(4, 1, 2)$. We will be using this model to make predictions.

An input for this ARIMA model consists of 5 inputs for the AR model and 2 inputs for the MA model. Note that there are 5 inputs for the AR model because in the process of differencing we end up with 4 inputs. Here is an example of how ARIMA predicts a value:

Inputs for AR model	$8127.0, 8136.84, 8131.46, 8092.0, 8113.21$
Inputs AR model after differencing	$9.84, -5.38, -39.46, 21.21$
Inputs for the MA model	$-33.42, 18.2$
Values found for ϕ	$-0.1753, 0.5477, 0.0380, 0.0369$
Values found for θ	$0.0761, -0.5139$
Prediction by ARIMA$(4, 1, 2)$ model	$Y_t = Y_{t-1} + (\phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \phi_3 Y_{t-3} + \phi_4 Y_{t-4} + \theta_1 u_{t-1} + \theta_2 u_{t-2}) = 8113.21 + (9.84 * -0.1753 - 5.38 * 0.5477 - 39.46 * 0.0380 + 21.21 * 0.0369 - 33.42 * 0.0761 + 18.2 * -0.5139) = 8095.93$

3.4 Neural networks

Now we will try to use neural networks to predict the price of Bitcoin. During this research project we used the Keras API [12]. Keras provides an easy to use interface for neural networks in Python on top of Tensorflow [20]. Tensorflow is a library by Google that includes neural network implementations.

The results of the neural network are strongly dependent on the input. This makes sense because this is the only information the neural network gets to give a prediction. We will be using the following inputs calculated from the raw transaction data:

Input type	Example input value
Previous price movement	1
Number of *Down* movements in the last 50 measuring points	15
Number of *Stay* movements in the last 50 measuring points	18
Number of *Up* movements int the last 50 measuring points	17
Maximal length of consecutive *Down* movements in the last 50 measuring points	5
Maximal length of consecutive *Stay* movements in the last 50 measuring points	5
Maximal length of consecutive *Up* movements in the last 50 measuring points	4
Number of trades since the last measuring point	127
Second in current minute	29
Minute in current hour	24
Hour in current day	14
Day in current week	4
Day in current month	13
Expected output	*up*

Note that the last row in this table is not an input but the expected output.

The results of the neural network also depend partially on the structure of the neural network. There are no sharply defined methods to find the optimal structure but there are some guidelines to find a competent neural network structure. We will start with 2 hidden layers and try to add more to see if this increases performance.

We also have to determine the number of neurons in each hidden layer. We do not want too many neurons because this will cause overfitting. Another problem that could occur is that the network structure becomes too big to train. Because neurons bring more weights to the network that need adjusting, training the network becomes more complex when you add more neurons. We also do not want too few neurons because then the network will not be able to learn very well.

There are a few rules of thumb identified by [9] about the number of neurons in a hidden network layer:

- The size of a hidden layer should be between the size of the input layer and the output layer
- The size of a hidden layer should be 2/3 of the size of the input layer

+ the size of the output layer

- The size of a hidden layer should be smaller than twice the size of the input layer

Because we have thirteen inputs and three outputs, the size of the two hidden layers should be between thirteen and three neurons according to the first rule of thumb. According to the second rule our input layer should be consisting of $\frac{2}{3} \times 13 + 3 = 11$ neurons. Eleven neurons is also in line with the last rule of thumb.

This gives us the following neural network structure: 13 input neurons in the input layer, two hidden layers consisting both of 11 neurons and finally an output layer consisting of 3 neurons. This is a good starting point, but we will try several other structures with more hidden layers to see if performance improves or not.

3.5 Decision making

Given our models' predictions, we need to make profitable decisions. We have already decided that our thresholds T^+ and T^- are 1.001 and 0.999 respectively. The values of these thresholds are important to the decision we make given our predictions because these values give meaning to the predicted classes:

- *Up*, price is predicted to go up by at least more than 0.1%
- *Same*, price is predicted to stay the same or go up by at most 0.1% or go down by at most 0.1%
- *Down*, price is predicted to go down by at least more than 0.1%

Furthermore, we also need to consider whether or not we have already invested or not when making a decision. What follows is a table of what the system does in what situation and the reason behind it.

Prediction	Invested	Decision	Reasoning
Up	No	Buy	According to our prediction the investment will increase more in value than the cost of investing, thus we will make a profit
Up	Yes	Wait	According to our prediction the value of our investment will increase
Same	No	Wait	According to our prediction we will not make a profit at the next time-step if we invest
Same	Yes	Wait	According to our prediction the cost of selling our investment will at least be higher than the maximum loss when not selling it
Down	No	Wait	According to our prediction the value of our investment will decline
Down	Yes	Sell	According to our prediction our investment will lose more value than the cost of selling

3.6 Measuring performance

We are using two performance tests in this project. The first performance test is measuring a model's prediction accuracy. The second performance test is how high the return rate is when the model uses its predictions to trade.

Our first performance test will be measuring the model's accuracy by simply comparing the predicted movement to the actual price movement. We will also consider the prediction accuracy for every class of the classification problem defined in Section 3.1. We do this because it will give us more insight in what the model is actually good at. For example, we could have model that when the price goes down it always predicts *Down*, when the price stays the same it predicts *Stay* 50% of the time and when the price goes up it is always wrong. This model will have an accuracy of roughly 50% but will still not be very useful because of always predicting one class wrong. Furthermore, the actual distribution of the price changes will be

plotted to compare this to the distribution of the predictions.

The second performance test will be measuring the return rate of the the model while trading. We will consider both the return rate while ignoring the Binance fees and the return rate without ignoring the Binance fees.

Chapter 4

Results

In this chapter we will go over the performance of our models. First, we take a look at ARIMA's results on the three-class classification problem. Then we will go over it's performance while it tries to predict the exact future price of Bitcoin. Then we take a look at the performance of Neural Networks on the three-class classification problem and the exact prediction of the price. We will also analyze the results of the models' performance on the two-class classification problem. Finally, we will compare the results of different models and different problems to come to conclusions about to what extend we have solved the main problem.

4.1 ARIMA

In this section we will discuss the results of our ARIMA$(4, 1, 2)$ model. When using the ARIMA model the time between each measure was increased significantly, otherwise the model would only predict that the price would stay the same. This is due to ARIMA usage of past values of the variable that it is trying to predict, which often results in predicting a value close to its predecessors. The time-step that was used to prevent ARIMA from predicting *Stay* every time is 1 hour and 40 minutes between each measurement.

4.1.1 Predicting classes

We get the following results, trying to solve the classification problem using ARIMA$(4, 1, 2)$:

Measure	Result
Prediction accuracy	31.452%
Prediction class distribution (*Down*, *Stay*, *Up*)	16.129%, 64.516%, 19.355%
Test data class distribution (*Down*, *Stay*, *Up*)	43.548%, 11.290%, 45.161%
Return rate without Binance fees	0.923
Return rate with Binance fees	0.901
Number of trades	24

The corresponding confusion matrix is given by:

		Correct class		
		Down	*Stay*	*Up*
Predicted class	*Down*	11.290%	0.000%	4.839%
	Stay	26.613%	8.871%	29.032%
	Up	5.645%	2.419%	11.290%

As we can see our prediction accuracy is not very high. This is mainly due to the strong bias towards the *Stay* class from ARIMA. ARIMA is not a model for classification and does not get punished necessarily for predicting the wrong class, it gets punished for how far off its exact predictions are from reality. This results in a lot of predictions around 0 because this is the mean change of the Bitcoin price. The small size of the *Stay* class in the test data can be explained by the large time-step that was used. The chance that a fluctuating price is still within 0.1% range of the previous value obviously gets smaller when larger time-steps are used. Furthermore, the return rates for ARIMA are both negative which is a logical consequence of a low prediction accuracy.

4.1.2 Predicting exact values

Now we will take take a look at the exact predictions of the ARIMA$(4, 1, 2)$ model. The model had a mean squared error of 4975.04. In Figure 4.1 we can see how the ARIMA predictions are just behind the actual price movement most of the time. In Figure 4.2 we can see both the actual price change distribution and the predicted price change distributions. We can clearly see ARIMA's bias towards no price change.

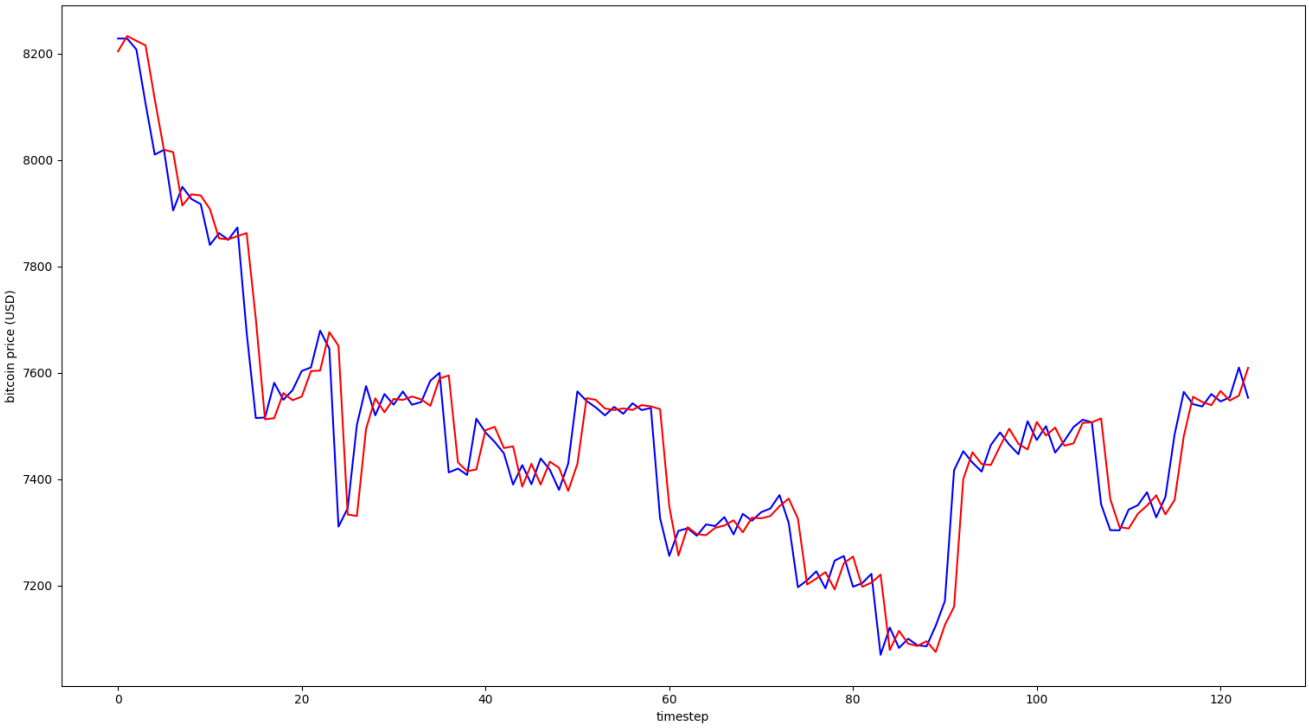
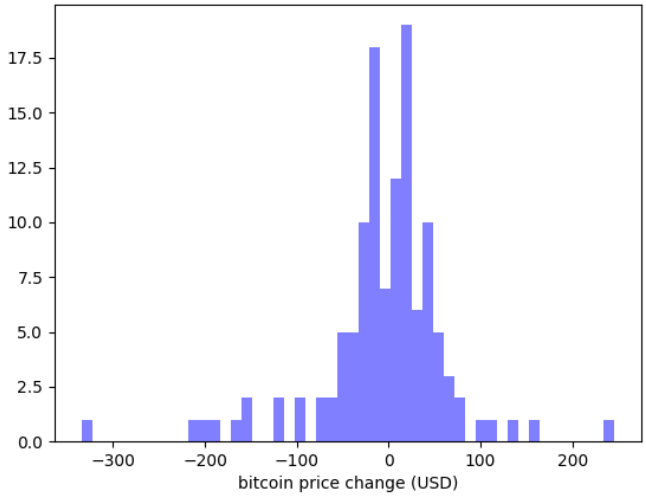
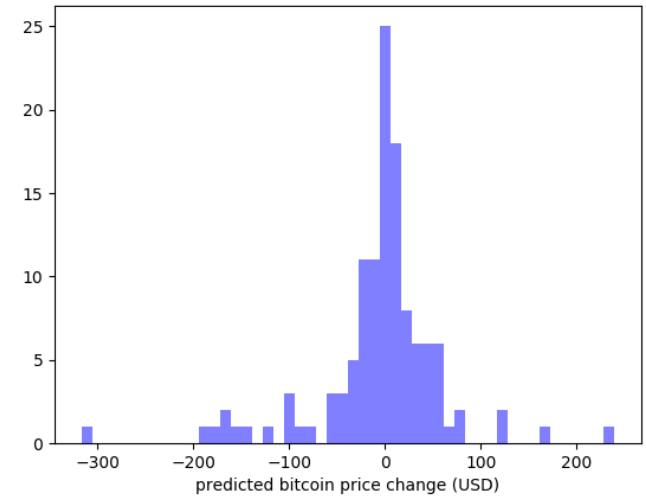


Figure 4.1: ARIMA predictions of the Bitcoin price in USD (United States Dollar) plotted over time in time-steps of 1 hour and 40 minutes (red) and Bitcoin's actual price movement in USD in time-steps of 1 hour and 40 minutes (blue).



(a) Bitcoin's actual price change distribution in a histogram.



(b) ARIMA's predicted price change distribution in a histogram.

Figure 4.2: Comparison between the actual price change distribution of Bitcoin and ARIMA predicted price change distribution. The price changes are recorded in time-steps of 1 hour and 40 minutes. Figure a shows the actual price distribution in a histogram and figure b shows ARIMA's prediction distribution in a histogram

4.2 Neural networks

In this section we will discuss the results of the Neural Network model. Similar to ARIMA we will first describe the results for the classification problem and then the results for predicting an exact value. While using the Neural Network model we used time-steps of 12.5 minutes.

4.2.1 Predicting classes

We get the following results, trying to solve the classification problem using a Neural Network:

Neural network results using two hidden layers:

Measure	Result
Prediction accuracy	39.804%
Prediction class distribution (*Down*, *Stay*, *Up*)	16.249%, 35.551%, 48.201%
Test data class distribution (*Down*, *Stay*, *Up*)	33.151%, 32.606%, 34.242%
Return rate without Binance fees	0.980
Return rate with Binance fees	0.819
Number of trades	179

The corresponding confusion matrix is given by:

		Correct class		
		Down	*Stay*	*Up*
Predicted class	*Down*	6.761%	4.471%	5.016%
	Stay	12.323%	13.522%	9.706%
	Up	14.068%	14.613%	19.520%

Neural network results using three hidden layers:

Measure	Result
Prediction accuracy	38.931%
Return rate without Binance fees	0.992
Return rate with Binance fees	0.870
Number of trades	132

Neural network results using four hidden layers:

Measure	Result
Prediction accuracy	38.713%
Return rate without Binance fees	0.934
Return rate with Binance fees	0.804
Number of trades	149

During testing we tried adding more layers but this did not significantly affect prediction accuracy. This is why we have decided to not describe all of the results in as much detail as the results of the network that performed best.

We can see that at a maximum of 39.804% the prediction accuracy is still not very high. However, because it is a higher percentage than the size of every class in the test data distribution, we at least know that the model is able to predict better than just random guessing.

The return rates are still both negative. The higher difference between the return rate without Binance fees and the return rate with Binance fees can be explained by the higher number of trades. Every time we trade our Bitcoin for USD or the other way around Binance takes 0.1%

4.2.2 Predicting exact values

Now we will take a look at the exact predictions of the Neural Network. The model had a mean squared error of 588.10. In Figure 4.3 we can see how the predictions is always just behind the actual movement. In Figure 4.4 we can see both the actual price change distribution and the predicted price change distributions. We can see that the predicted change distribution is way more centered than the actual price distribution.

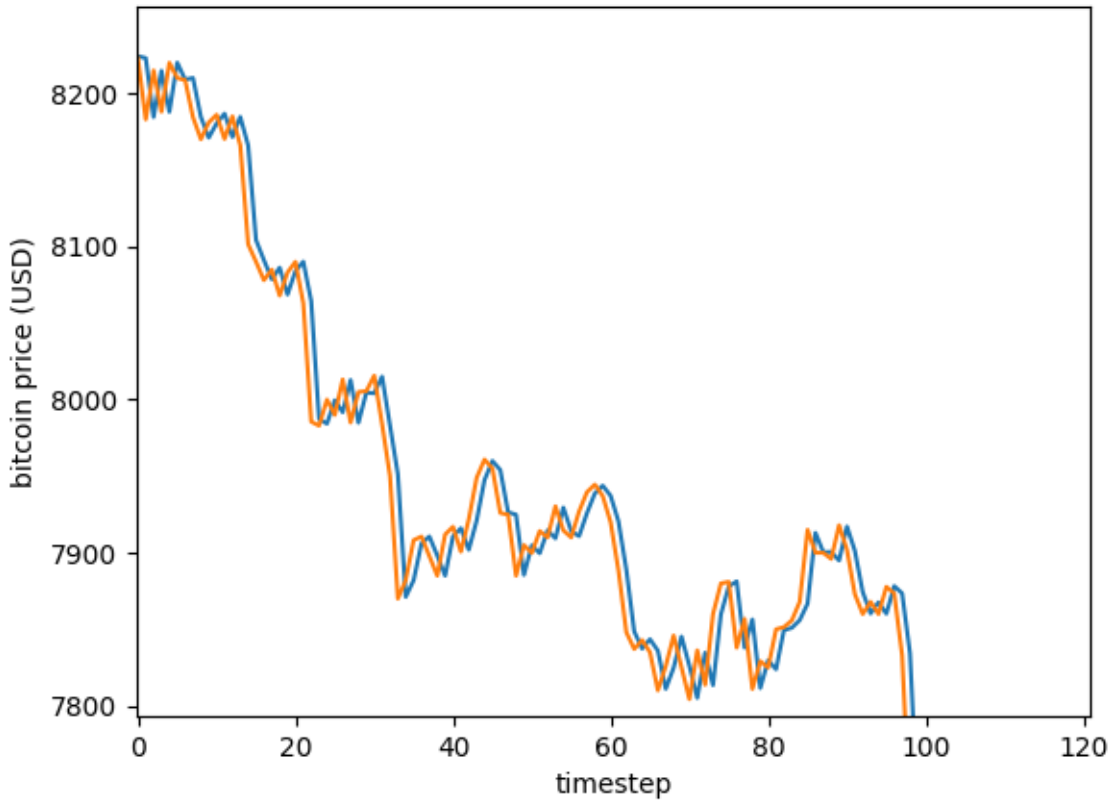
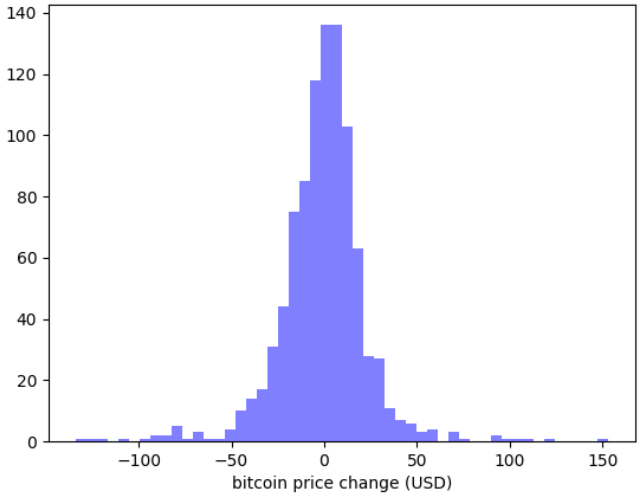
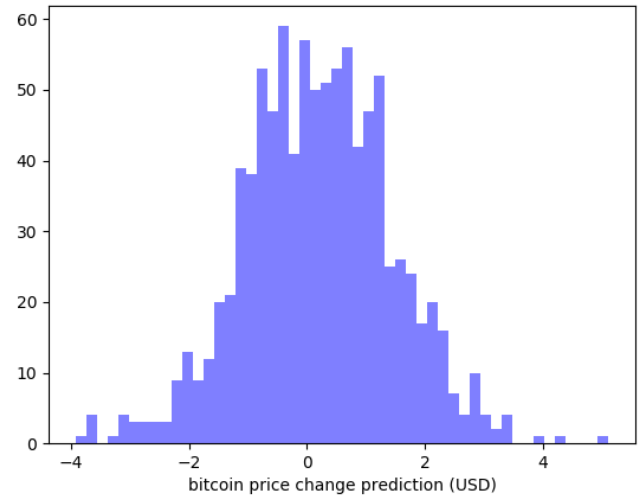


Figure 4.3: The Neural Network's predictions of the Bitcoin price in USD (United States Dollar) plotted over time in time-steps of 12.5 minutes (blue) and Bitcoin's actual price movement in USD in time steps of 12.5 minutes (orange). Note that this graph only shows about the first 100 time-steps to show the difference between the prediction line and the price movement line.



(a) Bitcoin's actual price distribution in a histogram.



(b) The Neural Network's predicted price change distribution.

Figure 4.4: Comparison between the actual price change distribution of Bitcoin and the Neural Network's predicted price change distribution. The price changes are recorded in time-steps of 12.5 minutes. Figure a shows the actual price distribution in a histogram and figure b shows the Neural Network's prediction distribution in a histogram. Note that the x-axis in both figures is different.

4.3 Two-class problem

Now we will examine the results of the Neural Network on the two-class classification problem. We did not need to use any time-step increase because there is already a balance between the two classes. First we observe the results using the Ethereum-Bitcoin data set and then we observe the results of the Bitcoin-USD data set.

Ethereum-Bitcoin data set

Measure	Result
Prediction accuracy	57.393%
Prediction class distribution (*Down*, *Up*)	$30.176\%, 69.824\%$
Test data class distribution (*Down*, *Up*)	$50.685\%, 49.315\%$
Return rate without Binance fees	$6.95937936774 * 10^{16}$
Number of trades	251504

The corresponding confusion matrix is given by:

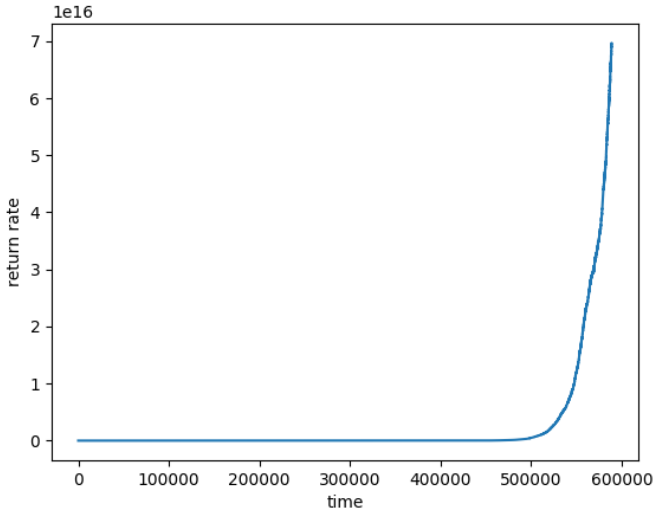
		Correct class	
		Down	*Up*
Predicted class	*Down*	18.954%	31.731%
	Up	31.731%	18.954%

Bitcoin-USD data set

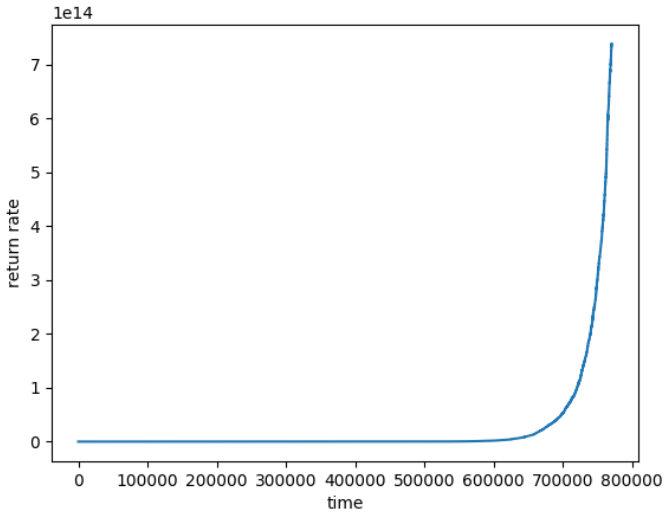
Measure	Result
Prediction accuracy	59.886%
Prediction class distribution (*Down*, *Up*)	$54.897\%, 45.103\%$
Test data class distribution (*Down*, *Up*)	$51.472\%, 48.528\%$
Return rate without Binance fees	$7.38226288632 * 10^{14}$
Number of trades	394094

The corresponding confusion matrix is given by:

		Correct class	
		Down	*Up*
Predicted class	*Down*	33.127%	21.769%
	Up	18.345%	26.759%



(a) Bitcoin-Ethereum data set



(b) Bitcoin-USD data set

Figure 4.5: Return rate over time for the two different data sets when not considering transaction costs

On the Ethereum-Bitcoin data set we achieve a prediction accuracy of 57.393%. On the Bitcoin-USD data set we achieve a prediction accuracy 59.886%. The return rates are exceptionally high, but if we consider the high number of trades in both cases and the prediction accuracy above 50% it is sensible. To show this we can calculate the average return rate per trade:

$$\sqrt[394094]{7.38226288632 * 10^{14}} = 1.0000109$$

In Figure 4.5 we can also see that the exceptionally high return rates are just a consequence of the exponential growth when the average return rate is just slightly above 1. Besides that, it would be impossible to actually achieve such a high return rate without influencing the market price by our own trading.

4.4 Comparing results

In this section we will compare our results and try to derive conclusions from these comparisons. According to our results in the three-class problem, Neural Networks outperformed ARIMA in both prediction accuracy and return rate without transaction fees. ARIMA did get a higher return rate with transaction fees but this is mostly due to the lower number of trades made by ARIMA. All return rates were negative.

When using the models to predict the exact future Bitcoin price, ARIMA and Neural Networks perform somewhat similar. Both show a strong bias towards no change at all. This is probably because when predicting the exact price, no change at all is the safest choice.

If we take a look at the results of the two-class prediction problem, we can clearly see that the Neural Network has some predictive value. The return rates are also exceptionally high but this due to exponential growth and a lot of trades.

We did not manage to predict the price of Bitcoin well enough to actually make a profit. All return rates when considering transaction costs are negative. Because of this, we would advise against using Neural Networks or ARIMA to trade Bitcoin. We do see that Neural Networks have some predictive value in the two-class problem, so maybe future research can improve these predictions such that we can actually get positive return rates while taking into account transaction costs.

Chapter 5

Related Work

Work related to the subject of predicting Bitcoin prices is mostly about using machine learning techniques on financial data. We will consider the most similar case to Bitcoin price prediction, which is stock price prediction. Afterwards we will examine the field of Bitcoin price prediction. Subsequently, we identify our own contribution to the field and future works.

5.1 Stock price prediction

In [16] the authors tried using machine learning techniques to predict the Indian stock market. They used Artificial Neural Networks, Support Vector Machines, Random Forest and naive-Bayes. By doing this they achieved a prediction accuracy of 75-83% with Random Forest having the highest accuracy. In [10] the authors used Support Vector Machines to predict the movement of the NIKKEI 225 index getting a prediction accuracy of 75%. In [21] Multiple Discriminant Analysis and Artificial Neural Networks were used to predict stock returns from companies mentioned in Fortune 500 and Business Week's "top 1000". They achieved a mean prediction accuracy of 65% using the Multiple Discriminant Analysis and 77.7%. In [5] they test whether Back Propagating Neural Networks or Support Vector Machines are more capable at predicting stock prices on six major Asian stock markets. They find that Support Vector Machines perform better most of the time, but not always. In [6] they introduced a Genetic Algorithm - Support Vector Machine hybrid model to predict the price of three stocks. They achieved a prediction accuracy of almost 62% using this method. In [19] they used a different approach. They collected news articles about S&P 500 stocks and the stock prices and used this data to predict the future stock prices. They achieved a prediction accuracy of 57.1%.

5.2 Bitcoin price prediction

In [1] the authors proposed several algorithms to predict the Bitcoin price, such as ARIMA, Random Forest, Logistic Regression and Linear Discriminant Analysis. They have achieved impressive results such as 60 to 70% prediction accuracy for all algorithms and return rates varying between 2.6 and 6.9. They also managed to reproduce these results using older training data and the same test data. In [14] the authors used more complex Neural Network structures such as Recurrent Neural Networks and Long Short Term Memory Networks. The Long Short Term Memory Network performed the best achieving a prediction accuracy of 52%. In [11] they used a Neural Network that optimizes return rates. In a 1.8 month period they achieved a return rate of 10.

5.3 Contribution

This thesis contributes to the relatively new field of Bitcoin price prediction in several ways. We used ARIMA and Artificial Neural Network models on real Bitcoin price data to predict the Bitcoin price. The Neural Network model did outperform the ARIMA model but we did not manage to get any positive return rates. This is because we took into account transaction costs when creating and evaluating our models. We are one of the first to take transaction costs in algorithmic Bitcoin trading into consideration. This has given us more realistic return rates and a better grasp of how well the applied Machine Learning techniques would actually work when trading Bitcoin. We would not recommend using Neural Networks for Bitcoin trading for now because return rates are negative.

5.4 Future work

As stated in the previous section, we did adjust for transaction fees in this research. However, to get even more realistic performance measures we should also consider other trading complications that Bitcoin has. For example, we did not consider transaction latency. In our trading simulation Bitcoin are bought and sold instantly but when you actually want to buy or sell Bitcoin there is a delay due to the transaction time itself and the time needed to find a buyer/seller. This delay is not very consistent. It is between 10 and 60 minutes most of the time but has peaks where it takes over 40 hours to complete a transaction [4].

Another complication that we did not look into is the impact we have on the Bitcoin market. If we only trade small amounts this impact will be negligible, but if we would trade large amounts of Bitcoin we might have an impact on the Bitcoin price. In our simulation we did not consider this.

In future work we should also develop models that try to look more than one time-step into the future. In this experiment the prediction models only predict one step into the future which result in us missing out on a profit. For example if the Bitcoin price increases gradually in the next three time-steps by 0.05% for each step. The price will increase by 0.15% but we will not invest because the predictions (if they are correct) will classify the future price as Stay for each step, even though there was a profit to be made because 0.15% is larger than 0.1%.

Finally, we should also consider using different Machine Learning techniques like the Support Vector Machine and Random Forest. Other research has shown that these can perform better than Artificial Neural Networks. We should also consider more advanced Artificial Neural Networks that such as Back Propagating Neural Networks to possibly achieve better predicting accuracy. In addition, we should also find the best inputs for our Neural Networks. We could take a look at the optimal inputs derived from financial data and we could also consider extra sources like news items and social media discussions about cryptocurrency.

Chapter 6

Conclusions

We tested an ARIMA model and several Artificial Neural Network models on our formalized Classification problems. We measured the performance of the models by their prediction accuracy and return rates.

Artificial Neural Networks outperformed ARIMA but did not perform well enough to make a profit. All of our return rates where negative when we considered transaction fees. However, this did not mean the Neural Network models had no predictive value at all. On the three-class problem a prediction accuracy of almost 40% was achieved and on the two-class problem we achieved a prediction accuracy of almost 60% by the Neural Network models. Also our return rates where exceptionally high when not considering transaction fees but this is mostly due to the nature of exponential growth and not considering our own impact on the market.

From the negative return rate results in the three-class problem we can conclude that we cannot make a profit by trading Bitcoin using the ARIMA or Neural Network models that were used in this thesis. This is why we would not recommend using these models to trade Bitcoin.

In the two-class prediction problem we see clearly that our Neural Network model has some predictive value because it is able to predict the sign of the future price with 60% accuracy. Perhaps future research can improve the prediction accuracy significantly, which could mean we can use these models to make a profit by trading Bitcoin.

Bibliography

[1] M.J. Amjad and D. Shah. Trading bitcoin and online time series prediction. *Proceedings of Machine Learning Research*, 55, 2016.

[2] D. Asteriou and S.G. Hall. *Applied Econometrics A Modern Aproach.* Palgrave Macmillan, 2006.

[3] Binance. `https://www.binance.com`. Accessed: 2018-05-24.

[4] Bitcoin stats. `https://blockchain.info/charts`. Accessed: 2018-05-17.

[5] W. Chen, J. Shih, and S. Wu. Comparison of support-vector machines and back propagation neural networks in forecasting the six major asian stock markets. *Int. J. Electronic Finance*, 1(1):49–67, January 2006.

[6] Rohit Choudhry and Kumkum Garg. A hybrid machine learning system for stock market forecasting. *Journal of International Technology and Information Management*, 2(3), 2008.

[7] Bitcoin price. `https://coinmarketcap.com/currencies/bitcoin/`. Accessed: 2018-05-17.

[8] S.S. Haykin. *Neural Networks and Learning Machines*, pages 1–46. Prentice Hall, 2009.

[9] J. Heaton. *Introduction to Neural Networks for Java, 2nd Edition.* Heaton Research, Inc., 2nd edition, 2008.

[10] W. Huang, Y. Nakamori, and S. Wang. Forecasting stock market movement direction with support vector machine. *Computers & Operations Research*, 32(10):2513 – 2522, 2005.

[11] Z. Jiang and J. Liang. Cryptocurrency portfolio management with deep reinforcement learning. In *2017 Intelligent Systems Conference (IntelliSys)*, pages 905–913, Sept 2017.

[12] Keras api. `https://keras.io/`. Accessed: 2018-05-24.

[13] J. Lansky. Possible state approaches to cryptocurrencies. *Journal of Systems Integration*, 9(1), 2018.

[14] Sean McNally, Jason Roche, and Simon Caton. Predicting the price of bitcoin using machine learning. *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pages 339–343, 2018.

[15] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012.

[16] J. Patel, S. Shah, P. Thakkar, and K. Kotecha. Predicting stock and stock price index movement using trend determenistic data preperation and machine learning techniques. *Expert Systems with Applications*, 42, 2014.

[17] Atul Prakash. Rocky markets test the rise of amateur 'algo' traders. *Reuters*, 2016.

[18] K Sagona-Stophel. Bitcoin 101 white paper. Technical report, Thomson Reuters, 2016.

[19] Robert P. Schumaker and Hsinchun Chen. Textual analysis of stock market prediction using breaking financial news: The azfin text system. *ACM Trans. Inf. Syst.*, 27(2):12:1–12:19, March 2009.

[20] Tensorflow api. `https://www.tensorflow.org/`. Accessed: 2018-05-24.

[21] Y. Yoon and G. Swales. Predicting stock price performance: a neural network approach. In *Proceedings of the Twenty-Fourth Annual Hawaii International Conference on System Sciences*, volume iv, pages 156–162 vol.4, Jan 1991.