Zurich University
of Applied Sciences

School of
Engineering

IDP Institute of Data Analysis
and Process Design

Bachelor thesis/Project work (Study programme, max. 2 lines)

Title (max. 4 lines)

Author	Ken Geeler Pascal Simon Bühler Philipp Rieser
Main supervisor	Marc Wildi
Industrial partner	Mobiliar
Date	11.06.2021

Please fill in the title sheet taking into account the following points:

- ➔ Please do not change the font type or font size. Text should only be written over!
- ➔ Please use only 4 lines max. per table row!
- Template: did you choose the right institute/centre? → Logo institute/centre
- Title: add your study programme directly after the word ‚Bachelor thesis / Project work‘ (max. 2 lines).
- Title: overwrite the running text with your Bachelor thesis title / Project work title (max. 4 lines).
- Author: fill in your first and family name (list alphabetical > family name).
- Supervisor: fill in your supervisor/s (list alphabetical > family name).
- Sup supervisor: if you do not have a sup supervisor → please delete this table row.
- Industrial partner: if you do not have an industrial partner → please delete this table row.
- External supervisor: if you do not have an external supervisor → please delete this table row.
- Date: please fill in current date.
- Finish: at the end please delete this description (grey) and safe the document in pdf format.

Zurich University
of Applied Sciences

School of
Engineering

DECLARATION OF ORIGINALITY
Bachelor's Thesis at the School of Engineering

By submitting this Bachelor's thesis, the undersigned student confirms that this thesis is his/her own work and was written without the help of a third party. (Group works: the performance of the other group members are not considered as third party).

The student declares that all sources in the text (including Internet pages) and appendices have been correctly disclosed. This means that there has been no plagiarism, i.e. no sections of the Bachelor thesis have been partially or wholly taken from other texts and represented as the student's own work or included without being correctly referenced.

Any misconduct will be dealt with according to paragraphs 39 and 40 of the General Academic Regulations for Bachelor's and Master's Degree courses at the Zurich University of Applied Sciences (Rahmenprüfungsordnung ZHAW (RPO)) and subject to the provisions for disciplinary action stipulated in the University regulations.

City, Date:	**Name Student:**
Winterthur, 11.06.2021	Geeler Ken
Winterthur, 11.06.2021	Pascal Simon Bühler
Winterthur, 11.06.2021	Philipp Rieser

Contents

Abstract

1. Introduction

Ken is testing working with Github.does it work now?

1.1. Intro

2. Theory

The following chapter is intended to provide the theoretical foundations necessary for our work. It is divided into a part that provides an overview of artificial neural networks. Followed by section 2.2. which shows the background and the ecosystem of Bitcoin. This knowledge should be kept in mind, which should help in understanding the price formation of Bitcoin.

2.1. Neural network

In the context of this work, artificial neural networks are used to answer supervised learning questions that focus on the classification of data. This means that a neural network finds a correlation between the data and their labels and optimizes its parameters to minimize the error for the next try. This process is called supervised training and is performed with a test data sample. An application example of classification is that a neural network is used for face recognition after it has learned the classification of different faces in the process of supervised training. Predictive analysis works similarly to the classification of labeled data. It estimates future values based on past events and can be trained with historical data. On the other hand, unsupervised learning (clustering) is applied to detect patterns from unlabeled data. Based on these patterns, for example, anomalies can be detected that are relevant in the fight against fraud (fraud detection). Unsupervised learning is not discussed further in this paper. Section 2.1.1. will demonstrate the functioning of a neural network using a simple perceptron.

2.1.1. Perceptron

The construction of an artificial neural network is demonstrated using a perceptron. It is a simple algorithm for supervised learning of binary classification problems. This algorithm classifies patterns by performing a linear separation. Although this discovery was anticipated with great expectations in 1958, it became increasingly apparent that these binary classifiers are only applicable to linearly separable data inputs. This was only later addressed by the discovery of multiple layer perceptrons (MLP) [1]. Basically, a perceptron is a single-layer neural network and consists of the following five components and can also be observed in figure 1.

1. Inputs
2. Weights
3. Bias
4. Weighted sum
5. Activation function

Inputs are the information that is fed into the model. In the case of econometric time series, it is mostly the current and historical log returns (lags). These are multiplied by the weights and added together with the bias term to form the weighted sum. This weighted sum is finally passed on to the non-linear activation function, which determines the output of the perceptron.

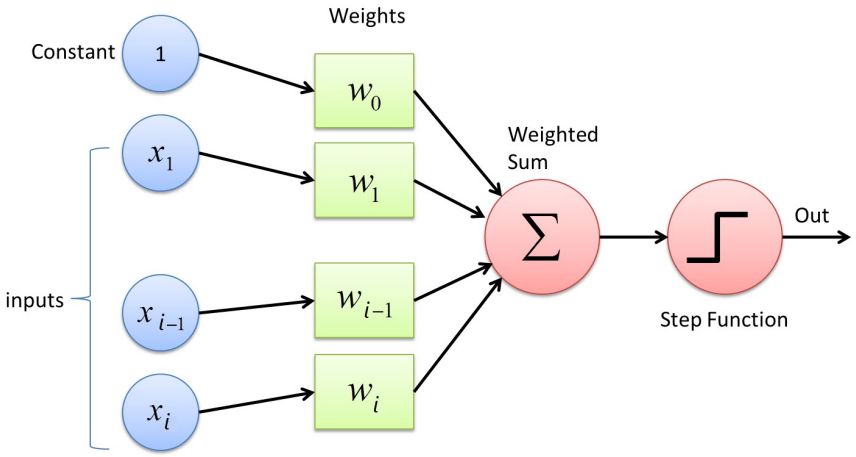

Figure 1: Schematic diagram of a perceptron.

The perceptron can also be represented as a function, which can be seen in equation 1. Analogous to the representation above, the inputs x_i are multiplied by the weights w_i in a linear combination. Then an error term is added so that the whole can be packed into the non-linear activation function $g(S)$. $\hat{y}$ is the binary output of this perceptron. With the aid of an activation function, binary output is obtained. The Heaviside step function shown in figure 1 is usually only used in single layer perceptrons, which recognize linear separable patterns. For the multi-layer neural networks presented later, step functions are not an option, because in the course of the backpropagation algorithm the gradient descent has to be minimized. This requires derivatives of the activation function, which in the case of this Heaviside step function equals 0. Because the foundation for the optimization process is missing, functions like the sigmoid function or the hyperbolic tangent function are used [2]. More about this topic is discussed in chapter 2.1.2.

$$\hat{y} = g(w_0 + \sum_{i=1}^{n} x_i w_i) \tag{1}$$

As just mentioned, the aim is to feed the perceptron with the training set and change the weights w_i with each cycle so that the prediction becomes more accurate. The output value is compared to the desired value. Finally, the sign of the difference $y - \hat{y}$ determines whether the inputs of that iteration are added to or subtracted from the weights. Ideally, the weights will gradually converge and provide us with a usable model [2].

2.1.2. Backpropagation algorithm

Finding the optimal weights of the neural network is achieved by finding the minimum of an error function. One of the most common methods for this is the backpropagation algorithm. This algorithm searches for the minimum of the error function by making use of a method called gradient descent. The gradient method is used in numerics to solve general optimization problems. In doing so, we progress (using the example of a minimization problem) from a starting point along a descent direction until no further numerical improvement is achieved. Since this method requires the computation of the gradient of the error function after each step, continuity and differentiability of this function must necessarily be given. The step function mentioned above in section 2.1.1. is therefore out of the question, but a non-linear function such as the logistic and the hyperbolic tangent functions (sigmoid) [3]. Both activation functions are visible in figure 2. While the target range of the 'ordinary' sigmoid function (equation 2) is between 0 and 1, the $\hat{y}$ of the hyperbolic tangent function (equation 3) ranges between -1 and 1. v_i equals the weighted sum including bias term.

$$\hat{y}(v_i) = (1 + e^{-v_i})^{-1} \tag{2}$$

$$\hat{y}(v_i) = \tanh(v_i) \tag{3}$$

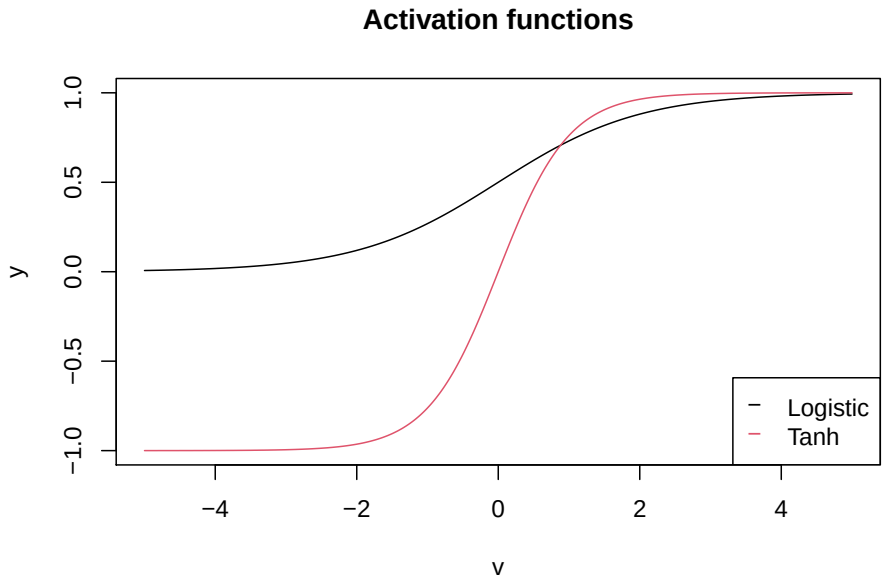

Figure 2: Two common sigmoid activation functions: logistic functions and hyperbolic tangent.

In the course of the error analysis, the output of the neural network respectively the result from the activation function in the output layer is compared with the desired value. The most commonly used error function E is the Mean Squared Error (MSE), which is seen in equation 4. y_i represents the actual value for the data point i, while $\hat{y}_i$ is the predicted value for data point i. The average of this error function is the average MSE, which is determined for a corresponding model. The learning problem is to adjust the weights w_i within the training sample so that $MSE(w)$ is minimized [4].

$$\begin{aligned} E &= MSE(w) \\ &= \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}(y_i - g(w_0 + x_i w_i))^2 \end{aligned} \tag{4}$$

As mentioned, this is searched for by the gradient descent method. The gradient of a function is a vector whose entries are the first partial derivatives of the function. The first entry is the partial derivative after the first variable, the second entry is the partial derivative after the second variable and so on. Each entry indicates the slope of the function in the direction of the variable to which it was derived. In this work, the notation ∇E is used when talking about the gradient for the error function E, which is displayed in equation 5 [3].

$$\nabla E = (\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots, \frac{\partial E}{\partial w_i}) \tag{5}$$

The weights get adjusted according to the following algorithm 6 where Δw_i is the change of the weight w_i and γ represents a freely definable parameter. In literature, this parameter is often called a learning constant [5]. The negative value is used because the gradient naturally points in the direction with the largest increase of the error function. To minimize the MSE, the elements in the gradient ∇E must be multiplied by -1.

$$\begin{aligned} \Delta w_i &= -\gamma \frac{\partial E}{\partial w_i}, \\ \text{for } i &= 1, 2, \ldots, n \end{aligned} \tag{6}$$

2.1.3. Multilayer perceptron

Multilayer perceptrons (MLP) are widely used feedforward neural network models and make usage of the backpropagation algorithm. They are an evolution of the original perceptron proposed by Rosenblatt in 1958 [1]. The distinction is that they have at least one hidden layer between input and output layer, which means that an MLP has more neurons whose weights must be optimized. Consequently, this requires more computing power, but more complex classification problems can be handled [6]. Figure 3 shows the structure of an MLP with n hidden layers. Compared to the perceptron, it can be seen that this neural network consists of an input layer, one or more hidden layers, and an output layer. In each layer, there is a different number of neurons, respectively nodes. These properties (number of layers and nodes) can be summarized with the term 'network architecture' and will be dealt with in this thesis.

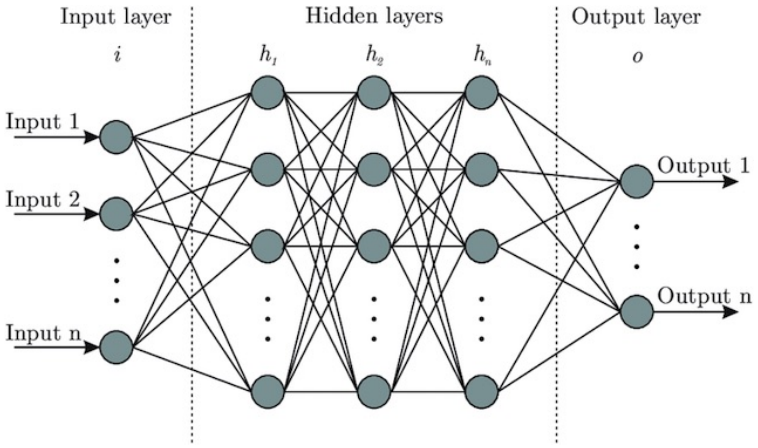

Figure 3: Schematic diagram of a multilayer perceptron

Every neural network has an input layer, which consists of one or more nodes. This number is determined from the training data and tells us how many features should be delivered to the neural network. In the case of bitcoin prices, we could use today's price and the prices of the last 10 days (lags 1-10), so the input layer would consist of 11 nodes. Some configurations also require a bias term to adjust the output along with the weighted sum, which is also added to the input layer. In contrast to the scheme of the MLP, this setup can be seen in figure 1 where the bias term is defined as 'constant.' Similarly to the input layer, each neural network has exactly one output layer. This can consist of one or more nodes. In this thesis, MLP is used as a regressor and therefore only one neuron is needed in this layer.

In between are the hidden layers, whose number and size can be configured as desired. The challenge is to find an optimal and efficient configuration without causing overfitting of the training data. The number of hidden layers depends primarily on the application area of the neural network. For example, working with image recognition would require more layers since the image file is broken down into individual pixels. Subsequently, the layers are used to optimize from rough outlines to the smallest detail. In our research, we came across several methods or 'rules of thumb' to optimize the model. A frequently suggested method is explained by Andrej Karpathy (director of the AI department of Tesla, Inc.). His GitHub entry recommends the approach of starting with a model that is too large that causes overfitting. Subsequently, the model is reduced by focusing on increasing training loss and improving validation loss [7].

2.1.4. Recurrent neural networks (RNN)

Recurrent neural networks (RNN) are a further development of conventional neural networks. While MLP use new inputs x_i in each epoch, RNN also use sequential data h_i in addition to x_i. This sequential data are called hidden states and result from the previous runs. This has the advantage that historical information stemming from past predictions is included for the prediction for $t+1$. This effect can be intuitively explained by an example in which the flight path of a scheduled flight is predicted using RNN. When predicting the exact location (coordinates) of a plane, it is of great advantage to know the location at $t-1$ and to derive the

flight direction from it. With the inclusion of this information, the target area can be narrowed down, which optimally leads to more accurate results. The same principle is used in applications like machine translation and speech recognition, where the result (here possibly letter or word) of the last epoch plays a big role for the next prediction [8].

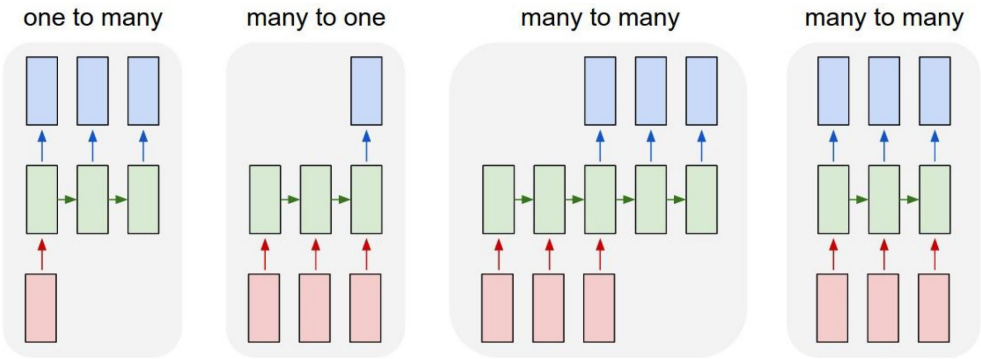

Figure 4: Process sequences of different applicances of RNN.

Figure 4 shows different process sequences of the RNN, which vary depending on the field of application. The red rectangles at the bottom represent the number of inputs. Similarly, the blue rectangles represent the outputs that come out of the RNN. The term 'many' refers to > 1 and is illustrated with three rectangles in the figure. The green ones represent the hidden states h_i of all time steps and thus can be seen as the memory of the neural network. The green arrows show that the previous hidden state is used as input for the current step. Starting from the left: one-to-many can be used for image captioning (extracting sequence of words from images), many-to-one for sentiment classification from sequence of words, many-to-many for machine translation (sequence of words in one language to sequence of words in another language) and many-to-many for video classification on frame level [9]. For the prediction of the BTC/USD exchange rate in this paper, we deal with the process many-to-one. This method combines information from inputs and hidden states into one single prediction value.

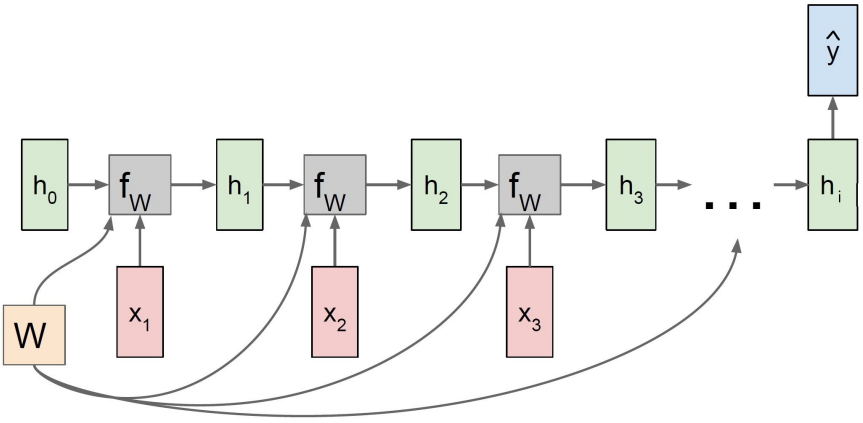

Figure 5: Computational graph of a many-to-one RNN.

$$\begin{aligned} h_i &= f_W(h_{i-1}, x_i) \\ &= \tanh(W_h h_{i-1} + W_x x_i + b) \end{aligned} \tag{7}$$

Equation 7 shows how the hidden states h_i are calculated at each time step, i where f_W is an activation function (here: hyperbolic tangent function), h_{i-1} is the previous state and x_i is the input vector at time step i.

In some cases, a bias term b is added to the parameters. W_h represents the weight matrix for h_i with dimension (length(h)×length(h)). Thus, W_x is the weight matrix for x_i with dimension (length(h)×length(x)).

$$\hat{y}_i = W_y h_i \tag{8}$$

Looking at equation 8, y_i equals the output and desired prediction of the RNN. The prediction results from the matrix-vector product of the weight matrix W_y with dimension (length(h)×length(y)) and the hidden states vector h.

2.1.5. Long-short term memory (LSTM)

2.1.6. Challenges

2.1.6.1 Overfitting

We have encountered several challenges that can occur when using neural networks. One of these possible problems is called overfitting. The goal of a neural network is to build a statistical model of the training set that is capable of generating the data. In overfitting on the other hand, the exact conditions of the training data including noise are reproduced. The focus is no longer on the underlying function. Last but not least, an unnecessarily large number of parameters or epochs can be 'consumed' for this, which makes the whole process relatively inefficient [8].

=> still needs clarification how we solve these challenges in this thesis!

2.1.6.2. Vanishing gradient problem

Another characteristic that requires our attention is the vanishing gradient problem. As explained in chapter 2.1.2., the weights of the neural network are adjusted using the gradient of the loss function. Thereby, the problem can occur that the gradient almost vanishes. The error function's gradients become so small that the backpropagation algorithm takes smaller steps towards the loss function's minima and eventually stops learning. For example, if the derivative of an activation function such as the logistic sigmoid function approaches zero for extremely large or small values for x. To avoid these extreme values for x, the inputs are scaled and normalized in this paper. This ensures that the definition range is within the range where the gradient is still large enough for the backpropagation algorithm.

=> still needs clarification how we solve these challenges in this thesis!

2.2. Model comparison

This thesis sets the goal to compare the different neural networks presented. Besides the types of neural networks, the network architecture (number of layers and nodes) is explored. In addition, a comparison is made with the winner of the Forecasting Competition M4, which combines a standard exponential smoothing model with an LSTM network. To make the comparison meaningful enough, the following two figures are compared.

2.2.1. Sharpe ratio

The first number refers to the performance of the trading strategy based on the sign of the prediction $t+1$ and is called Sharpe Ratio. Sharpe ratio is a very powerful and widely used ratio to measure performance and it describes return per risk.

$$SharpeRatio = \frac{R_p - R_f}{\sigma} \tag{9}$$

R_p represents the return of the portfolio, while R_f equals the risk free rate. σ is the standard deviation of the portfolios excess return (risk). For the comparison of different series, the Sharpe Ratio needs to be annualized with $\sqrt{365}$ as the crypto market is open 24/7.

2.2.2. Diebold Mariano

The second method used is the Diebold Mariano test, which compares the predictive accuracy between two forecasts. First, the loss differential d_i between two forecasts is defined in equation 11 where a loss function L of one model is subtracted from another model. The proposed loss functions include absolute errors (AE) and squared errors (SE) [10]. Given an expected value of $d = 0$, both forecasts are assumed to have the same accuracy. If the expected value differs from zero, the null hypothesis can be rejected. This would mean that the two methods have different levels of accuracy.

$$\begin{aligned} H_0 &: E(d_i) = 0 \\ H_1 &: E(d_i) \neq 0 \end{aligned} \tag{10}$$

with

$$d_i = L(e_{1i}) - L(e_{2i}) \tag{11}$$

and

$$\begin{aligned} e_{ti} &= \hat{y_{ti}} - y_i \\ &\text{for } t = 1, 2 \end{aligned} \tag{12}$$

Under the null hypothesis H_0, the Diebold Mariano test uses the statistics shown in equation 13 and is asymptotically N(0,1) distributed. On the other hand, the null hypothesis is rejected if the calculated absolute Diebold Mariano value is outside $-z_{\alpha/2}$ and $z_{\alpha/2}$. Thus, $|DM| > z_{\alpha/2}$ is valid when there is a significant difference between the predictions where $z_{\alpha/2}$ is the positive bound of the z-value to the level α.

$$DM = \frac{\bar{d}}{\sqrt{\frac{2*\pi*\hat{f_d(0)}}{T}}} \rightarrow N(0,1) \tag{13}$$

where $\bar{d}$ is the sample mean of the loss differential and $f_d(0)$ is the spectral density of the loss differential at lag k [11].

$$\bar{d} = \sum_{i=1}^{T} d_i \tag{14}$$

$$f_d(0) = \frac{1}{2\pi}\left(\sum_{k=-\infty}^{\infty} \gamma_d(k)\right) \tag{15}$$

In conclusion, the Diebold Mariano test helps us to understand whether the predictions of one model turned out better by chance or due to statistical significance.

2.2.3. Mean Squared Error

The third performance measurement method is also widely used and called Mean Squared Error. Its calculation is very simple, for every timestep the estimated value is subtracted from the real empirical value, squared and then summarized and divided by the absolute number of observations as seen in equation 16.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(realvalue_i - \hat{prediciton_i})^2 \tag{16}$$

Altough its wide application in many different sectors, the MSE has a problem with outliers, due to the squared term, a huge outlier could influence the MSE very strongly. Therefore one should always check for outliers.

2.3. Bitcoin

In this section bitcoin as a crypto-curreny is introduced. The historical data is analyzed and commented. Further the technology in and around crypto-currencies is briefly explained. A detailed explanation would require a paper itself, therefore the explanation is done as simple as possible.

In the following work bitcoin as a cryptocurreny is mentionend in its short term BTC, by the meaning of US Dollars per Bitcoin.

2.3.1. Historical analysis

The story of bitcoin began with a paper published by the name of Satoshi Nakamoto [12]. The publisher of the document cannot be assigned to a real person, therefore the technology inventor remains mysteriosly unknown until today. In 2009 the first bitcoin transaction was executed. On account of the opensource technology of bitcoin, lots of alternative currencies were created.

Until 2013 the cryptocurrencies operated under the radar of most regulatory institutions. Because of the anonymity of the transactions, criminals were attracted by the newborn payment method. Headlines, such as the seizure of 26,000 bitcoins by closing the "Dark-Web" Website Silkroad through the Drug Enforcement Agency, followed moreoften in the newspapers.

Nevertheless in 2014 more companys, such as: Zynga, D LasVegas Casinos, Golden Gate Hotel & Casino, TigerDirect, Overstock.com, Newegg, Dell, and even Microsoft [13], began to accept bitcoin as a payment method. In 2014 the first derivative with bitcoin as an underlying was approved by the U.S.Commodity Futures Trading Commission. 2015 an estimated 160000 merchants used bitcoin to trade.

Let us first look a the price in Figure 7 and the log(price) in Figure 6 and get a sense of the chart. Note: The data in the charts start in 2014 where it was listed in coinmarket, events between 2009 and 2014 are described without visualization.

Around 2010 bitcoin had the first increase in price as it jumped a 100% from 0.0008 USD to 0.08 Dollar [14]. In 2011 the price rose from 1 USD to 32 USD within 3 months and recessd shortly after to 2 USD this can be referred as a first price bubble in bitcoin, for the next year the price climbed to 13 Dollars and reached a never seen level of 220 USD, only to plunge to 70 USD within a half month in April 2013. By the end of the year a rally brought btc up to a peak of 1156 USD. The following year brought bad news and the price slowly decreased to 315 USD in 2015 after an observed drop of 20% after news from the trial of Ross Ulbricht, founder of Silk road marked in Letter **A**.

From this point in time, things began to change, more volume was flushed in the market and the price of BTC began to ascend and the real rally began ,the BTC rose up to 20k USD / BTC on 17th September 2017 **B**. After the rise comes the fall and BTC lost value for more than a year until **C** 2018-12-15 the trend reverted and found its peak after 6 months in **D** 2019-06-26, but oncemore it was not lasting for long as bitcoin lost **D** 2020-03-12 nearly half its value in 4 days.

But the story wasn't over by now, after the drop the price of the cryptocurrency regained value, passed previous levels and shortly after exploded, after companies like tesla and signal bought a big chunk of bitcoins, into a maximum of 58000 USD per bitcoin. It is also observed that the value of bitcoin is very volatile, we will discuss this in section 3.2..

Figure 6: Logarithmic Bitcoin price USD

Figure 7: Bitcoin Price USD

2.3.2. Bitcoin Technology cryptocurrencies

In this section bitcoin as a cryptocurreny based on the blockchain technology is briefly described. The term cryptocurrency as a more general term is used because bitcoin was only the first of its kind.

One may look at a cryptocurrency similar to a normal currency because you can buy and sell things and get bitcoin in exchange. But cryptocurrencies fundamentally differ to conservative currencies in merely all ways. The cryptocurrencies (not just bitcoin) are based on the blockchain technology introduced in Nakamotos paper[12].

The system is decentralized, where no institution or Government regulates the market in terms of the blockchain itself. The transactions are signed by the participants via cryptographic hash functions which generates a private and public key. This means that every signature can only be accessed by the owner of the private key. Once a transaction is signed, it is broadcasted into the network to all participants, so that everyone sees a transaction has been made. Around 2400 transactions are packed in a block (the blocksize is limited by memory) which are broadcasted to all participants of the system. Ever block consist the transaction information, previous hash ,the special number and their resulting hash as visualized in Figure 8. Miners are now trying to approve the block by generating a hash with a certain pattern with the hashfunction f(prevhash, data of the block, special number), the so called proof of work. The first Miner who finds the according special number to the hash with the certain pattern, gets an amount of bitcoin in reward. The block with its new hash and the special number are now added to the chain and are broadcasted to the network. If someone manipulates transactions in a block and finds the special number to the hash he could potentially get away with it but no for long because for the next block he must be also the first to find the right hash and so on ,in Figure 8 the red block is a false one. Only the longest chain is to be trusted, and because there are so many miners one must have more than 50 % off the calculating power to get the best chance to find the right hash. Therefore its almost impossible to manipulate the chain. The cryptocurrency itself is now entirely defined by a chain of approved blocks by all participants.

Another interesting fact about bitcoin is that the amount of coins is determined by the rewards of the miners. The first block (genesisblock) had a reward of 50 bitcoins, every 4 years this reward gets halved. Therefore the maximum amount of bitcoins is 21 million.

In recent days the cryptocurrency has come under a lot of critisicm because of its impact to the natural environment. The immense computing expenditure has a very high power consumption.

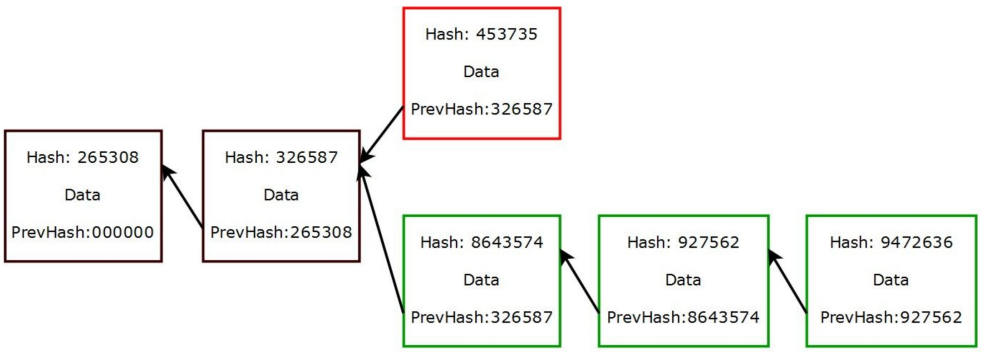

Figure 8: Blockchain schema

3. Methodology

The focus of this thesis can be divided into two areas. First, the aim is to find an optimal neural network including a network architecture. This should perform well in the application area, in which the future log return of the Bitcoin is predicted on the basis of historical log returns. In a second step, we will focus on defining a trading strategy based on our findings. All considerations and findings will be presented in a quantitative way and compared with each other. Figure 9 helps to get an overview of the individual steps followed in this chapter.

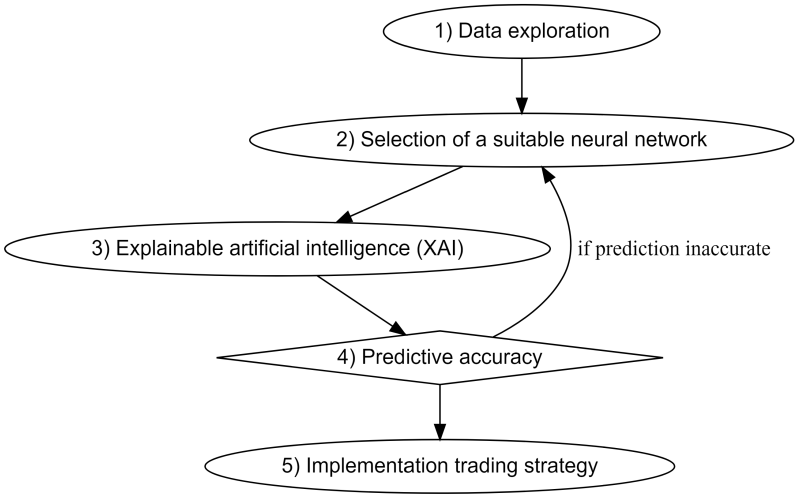

Figure 9: This flowchart illustrates an overview of the individual intermediate steps that are covered in the Methodology chapter.

3.1. Data exploration

The data in this paper is accessed via yahoofinance provided by coinmarket https://coinmarketcap.com/. We use the daily "closing price" of bitcoin in US Dollars with the ticker BTC-USD. Cryptoassets are tradeble 24 hours a day 365 days a year. There is no real "closing price" for the bitcoin, hence the "closing-Price" is just the last price of the day evaluated at the last timestamp with timeformat UTC.

In chapter 2.3. the bitcoin price and the logarithmic price is visualized. For processing and analyzing the data in order to fullfill the weak stationarity assumptions we transform the data into logreturns according to equation 17.

$$\text{LogReturn} = \log(x_t) - \log(x_{t-1}) \tag{17}$$

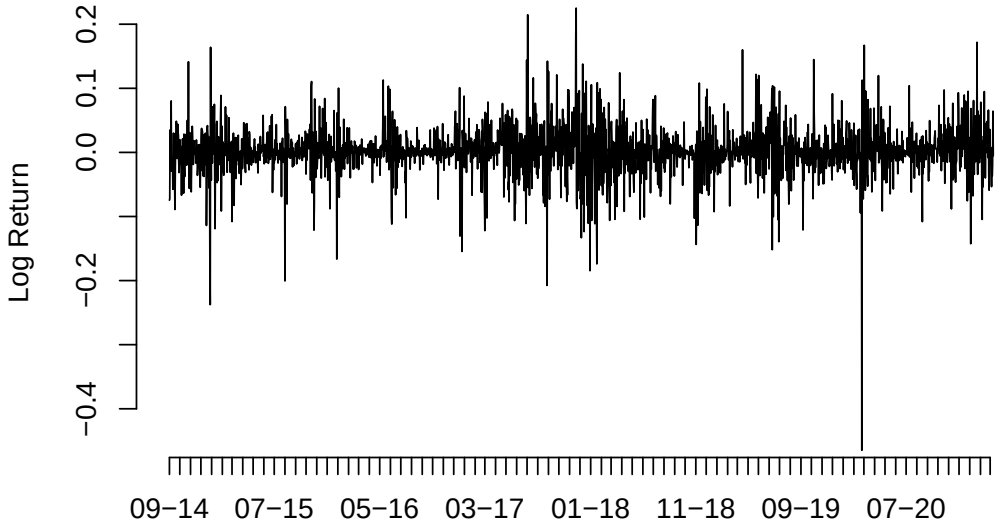

Figure 10: logreturns BTC

In plot 10 various large outliers are visible, especially one outlier towards the end catches the eye. The series seems to be very noisy, nevertheless volatility clusters are observable. By computing the ACF of the series in figure 11, we can describe the dependency in these clusters. According to the ACF the lags 6 and 10 are significant on a 5% level.

Curios from which distribution the logreturns might originate, we are fitting a Normaldistribution and a Students-t distribution to the data in Figure 12.

Interestingly the mean is shifted slightly (0.002) to the positive side.By inspecting the tails, one can observe that the negativ tail is not fitted as good as the positive part by the t distribution. The two Normaldistributions either over- or underestimate the values in the tails, therefore we conclude that the proposed t-distribution fits the data better but also not perfect.

Pointing at the extreme outlier discussed earlier, visible in Figure 10 towards the end, the density plot makes clear how unimaginable small the probability of this extreme observation is.

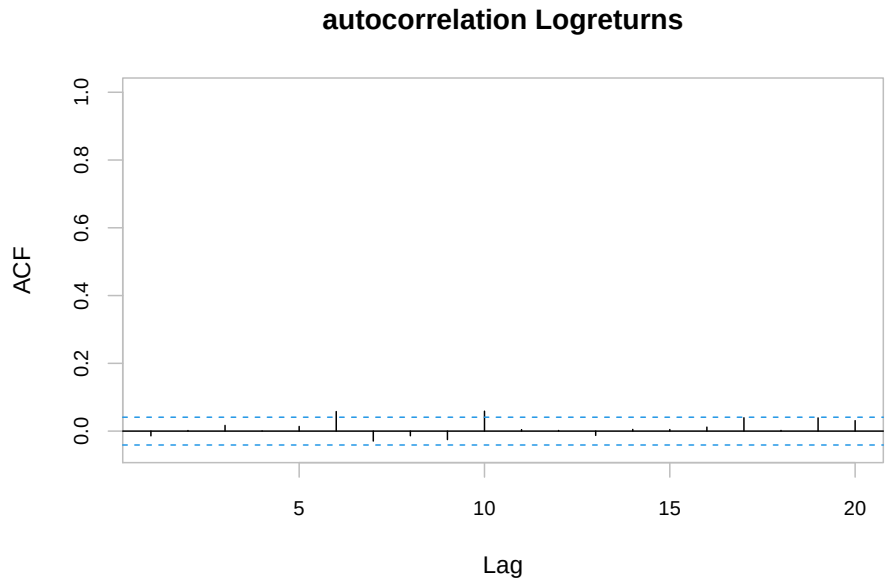

Figure 11: logreturns BTC

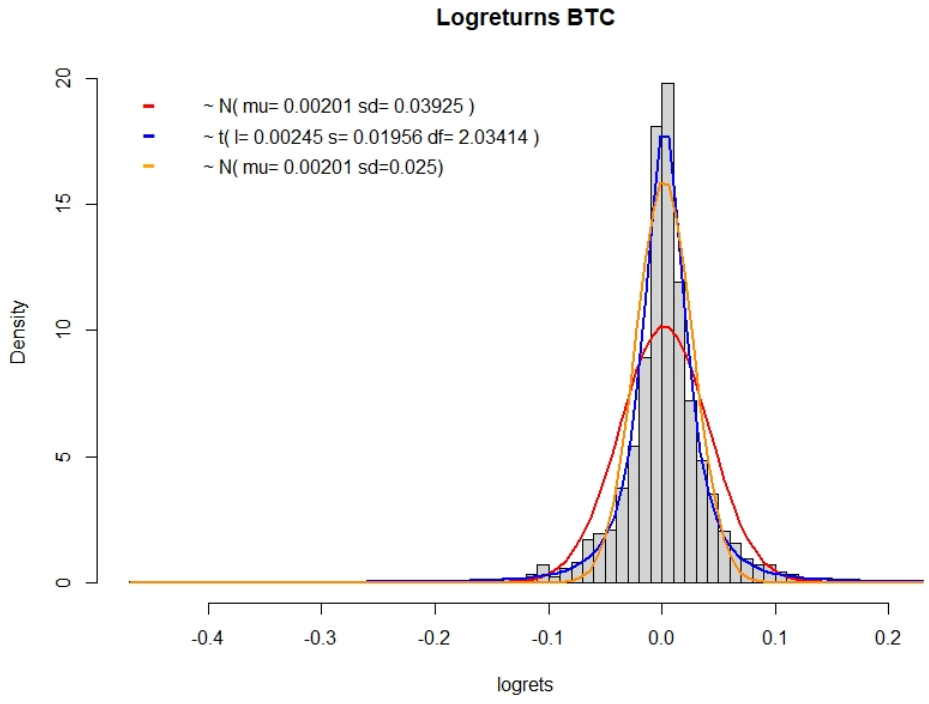

Figure 12: Distribution of Logreturns

3.2. Network architecture

As mentioned in chapter 2.1.3., choosing an appropriate network architecture for bitcoin price prediction is a crucial step in order to achieve useful forecasts while avoiding overfitting. Due to the complexity as well as the non-linearity of neural networks, the interpretation cannot be performed intuitively. For this reason, an approach is pursued in which neural networks with different numbers of layers and neurons are compared with each other by using the MSE loss. This allows us to compare accuracy and possibly see a connection with network architecture.

To find the optimal network architecture, we test a maximum of 3 layers with a maximum of 10 neurons each. We limited ourselves with this number, because on the one hand in the extent of this bachelor thesis the time lacks, and on the other hand we think that with time series, too complex models on no added value and rather to overfitting lead. The simplest network has one layer with one neuron (1), while the most complex has 3 layers with 10 neurons each (10,10,10). The total number of different combinations can be expressed as follows:

$$\text{comb} = \sum_{i=1}^{L} N^i \tag{18}$$

with:

L = maximum Layer $\in \mathbb{N}^*$

N = maximum Neuron $\in \mathbb{N}^*$

comb = Number of all combinations

Thus, with our initial setup, we obtain a maximum neuron-layer combination of 1110. To respond to the challenges mentioned in section 2.1.6., not only a single network per neuron-layer combination is trained, but a whole batch of 50 networks. So you end up with a total of 55500 trained networks. For each individual network, the in-sample and out-of-sample MSE as well as the Sharpe ratio are determined. We use these values to find an optimal network architecture based on the statistical error as well as on the trading performance (daily trading).

This now leads us to our next consideration. For which period and for which in-sample and out-of-sample split should the architecture be defined.

3.2.1. Defining train and test samples

We are looking for an optimal network, the optimal network should also provide reasonable and reliable predictions for different periods. So, for further analysis, we are going to use a subset of the introduced closing prices of bitcoin. Starting from the first of January 2020 to the 27th of March in 2021, we only consider 15 months for our data.

The reason for doing so is, we don't believe that the historical data longer than a year is consisting of any information about the price tomorrow. By optimizing our models we found that more data would bring no additional performance, therefore the selected subset should be sufficient. As proposed in [15] we choose a test train split from 6 months in-sample and 1 month out-of-sample. This split is applied to the whole subset in form of a rolling window. By stepping forward with this 6/1 split by steplength of one month we end up with 9 data splits in total. In figure 13 this procedure is visualized, for every new timestep a new month is considered for the out of sample and the first month of the in sample falls out of the frame.

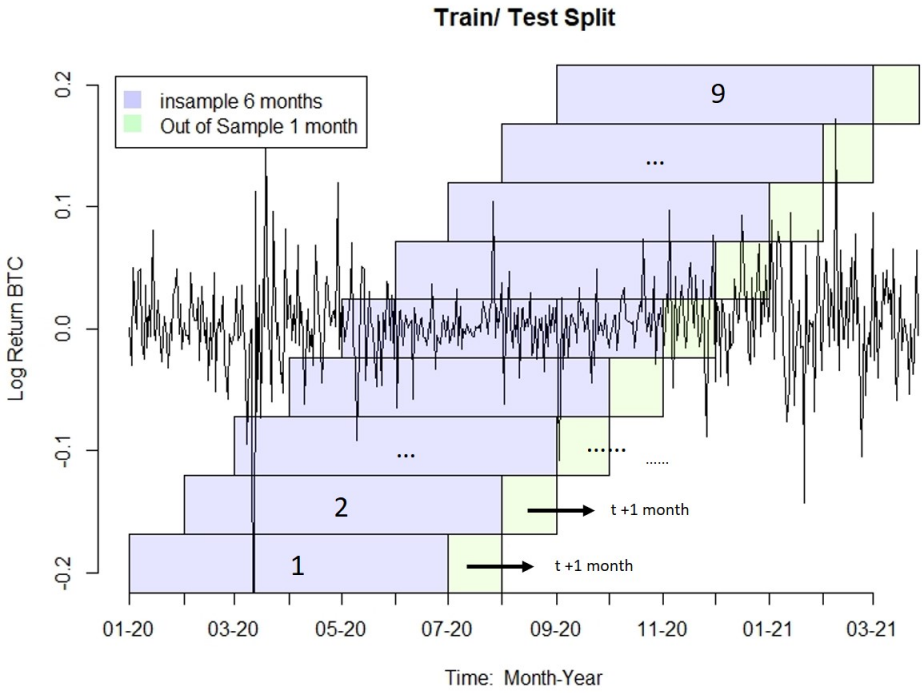

Figure 13: Distribution of Logreturns

In the course of the time series in figure 13, one can see different periods. Strongly volatile as well as rather calm phases occur. With the rolling window, we can train and test the networks based on different phases. Thus, we can also evaluate the performance of the networks based on different phases and not only on a predefined single test and train split.

The complexity of the search for the optimal network architecture increases significantly here. With the conditions defined for us, we train and test a total number of 499500 networks to define the optimal network.

3.2.2. Evaluating network architecture

Here we would like to focus on some findings that we discovered during the processing of the trained networks. To illustrate the results, an extract is discussed here, namely only the 5th train/test split.

The plot in figure 14 compares different neural networks with one layer. Networks with a maximum of ten neurons are compared. These different configurations can be seen on the x-axis. The first data point corresponds to a simple network with one neuron at only one layer. The y-axis shows the MSE values obtained with the respective trained model. As already explained, we use 50 different optimizations of each configuration to get a better idea of a potentially systematic relationship with the MSE. In the plot, each of the configurations is drawn using a different color.

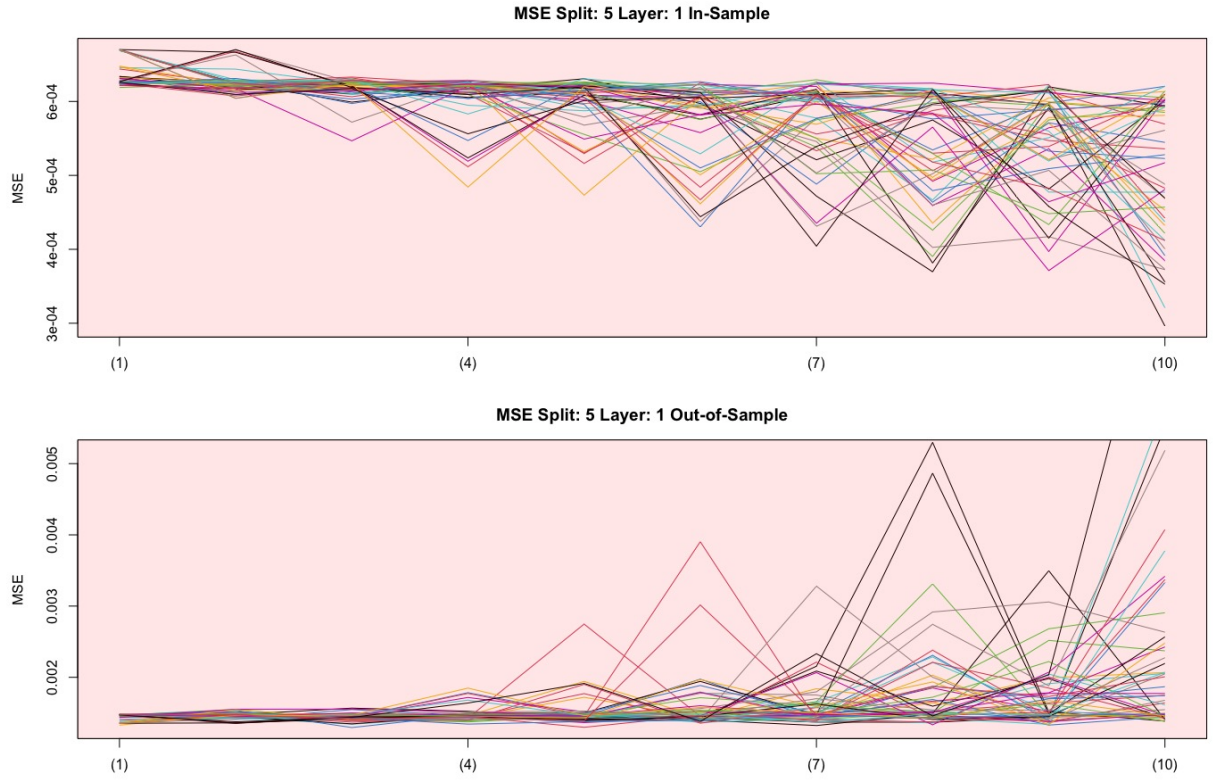

Figure 14: Fifth train/test split, 1 layer with 10 different networks.

What is already noticeable here is that with increasing complexity, i.e. with the increasing number of neurons, the in-sample MSE decreases. So it gets better. At the same time, you can see how the out-of-sample MSE increases with increasing complexity, which means that it tends to get worse.

If you add another layer to the network architecture, the number of different networks with the same number of layers also increases. In the following figure 15, the simplest network is a (1,1) network. So 2 layers with one neuron each. The most complex is a network with a (10,10) architecture.

As noted earlier in figure 14, the values for the MSE also fluctuate more and more with increasing complexity. Small in-sample MSE for more complex networks lead to rather high out-of-sample MSE. This leads us to the previously mentioned challenges in section 2.1.6.1., and that is that too many estimated parameters can lead to overfitting of the network.

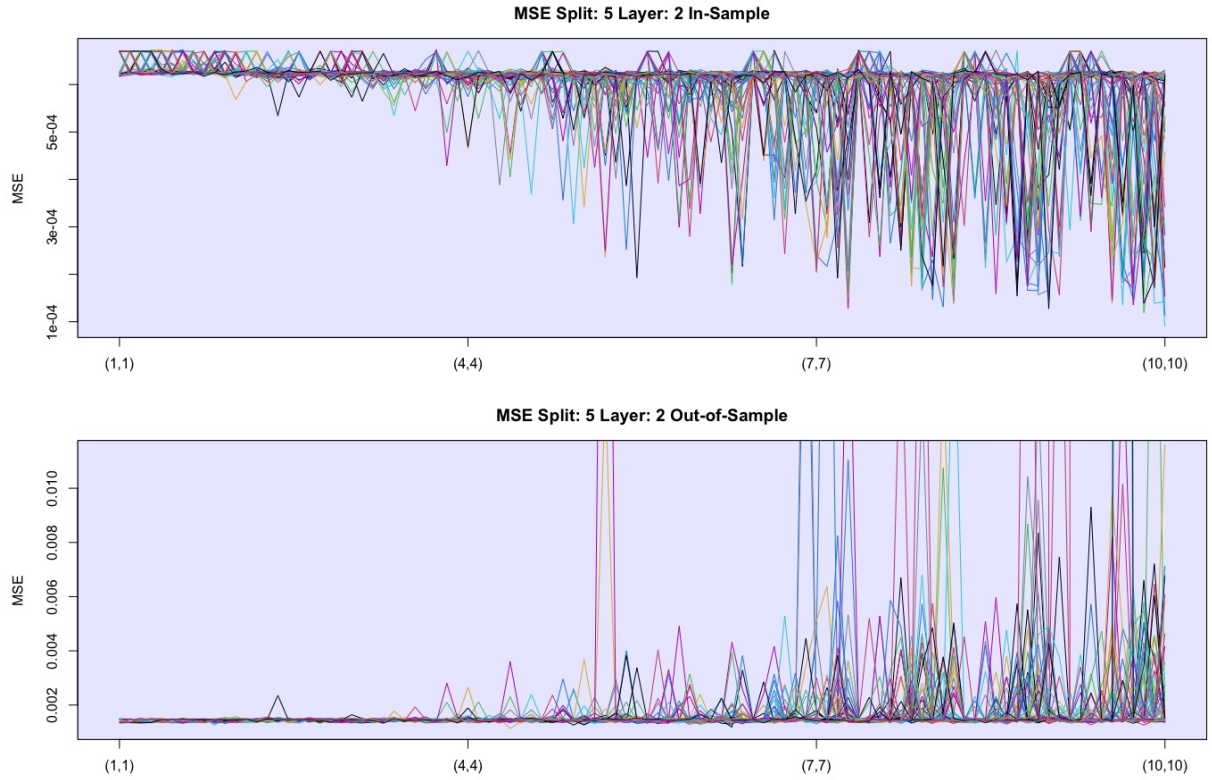

Figure 15: Fifth train/test split, 2 layers with 100 different networks.

Looking at the out-of-sample MSE's in the graph below in figure 15, you can see lines that are outside the blue rectangle. These values are extreme outliers that indicate the randomness of neural networks. This again confirms that choosing an optimal network over several equal networks (50 in our case) makes more sense than making the choice depend on only one randomly trained network. Depending on which solution the training algorithm finds, the results can be very different. The y-axis was scaled for better comparability of in-sample and out-of-sample, but one loses the overview of how much the outliers differ from the rest.

Lastly, we look at the results of the different network architectures with a third layer. In figure 16, we can see very well the inverse correlation between the in-sample and out-of-sample MSE. Again, the in-sample MSE gets better with increasing complexity while the out-of-sample MSE gets worse. There is also a certain recurring pattern that is striking. After a certain complexity, the in-sample MSE decreases steadily and then increases abruptly. The opposite pattern can also be observed out-of-sample. These patterns emerge during transitions from more complex to more simple architectures. For example, the transition from a model with (8, 10, 10), with a total of 28 neurons, to a model with (9, 1, 1) with only 11 neurons.

It is interesting that at the beginning, with the rather simple model architectures, the MSE of all realizations is very constant and only varies very slightly.

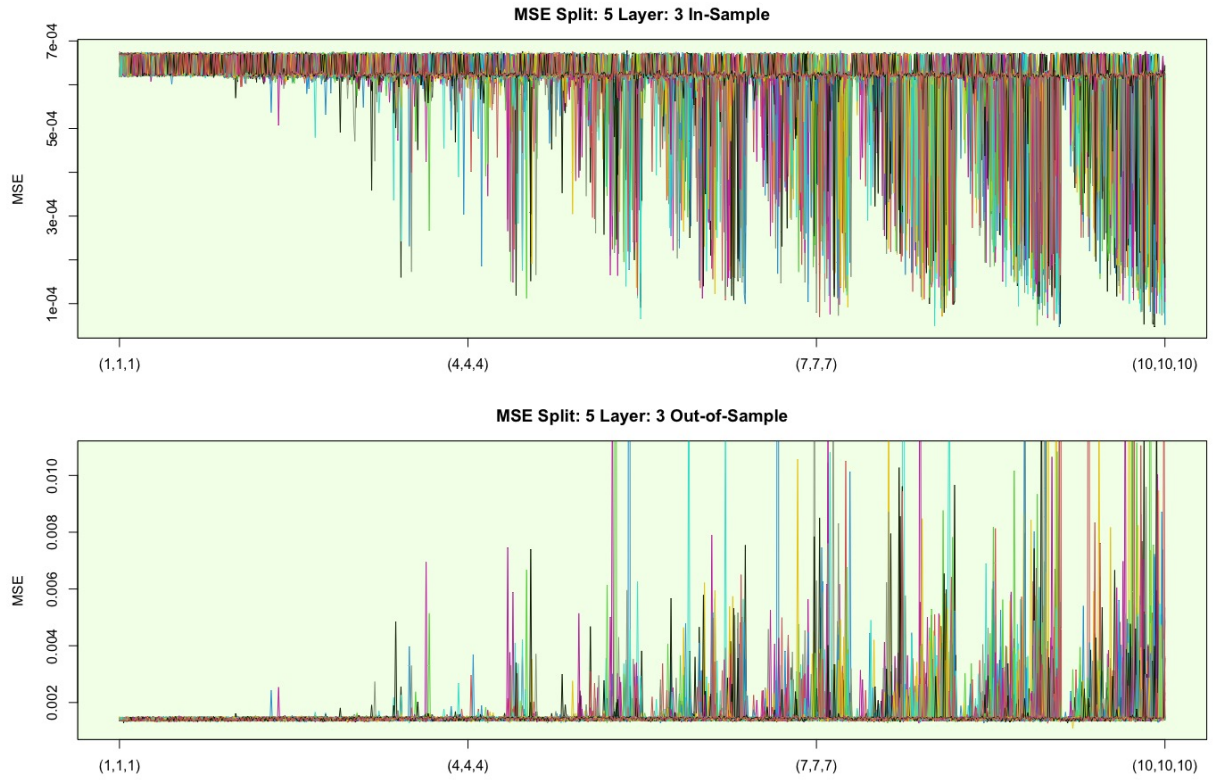

Figure 16: Fifth train/test split, 3 layers with 1000 different networks.

- ERKLÄREN, DASS ES UNMÖGLICH IST FÜR 9 VERSCHIEDENE SPLITS EIN EINE REIN VISUELLE ANALYSE DURCHZUFÜHREN

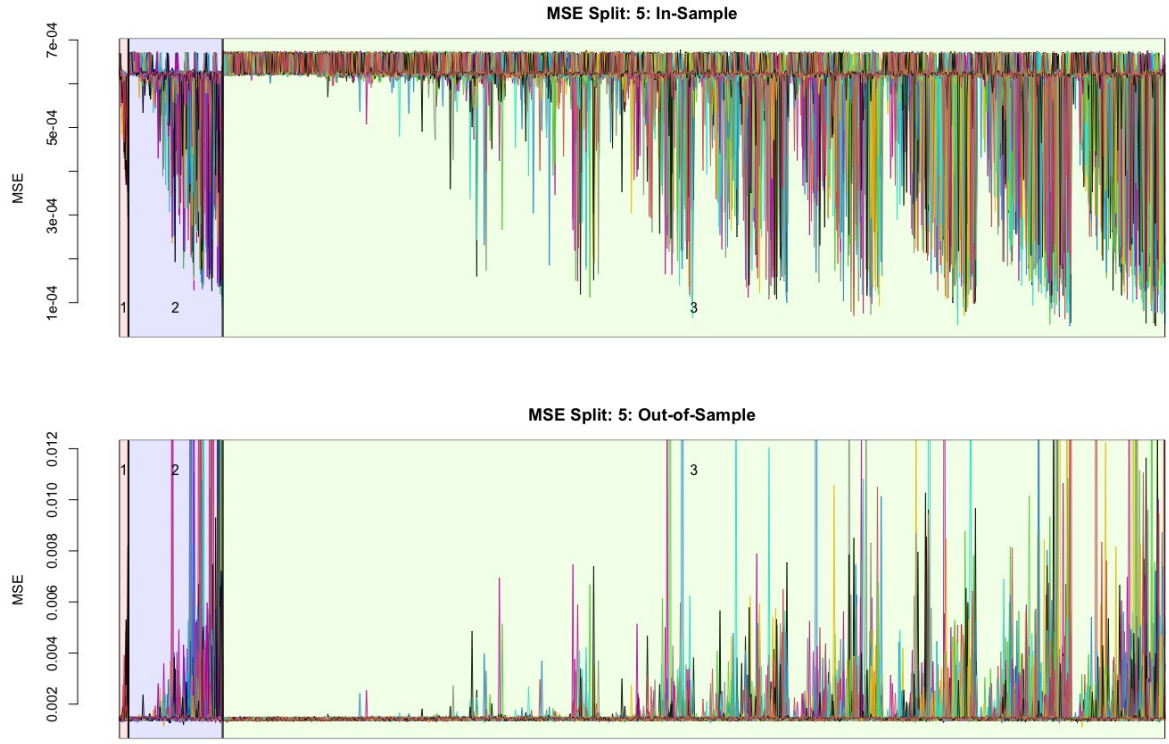

Figure 17: Fifth train/test split, all layers with 1110 different networks.

- MITTELWERTE ÜBER SÄMTLICHE DER 9 SPLITS BERECHNEN UM DEI BESTIMMUNG DES OPTIMALEN NETZTES ZU VEREINFACHEN BESPIEL: Man hat 9 Splits à je 1110 Neuronen-Layer-Kombinationen. Für jede NL-Architektur wurden 50 neuronale Netzwerke trainiert. Somit kommt man auf 55500 Netze pro Splits. Also für alleine diese Versuchreihe wurden 499500 Netzwerke trainiert. Nun möchte man diejenige NL-Architektur finden, welche im Verlauf der Zeit (Verlauf der 9 Splits) am besten geblieben/am wenigsten an Performanz verloren hat/am besten geworden ist finden. Man hat herausgefunden, dass Netzte sehr zufällig sind. Somit werden die Mittelwerte über die 50 Netzte pro Layer, pro Split gebildet. Man erhält für jede NL-Kombination 9 verschiedene Mittelwerte der MSEs. Die Suche nach einer optimalen Netzwerk-Architektur ist so um einiges einfacher. (siehe 18)

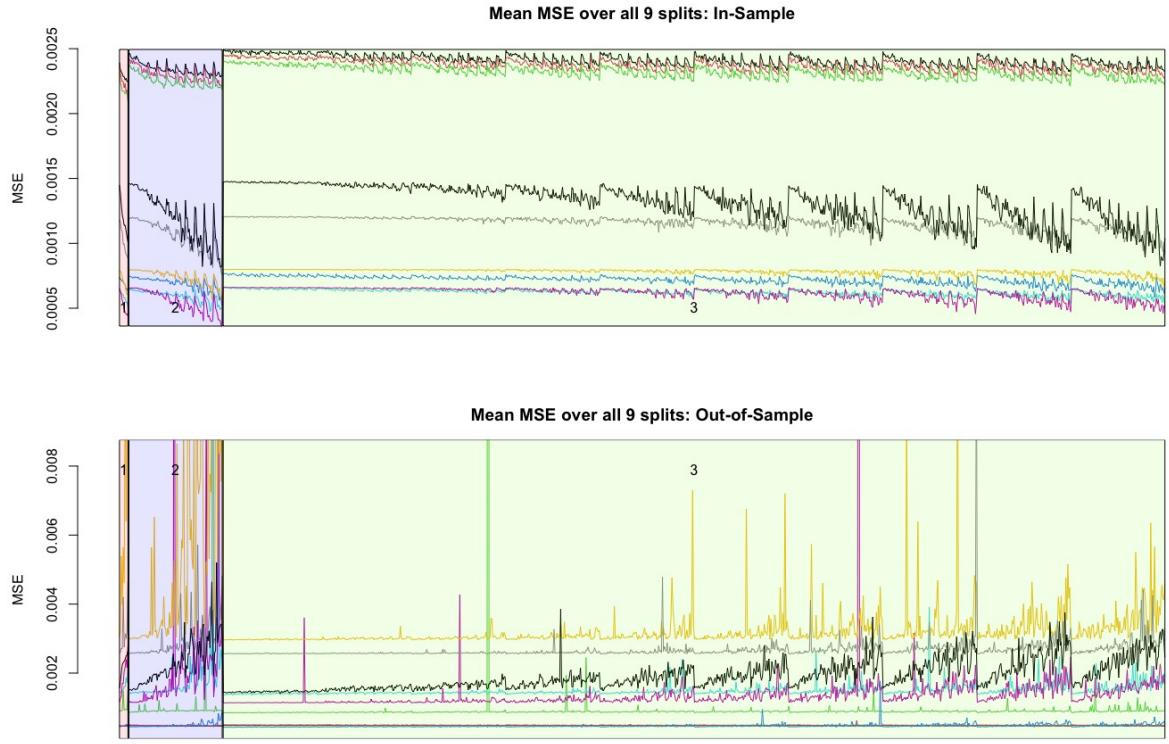

Figure 18: MSE mean over all 9 splits.

- Eventuell noch Sharpe hinzunehmen, wäre noch interessant (work in Progress)? oder evtl doch erst später im Trading?

3.3. Benchmark

To compare the models we choose two simple benchmarks the well known buy and hold and an Ar(1) process as you can see in Figure xy and Figure xxy.

3.4. Trading strategiesg

- Define trading strategies
- Sign-trading (daily)
- Vola-gewichtet trading
- Define realistic fee structure for trading (Coinbase Pro, Binance, Kraken etc.)

3.5.1. Other cryptocurrency

- Test our best model with another time series

3.6. Explainability

- Performing the predictions with the two (?) best models
- Include variations to find possible starting points for explainability (number of nodes, layers)

3.7. (Relationship between accuracy and market phase)

- Test

4. Results

4.1. Results chapterino

5. Conclusion

Best Trading Algorithm ever!

5.1. Get rich or die tryin

Neque volutpat ac tincidunt vitae semper quis. At elementum eu facilisis sed odio morbi quis commodo odio. Eget dolor

5.2. Be GME stock, or not to be GME stock

Tellus at urna condimentum mattis pellentesque id nibh. Morbi tempus iaculis urna id volutpat lacus laoreet. Sem fringilla

References

[1] F. Rosenblatt, *The perceptron: A probabilistic model for information storage and organization in the brain.* Psychological Review, 1958, pp. 386–408.

[2] P. L. B. Martin Anthony, *Neural network learning: Theoretical foundations.* Cambridge University Press, 1999.

[3] R. Rojas, *The backpropagation algorithm.* Springer Berlin Heidelberg, 1996, pp. 149–182.

[4] G. B. O. Yann Lecun Leon Bottou, *Efficient BackProp.* Image Processing Research Department AT& T Labs, 1998, pp. 1–44.

[5] L. Hunsberger, "Back propagation algorithm with proofs." https://www.cs.vassar.edu/~hunsberg/cs365/handouts-etc/backprop.pdf (accessed Mar. 21, 2021).

[6] M. A. J. I. Hassan Ramchoun Youssef Ghanou, *Multilayer perceptron: Architecture optimization and training.* International Journal of Interactive Multimedia; Artificial Intelligence, 2016, p. 26.

[7] A. Karpathy, "A recipe for training neural networks." https://karpathy.github.io/2019/04/25/recipe/ (accessed Mar. 24, 2021).

[8] M. N. S. S. Ke-Lin Du, *Recurrent neural networks.* Springer London, 2014, pp. 337–353.

[9] S. Y. Fei-Fei Li Justin Johnson, *Lecture 10: Recurrent neural networks.* Stanford University, 2017.

[10] R. S. M. Francis X. Diebold, *Comparing predictive accuracy.* University of Pennsylvania, 1995.

[11] U. Triacca, *Comparing predictive accuracy of two forecasts: The diebold-mariano test.* Universita dell Aquila, NA.

[12] S. Nakamoto, *Bitcoin: A peer-to-peer electronic cash system.* online: www.bitcoin.org, 2008, p. 9.

[13] U. W. Chohan, *A history of bitcoin.* University of New South Wales, Canberra, 2017.

[14] J. Edwards, "Bitcoins price history." https://www.investopedia.com/articles/forex/121815/bitcoins-price-history.asp (accessed Mar. 01, 2021).

[15] R. R. Georgios Sermpinis Andreas Karathanasopoulos, *Neural networks in financial trading.* Springer Science+Business Media, LLC, part of Springer Nature 2019, 2019, pp. 204–308.

Attachment

This project work is created with R-4.0.2 , RStudio Version 1.4.904 and RMarkdown in collaborative working via Git / Github