Neural Networks and the Stock Market

- a practical experiment on predicting price changes

Lars Åström
Bachelor Thesis
Department of Economics
Supervisor: Peter Jochumzen

July 26, 2018

Abstract

In the 19th century, gold diggers emigrated from Europe to North America with the hopes of a brighter future. Gold diggers today might look to the stock market with hopes of finding the key to incredible wealth. The introduction of neural networks has revolutionized data analysis and the possibility of using them to do significantly better than chance on the stock market is investigated in this report. The aim of the report is to investigate a short period after a stock drastically decreases in price with neural networks.

Firstly, neural networks are constructed and trained to estimate price changes. Thereafter an investment strategy is constructed and evaluated. The results of the investigation is that it is possible, under some assumptions, to do significantly better than chance in the stock market. It appears to be possible to do this for all days from the third to the ninth after a drastic crash. The predictions are significantly better than chance, however probably not large enough to compensate for transaction costs.

It appears that the gold diggers of the 21st century, operating on the stock market, have to wait a while longer. It is possible to guess significantly better than chance if the price will increase or decrease, however the difference might not be large enough to profit from the trades. The networks that are developed in this thesis are not good enough to give rise to sufficient profits on the stock market. However, maybe it is possible to do so with slight changes in the structures of the networks, and thus earn incredible profits by investing wisely on the stock market?

Key words: *Neural Networks, Machine Learning, Stock Market, Efficient Market Hypothesis, Artificial Intelligence*

Contents

1 Introduction, aim and problem formulation

Predicting if, and when, a stock increases in price has been a big question for capital investors since the debut of the stock market. The introduction of large technical analyses has given hope to investors who wish to easily earn money by simply analyzing previous stock price movements.

Analyzing data series is a common problem within various fields of study, and recently neural networks have made their great breakthrough. These networks try to simulate the functioning of the human brain, in order to teach the computer to find patterns people cannot find themselves.

It has long been an unproven, but still widely established, hypothesis that it should not be possible to detect patterns to beat the market. This has seemed to be true for humans, but possibly computers can detect patterns even humans could not find. Furthermore computers have over and over proved their ability to teach themselves how to detect patterns, meanwhile humans only can teach computers patterns they can find themselves. This makes it possible to hope that neural networks might be able to predict stocks better than just randomly guessing.

The aim of this thesis is to investigate if neural networks can be used to predict changes of stock prices. More specifically the unstable period after a drastic decrease in price will be investigated. The main question is "Is it possible to predict increases and decreases in stock prices shortly after a drastic decrease in price with a neural network?"

2 Limitations and theory

Throughout this paper, the following convention will be used: In the time series data day 0 is defined to be the day of the crash, and the following days are numbered increasingly.

The rest of this section is dedicated to assumptions, limitations, general theory and finally a brief presentation of previous papers correlated to this one. The theory is intended to explain the main concepts of the method and results.

2.1 Assumptions and Limitations

In the thesis some assumptions are made, and some limitations to the problem are used. The underlying assumptions are:

- No transaction costs are present for stock investments.
- It is possible to short sell all stocks at no cost.
- Stock prices are never more than doubled in one day.

The investigated problem is restricted to

- Only changes of stock prices directly after a crash are investigated.
- Only a period of a total of ten days is investigated for each stock and crash, meaning that only nine first business days after the crash are investigated.

2.2 Introduction to neural networks

Neural networks were developed in the middle of the 20th century, and is a way to mathematically resemble the human brain in order to try to teach a computer to find patterns in data sets. In order for a neural network to succeed, huge data sets are usually needed and these are used by the network to learn. (Bishop, 2006) The reason large sets are needed is that otherwise there is a major risk of overfitting, which is that the network adapts and becomes biased on noise specific to the dataset. If the dataset is big however, there is still noise in the data but it will be more evenly distributed. The need of large data sets was a major problem in the first centuries after neural networks were developed, since computers were very slow compared to those that are used today.

(Rogers & Girolami, 2016) Now, in the 2010's, neural networks really have gotten their breakthrough and are used in various fields solving different complex problems.

This subchapter will briefly introduce the concept of neural networks, however no deeper insights will be presented - only the most fundamental concepts are introduced. It should be noted that all the methods described here are not complete, but they should rather be viewed as the main principals of the methods of a neural network. In the end of this section, a source for further understandings of neural networks will be presented.

Construction of network

A neural network can be viewed as a directed graph, where all nodes in previous layers are connected to all nodes in the following layer. (Rogers & Girolami, 2016) In figure 2.1 a basic neural network is shown, with one input layer consisting of five data points, two hidden layers with four nodes each and one output layer, consisting of a single output data. The arcs between a layer and its successor are supposed to visualize the connections that transfer information from one layer to the next one.

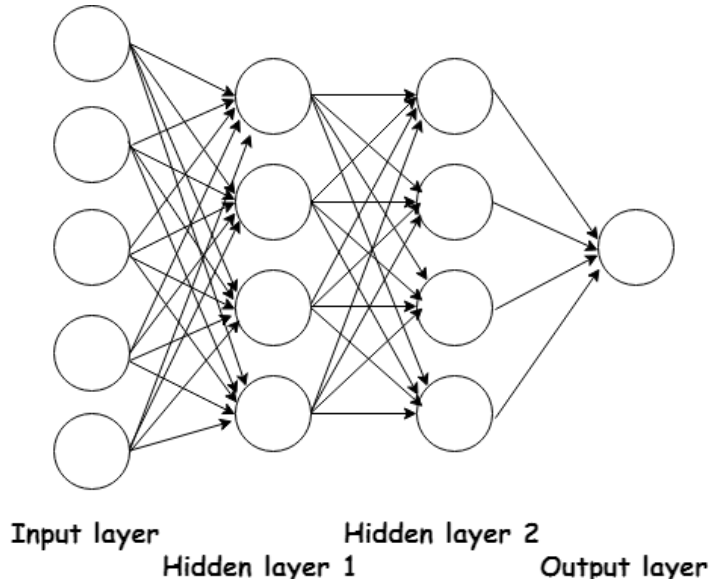

Figure 2.1: A schematic illustration of a basic neural network.

To illustrate what a layer might do, consider the following problem which is widely used to explain neural networks and deep learning. (Rogers & Girolami, 2016) Input is a picture of a hand-drawn digit, and the goal is to predict which digit the user tried to write. Then the input layer might consist of the picture (or more precisely the pixels of the picture), the first hidden layer could find patterns like straight lines or circles, the second hidden layer could combine the patterns and finally an output is produced - which is a guess of

which digit was drawn.

Transfer of information in the network

The previous description of a neural network and what the layers might do is a good thought experiment, but not how the network really works. Instead a neural network is trained by optimizing parameters in the model. (Bishop, 2006) First of all it is important to see how information is transfered from one layer to the next. Assume two consecutive layers are given - layer P and layer S, where P and S are acronyms for *Predecessor* and *Successor*. These might be any of the consecutive layers in figure 2.1, for example input layer and hidden layer 1, or hidden layer 2 and output layer. In a general network there might be many more hidden layers than in figure 2.1, although each step from one layer to its successor is performed in the same fashion.

Assume that layer P has n_P nodes and layer S has n_S nodes. Then there are $n_P \cdot n_S$ weights between the layers, since all nodes in P are connected to all nodes in S. Let $w_{i,j}$ denote the weight from node j in layer P to node i in layer S. Also let the previously calculated values of the nodes in P be denoted by $\begin{bmatrix} p_1, p_2, \dots, p_{n_P} \end{bmatrix}$. Now the values in the nodes in S are being calculated, which are denoted by $\begin{bmatrix} s_1, s_2, \dots, s_{n_S} \end{bmatrix}$. In matrix notation, where W is the matrix with dimensions $n_S \times n_P$ and the element in position (i, j) is $w_{j,i}$, $P = \begin{bmatrix} p_1, \dots, p_{n_P} \end{bmatrix}^T$ and $S = \begin{bmatrix} s_1, \dots, s_{n_S} \end{bmatrix}^T$ the transformation could be written as

$$S = WP,$$

which is the same as saying

$$s_i = \sum_{j=1}^{n_P} w_{i,j} \cdot p_j, \text{ for } i = 1, 2, \dots, n_S.$$

There is a major limitation with this transformation in that it is linear. This would imply that even the output layer would be a linear function of the input layer - which is not desired. (Rogers & Girolami, 2016) Therefore an activation function is used - which is an non-linear function - for example tanh or the step-function. Also a bias is used, which is a constant that is added to each node before applying the activation function. Let AF denote the activation function and let $B = \begin{bmatrix} b_1, \dots, b_{n_S} \end{bmatrix}^T$ be a bias vector, then the transformation can be written as

$$S = AF(B + WP),$$

which is the same as

$$s_i = AF(b_i + \sum_{j=1}^{n_P} w_{i,j} \cdot p_j), \text{ for } i = 1, 2, \dots, n_S.$$

This transformation from one layer to the next is what actually is used in neural networks.

Training of a neural network and the loss function

The training of a neural network is when the weights are optimized in order to minimize the error of the predictions. This is done by optimizing with respect to a loss function, which is a measurement of the error. (Rogers & Girolami, 2016) Usually the mean absolute error or the mean squared error of the predictions is used as a loss function, but the important thing is that the function is continuous, differentiable and that a smaller value means a better prediction.

Let L denote the loss function. Then the total loss for a given training set is a function of all weights and biases:

$$L = L(W^*, B^*),$$

where W^*, B^* denote all weights and biases between all pairs of consecutive layers. There are many weights and biases in a general neural network. In the very small network in figure 2.1 the number of weights and biases are described in the following table:

Layer pair	**Weights**	**Biases**
Input layer to hidden layer 1	$5 \cdot 4 = 20$	4
Hidden layer 1 to hidden layer 2	$4 \cdot 4 = 16$	4
Hidden layer 2 to output layer	$4 \cdot 1 = 4$	1

As can be seen in the table there are a total of 49 weights and biases in the small neural network in figure 2.1. This means that the loss function is a function of 49 variables - and in general neural networks it is a function of thousands of variables. The goal is to minimize the error, which is minimizing the loss function. In order to do this, partial derivatives are used to see if a weight has a positive or a negative impact in the loss.

In each training step the partial derivatives are considered; which are

$$\left[\frac{\partial L}{\partial w_1^*}, \ldots, \frac{\partial L}{\partial w_{w_n}^*}, \frac{\partial L}{\partial b_1^*}, \ldots, \frac{\partial L}{\partial b_{b_n}^*}\right],$$

where $w_1^*, \ldots, w_{w_n}^*$ are all the weights and $b_1^*, \ldots, b_{b_n}^*$ are all the biases. Then for all weights with positive partial derivative the weight is decreased and for all weights with negative partial derivative the weight is increased - hence the weight is moved in the opposite direction of the partial derivative. This method is called "Gradient descent". (Bishop, 2006) In one dimension, this is very easily understood and shown in figure 2.2. As can be seen graphically in the figure, the minimizing method should move right from the left tangent point and left from the right tangent point in order to approach the minimum point of the function. This is the same as moving opposite to the gradients direction at the points.

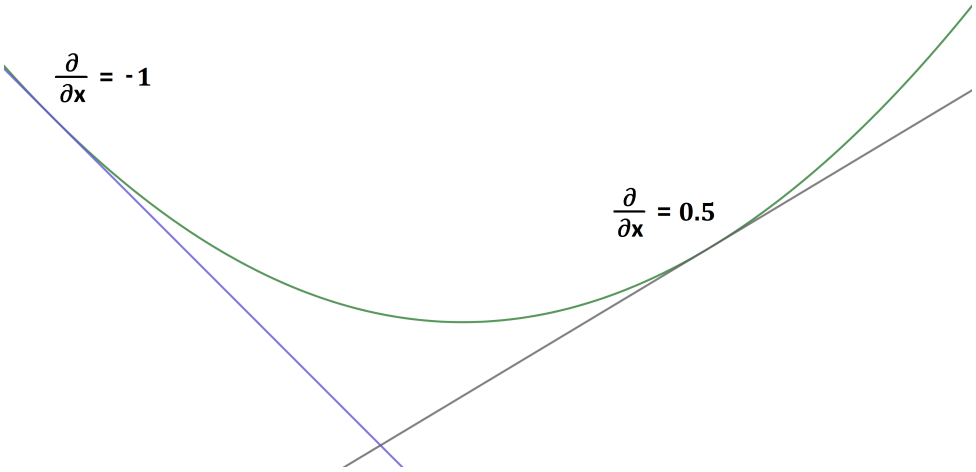

Figure 2.2: Illustration of gradient descent in one dimension.

In more complex neural networks, more sophisticated methods to reach the minimum have to be used, otherwise too much computer power would be needed. In simple cases gradient descent is sufficient, and all other methods are somewhat similar - although the methods differ in some ways the main principals are the same. (Bishop, 2006)

For a deeper understanding of neural networks the reader is referred to the great course literature books used in many introductory Machine Learning courses, namely *A first course in Machine Learning* by Rogers and Girolami (2016) and *Pattern Recognition and Machine Learning* by Bishop (2006). These books are used in the introductory courses of Machine Learning at, among others, Chalmers Institute of Technology and the faculty of Engineering at Lund University, respectively. Furthermore these books offer a deep understanding of all concepts used in this thesis. If the reader instead is interested in a brief introduction in order to get a grasp of the most important concepts of Machine Learning, then the videos explaining deep learning and neural networks by the Youtube-channel "3blue1brown" (2017) are recommended.

2.3 Short sale investments

Short sale investments is a sort of opposite of long investments - which is when a stock is bought. This method is used when the investor believes the stock price is going to decrease. Short sale investments can be viewed as the following steps:

1. Loan a stock day i

2. Sell the stock day i

3. Buy the stock day $i + 1$

4. Return the stock day $i + 1$.

The horizon of investment does not have to be one day, although in this thesis all investments are done from one day to the next. The profit of this investment strategy, given stock prices S_i and S_{i+1} is $S_i - S_{i+1}$, or in relative prices

$$\frac{S_i - S_{i+1}}{S_i}.$$

In the calculations in the report, it is assumed that a loan of stocks with equal size as the current wealth is taken. This means that if the wealth today is W then using a short sale strategy gives a value the next day of

$$W \cdot \left(1 + \left(1 - \frac{S_{i+1}}{S_i}\right)\right) = W \cdot \left(2 - \frac{S_{i+1}}{S_i}\right),$$

hence the change factor is $2 - \frac{S_{i+1}}{S_i}$. Here there is a problem if $\frac{S_{i+1}}{S_i} > 2$, since then the change factor would be negative, meaning that the wealth would be negative. This is a risk while using a short sale strategy; if the stock price more than doubles the wealth would be less than zero. It is assumed that this never occurs shortly after a drastic decrease in price, which is not too controversial.

Short sales are possible to trade on some market platforms - where it is possible to invest with a payoff scheme similar to the one that has been derived in this subchapter. Then the steps 1-4 are not actually performed, but rather a trade contract with similar payoff structure is constructed and used instead.

2.4 Efficient Market Hypothesis

The efficient market hypothesis states that the price of a stock reflects all available information. This in turns implies that it is impossible to beat the market in the long run. There are three forms of the hypothesis - weak, semi-strong and strong. The weak form states that the prices reflect all previous information; the semi-strong states also that the price reflects all new public information instantaneously; and the strong states that the price even reflects insider information instantaneously. (Fama, 1970) Even though this hypothesis, in all its forms, implies that it should not be possible to profit through technical analysis, this thesis does not necessarily contradict the efficient market hypothesis. The reason for this lies in the assumptions that short sales can always be done at no loan rate, and that there are no transaction costs.

Critiques against the hypothesis

There is one large critique against the efficient market hypothesis, which is somewhat related to the hypothesis behind this thesis and is presented by behavioral economists. Here the counter hypothesis is that human behavior is irrational, not rational as usually is assumed, and not only reflecting the information about a stock. (Malkiel, 2003) One example of this is "herding". Herding is when lots of people make the same decision, others follow - possibly without any kind of investigation of the information. This can be the case if for example experts recommend a stock. One consequence of herding is that overshoots might appear. Overshoots is when for example the change in price first increases more than it should, and thereafter it decreases.

2.5 Evaluation of estimations

In order to evaluate the performance of the neural networks, some metric has to be constructed. The neural networks are constructed to estimate the relative price change a given day after a crash. From these estimations a decision has to be made regarding investing or short selling the stock. This metric is defined by the following investment scheme.

Given is a vector of estimated price changes $P^e = \left[P_1^e, P_2^e, \ldots, P_n^e\right]$, and a vector of real price changes for the corresponding stocks $P^r = \left[P_1^r, P_2^r, \ldots, P_n^r\right]$. Here P denotes relative price change, superscript e is short for estimated and superscript r is short for real. Furthermore the numerical index is the index of the current stock. Note that when the price changes are estimated, the neural network of course does not know the real price changes.

The decision for stock i is given by:

- If $P_i^e > 1$, the strategy invests in the stock,
- If $P_i^e \leq 1$, the strategy short sells the stock.

Let C_i denote the change factor of the portfolio given the decision. This means that C_i is given by:

$$C_i = \begin{cases} P_i^r, & \text{if the strategy invests in the stock, i.e. } P_i^e > 1 \\ 2 - P_i^r, & \text{if the strategy short sells the stock, i.e. } P_i^e \leq 1 \end{cases}$$

From this the total change of the portfolio, C, is given by:

$$C = C_1 \cdot C_2 \cdot \ldots \cdot C_n = \prod_{i=1}^{n} C_i.$$

2.6 Cross Validation

Cross validation is a statistical method used to validate different estimation methods. The total data set is divided into two parts - the training and the testing set. The former is used to calibrate the model - in the case of neural networks it is the weights and biases - while the latter is used to test the performance. Of course a model cannot be tested on the same data is has been trained, since the model simply could remember the answers - which is called overfitting. Therefore the test set is hidden from the training part. (Rogers & Girolami, 2016)

One common type of cross validation is "Leave-p-out cross validation". Here p data points are left out from the training, while the rest are used. Then the left out data points are used for testing. (Rogers & Girolami, 2016) This procedure may be performed for all possible choices of p data points as test data. There is a major problem with this method however, namely that there is an enormous number of ways to pick p data points out of all data points. If for instance 100 data points are going to be picked out of 10000 data points in total, then the number of ways to combine those is far more than 10^{200}. Therefore, it is not possible to perform a complete Leave-p-out cross validation.

An alternative way is to perform some randomly picked cross validations among the possible ones in Leave-p-out cross validation. Since the mean performance is of interest, this can be estimated for all possible ways to Leave-p-out, only by investigating a rather small subset of those.

2.7 Confidence interval of mean estimation

Let $a = \big[a_1, a_2, \ldots, a_n\big]$ be a large random sample from an arbitrary distribution. Let μ be the expected value and σ be the standard deviation of the distribution. Furthermore let $\bar{a}$ denote the mean and $\bar{s_a}$ denote the standard deviation of the sample. If n is large, then

$$\bar{a} \sim \mathcal{N}\big(\mu, \frac{\sigma}{\sqrt{n}}\big)$$

according to the central limit theorem. This yields a confidence interval of μ, given by

$$I_\mu = \Big(\bar{a} - z_{\alpha/2} \cdot \frac{\bar{s_a}}{\sqrt{n}}, \bar{a} + z_{\alpha/2} \cdot \frac{\bar{s_a}}{\sqrt{n}}\Big),$$

with approximate confidence level $1 - \alpha$. Here $z_{\alpha/2}$ denotes the value the normal inverse cumulative distribution function evaluated at probability $\alpha/2$. (Blom, Enger, Englund, Grandell & Holst, 2005)

2.8 Arithmetic vs Geometric mean

When evaluating a mean performance of a given investment strategy, either an arithmetic or a geometric mean can be used. The interpretation of the two are quite different, and therefore they should be used differently. The arithmetic mean of $\{a_1, a_2, \ldots, a_n\}$, which is what usually is meant by "the mean", is given by

$$\bar{a} = \frac{1}{n} \cdot \sum_{i=1}^{n} a_i.$$

This can be used to answer questions like "Given an investment of size q in n portfolios simultaneously and independently, what is the total performance?" If a_i is the change factor of the i-th portfolio, then the total profit, Π, from all the n portfolios is given by

$$\Pi = n \cdot q \cdot \bar{a}.$$

The geometric mean is instead a multiplicative mean, whereas the arithmetic is an additive, and is given by

$$\tilde{a} = \left(\prod_{i=1}^{n} a_i \right)^{\frac{1}{n}}.$$

This mean is used to answer questions like "Given an investment of size q in n portfolios sequentially, where the amount invested in the $i+1$-st portfolio is what was left from the i-th, what is the total performance?" If a_i is the change factor of the i-th portfolio then the total profit, Π, of the investment in all n portfolios sequentially is given by

$$\Pi = q \cdot \tilde{a}^n.$$

It should be noted that the arithmetic mean is always larger than the geometric mean, which follows from the arithmetic-geometric mean inequality. This means that the arithmetic mean might be larger than 1, which corresponds to an increased value of the portfolio, while the geometric mean is smaller than 1. In this case, the estimated value is larger by investing in n different portfolios simultaneously. On the other hand if both the arithmetic and the geometric means are larger than 1, it is almost always preferable to invest sequentially since profits from previous investments also increase - just like interest on interest. This means that a much smaller amount can be invested, and still the same expected profit can be obtained.

Note that whenever "mean" is written in the report it refers to the arithmetic mean, and whenever a geometric mean is meant this is clearly stated.

2.9 Overview of previous work on Neural Networks in finance

The approach of trying to predict stock prices with neural networks is quite natural. Some reasons for this are that the amounts of data of historical stock prices are huge, and at the same time the possible earnings of predicting stock prices accurately are enormous. This also means that the work done in this thesis is not the first of its kind; trying to predict stock prices with neural networks has been a field of study since at least the 1990s and still lots of papers are produced annually on the subject. In this subchapter some of the most interesting papers are presented briefly, just to give the reader a hint of what has been investigated related to neural networks and stock prices.

In 1990 Kamijo and Tanigawa tried predicting stock price patterns by a recurrent neural network. Then the authors investigated certain so called triangular patterns, which are related to future stock price movements. Then Saad, Prokhorov and Wunsch (1998) tried predicting stock trends and compared three different kinds of neural networks. The authors concludes that all three are feasible to use. These two papers are both cited various times and can be viewed as some sort of foundation for the field.

Looking at some even closer related to the subject investigated in this thesis Malliaris (1994) looked at modelling the S&P500 index using neural networks. Malliaris' paper supports that the market has an underlying chaotic structure.

In 2010 Li and Ma appraised the potential of using neural networks when trying to predict financial systems. In their paper future possible opportunities of research are presented, along with presentations of neural networks and their application to exchange rates. Lastly Gündüz, Yaslan and Çataltepe (2018) tried predicting stock prices of nine banks using LSTM-neural networks. Their results were better than the random benchmark, and the approach used in their paper is very similar to the one of this thesis although the time frame of investigation is completely different.

3 Method

There are five steps in the analysis performed in this paper, all of which will be thoroughly explained in the following subchapters. Firstly, data is extracted and reshaped to only contain the relevant days and stocks. Secondly, the neural networks are constructed and trained. Thirdly, the networks estimate price changes of test data. Thereafter some choices for the networks are presented and finally the mean performances of the networks are analyzed using statistical methods.

3.1 Construction of data sets

Firstly historical daily stock prices are found in a database. (Kaggle, 2018) Daily data (closing values) are used for the last five years from S&P500. Then data is transformed from stock prices to relative price changes. Thereafter the relevant data is filtered out, obtaining a total data set of periods of ten consecutive relative price changes. The relevant data is defined to be when a large decrease in price is found, and then the nine days following the crash. A large decrease is defined to be a decrease of at least 5%. This yields a time series of ten data points, where the first one always is less than 0.95.

Then this total data is further divided into two sets called training and testing data. In order to prevent the network from overfitting, the test set is used to evaluate the performance of the network, hence the testing is done by cross validation. When multiple rounds are run, the division into training and testing data is randomized each time.

3.2 Construction and training of the neural networks

Several neural networks are constructed and trained on the training data set, one for each day to predict. This means that one neural network is constructed to predict the change of price for day 1 after the crash, given the magnitude of the crash which is on day 0. Another neural network is constructed for predicting the price day 2, given the previous data from day 0 and 1, and so on. This means that a total of nine neural networks are constructed and trained.

The model used in this thesis is a LSTM-network and the training is performed in a number of epochs, where each epoch consists of training the network on the training data once. The training is done by giving the network both an input and the corresponding answer, then the network optimizes its weights and biases in order to minimize the loss function.

3.3 Estimation of test data

When the network has been trained for some epochs the values of the test data is estimated - given the inputs. Then the network estimates the relative price change the given day of estimation. This estimation is further used to decide how to invest the following day.

Note that it is not the trading strategy that is given by the neural network, it will only provide you with a prediction of the relative price change. From this, the user can choose different strategies for investment. The strategy used in the following investigations of this report is the one described in section 2.5.

3.4 Description of choices

When a network is constructed and trained some choices are made. First of all some parameters of the network are chosen:

- Number of layers - in the networks only one hidden layer is used.
- Number of nodes in each layer - in the networks the layer consists of 20 nodes.
- How much data to use as training and test data - these proportions are chosen to be 90% and 10% of all data that is collected, respectively.
- How many epochs the network is trained for - in the experiments presented in this report the networks are trained for 200 epochs.

Besides the characteristics of the network, the method of decision making has to be chosen, which is how to invest given the estimations made by the network. In all cases in this report the method described in section 2.5 is used with performance of a neural network, C, given by

$$C = \prod_{i=1}^{n} C_i,$$

where C_i are the change of the value of the portfolio. Lastly, the change factor, C, is transformed by subtracting 1 and multiplying by 100, in order to get the relative price change.

3.5 Statistical analysis of mean performance

When investing in stocks, one important metric to consider is mean performance. In order to get an estimate of the mean performance of the neural networks, multiple rounds are

run. In each round each of the steps in sections 3.1 - 3.3 are performed, with the choices presented in 3.4. Then the performance is measured, according to section 2.5. In total 100 rounds are run and the performance for each round is calculated. The mean, the standard deviation and the confidence interval of the mean performance are estimated - all according to section 2.7.

Once the confidence intervals have been calculated, they are analyzed to answer the hypothesis "The mean performance is $C = 0$" - with alternative hypothesis "The mean performance is not $C = 0$." This is done by a hypothesis test, and the hypothesis is rejected if the confidence interval for C does not contain 0.

4 Results

When mean performance is estimated for each individual stock, the results in figure 4.1 are given for each investigated day. The evaluations are made for each day, one at the time. For each day to predict, the stocks are given from 100 individual and separate rounds, and for each round 355 stock prices are predicted and evaluated.

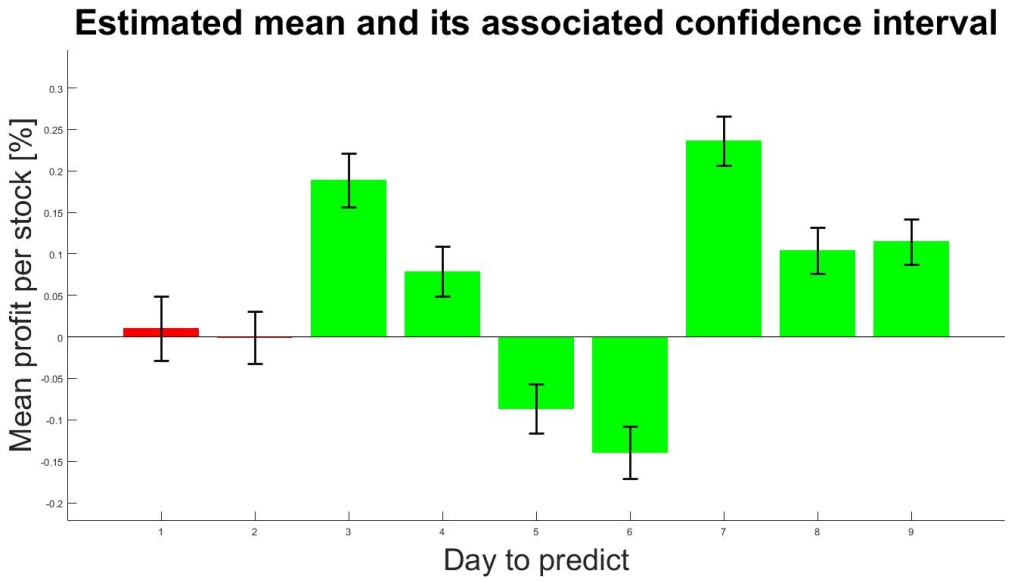

Figure 4.1: Estimation of mean performance and confidence intervals for a single stock, divided into separate days.

Looking at this for an estimated annual profit, the results in figure 4.2 are given for each investigated day. For each day there have been 100 rounds of testing. In each round 355 stocks are predicted. Thereafter two rounds are joined together to form an estimation of annual profit since there are on average 710 drastical decreases in price per year in the observed data.

In both figures green bars indicate that the mean value is significantly different from 0, hence the null hypothesis is rejected. Red bars indicate that the mean is non-significantly different from 0, hence the null hypothesis cannot be rejected. Both for individual stocks and for annual profit the mean is different from 0 for days 3-9.

The rest of this section will consist of a graphical representation of the profits from investing in stocks. This will be presented by one figure per day and then summarized in a table. For each day, all predictions of change factors are used and plotted in increasing order. This means that the first stock is defined to be the one with smallest change factor, while the last stock is defined to be the one with largest change factor. It will be seen that

the days with means significantly different from 0 also will have a prediction accuracy for a price increase or decrease quite different from 50%.

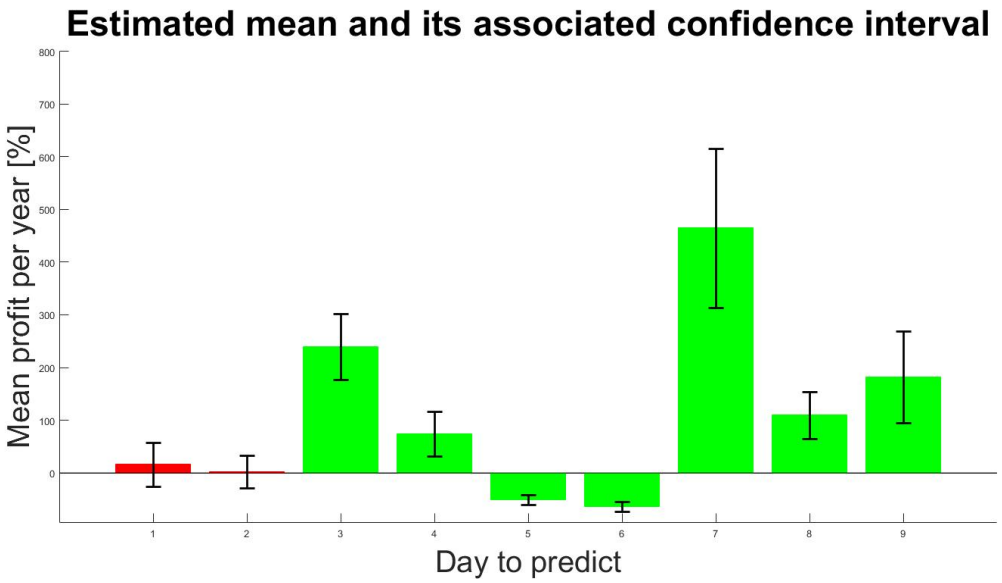

Figure 4.2: Estimation of mean performance and confidence intervals for one year of investments, divided into separate days.

In order to make the reading easier, only three figures are shown in section 4.1 while the rest are shown in appendix A. The three chosen figures each show one interesting result each, which are described in the following section.

4.1 Predictions for each day

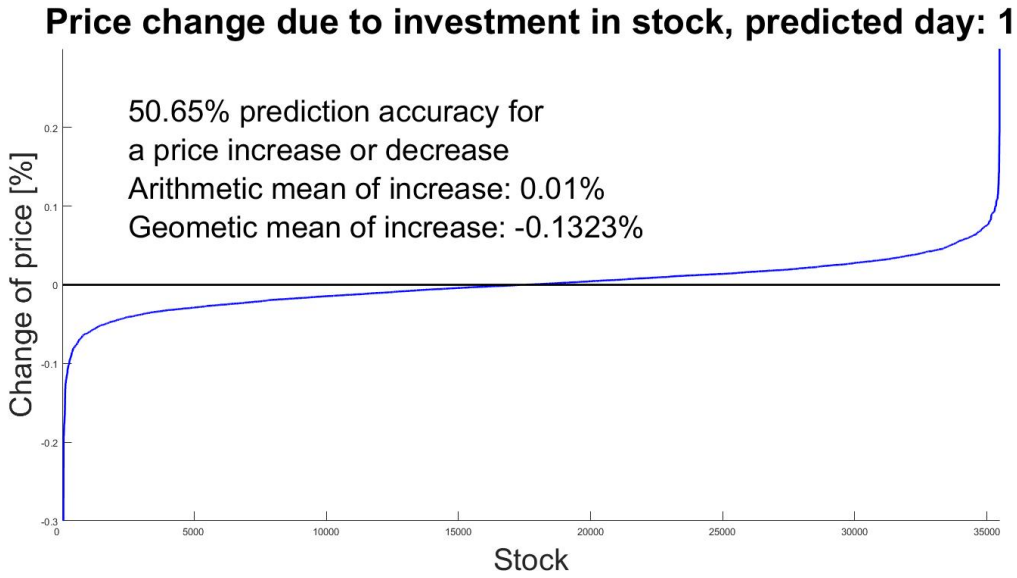

Figure 4.3: Change factors given by the investment strategy for each stock.

On day 1, the performance of the neural network is not significantly different from 0 in figures 4.1-4.2. This can also be seen in the corresponding figure for day 1, namely figure 4.3, where the prediction accuracy is very close to 50%.

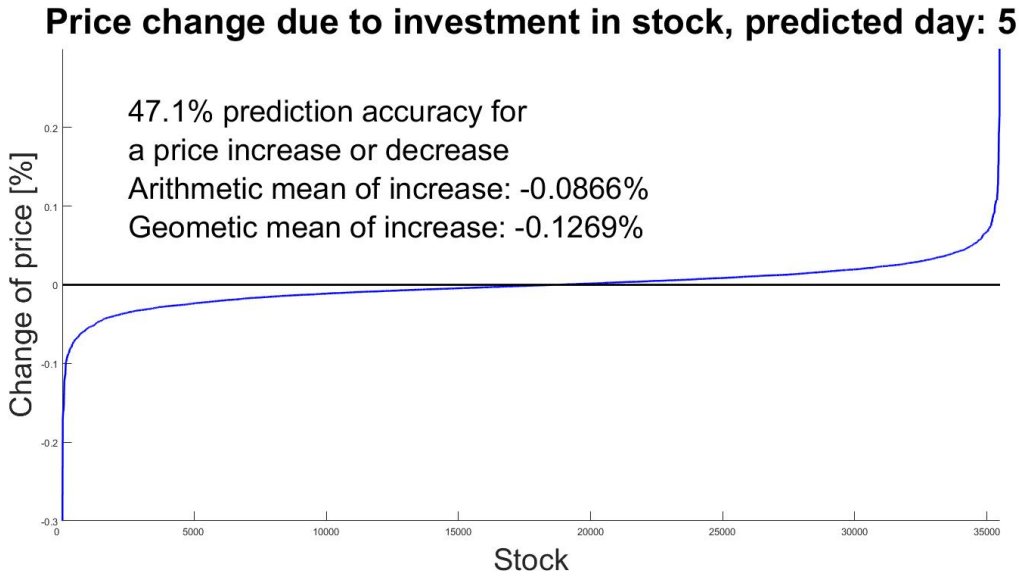

Figure 4.4: Change factors given by the investment strategy for each stock.

On the other hand, the performance of the neural networks of day 5 and 7 are significantly different from 0 in figures 4.1-4.2, with smaller than 0 on day 5 and larger than 0 on day

7. This can be seen in figures 4.4-4.5, where the prediction accuracies are well below and well above 50% for day 5 and 7, respectively.

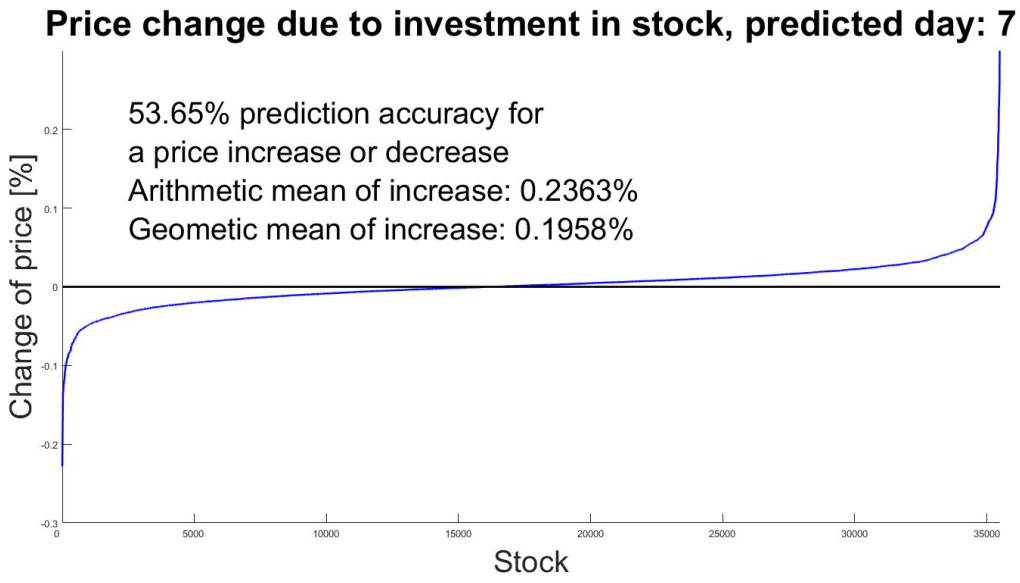

Figure 4.5: Change factors given by the investment strategy for each stock.

When the statistics of figures 4.3-4.5 and A.1-A.6 are extracted, the following table is given.

Day	Prediction accuracy	Arithmetic mean	Geometric mean
1	50.65%	0.0100%	-0.1323%
2	49.56%	-0.0006%	-0.0474%
3	53.92%	0.1890%	0.1404%
4	51.17%	0.0786%	0.0361%
5	47.10%	-0.0866%	-0.1269%
6	46.95%	-0.1394%	-0.1859%
7	53.65%	0.2363%	0.1958%
8	52.05%	0.1042%	0.0683%
9	52.39%	0.1149%	0.0805%

Now it can be seen that days with a mean performance that was far from 0 in figure 4.1-4.2, also have prediction accuracies far from 50%. Usually the prediction accuracy is compared 50%, since that is what randomly guessing if price will increase or decrease would obtain. Although the neural networks for some days have prediction accuracies larger than 50%, the accuracies are still far from 100%. This result is quite expected, since otherwise the efficient market hypothesis could be rejected.

5 Analysis

It has been shown that the mean performance, if the strategy presented in section 2.5 is used, is significantly different from 0 for days 3-9. There are two unintuitive results, namely those for day 5 and 6. On these days, the mean performance is significantly less than 0, meaning that the value of the portfolio decreases. This, on the other hand, means that if each decision is reversed for these days then the performance might be significantly larger than 0, yielding an increased value. Reversing the decision means that if the previous decision was to invest, then the new decision is to short sale and vice versa. Then this means that if the change factor was C_i for a given stock, then the change factor of the reversed decision would be $2 - C_i$. Doing this reversed decision making for day 5 and 6 yields the results in figures 5.1-5.2. These figures should be compared to figures 4.1-4.2.

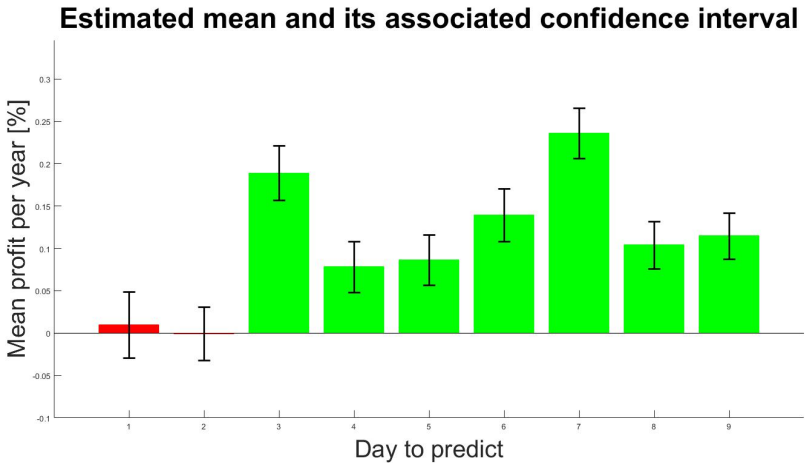

Figure 5.1: Estimation of mean performance and confidence intervals for a single stock, divided into separate days, and with reversed decisions for day 5 and 6.

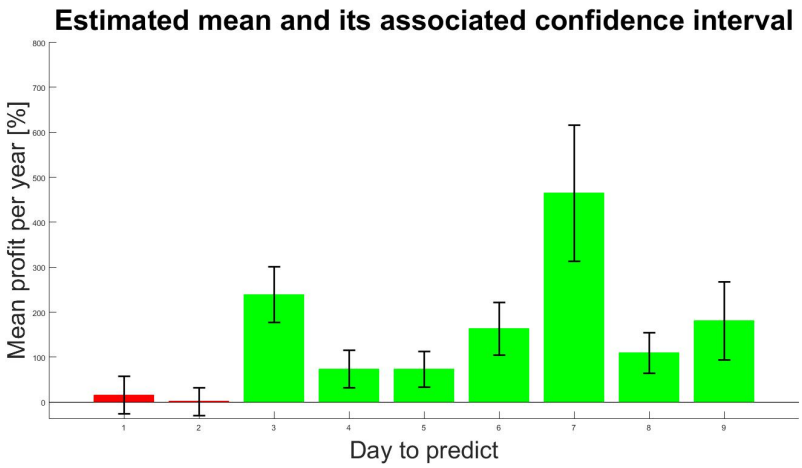

Figure 5.2: Estimation of mean performance and confidence intervals for one year of investments, divided into separate days, and with reversed decisions for day 5 and 6.

These results show that the change of price can be estimated such a strategy can be derived that gives an estimated performance which is significantly larger than 0, for all days except the first two after the crash. The neural networks are not extremely good, however, at guessing if the price will increase or decrease. The best network has a prediction accuracy of 53.92% which is just 3.92 percentage points above randomly guessing. This is expected to be the case, since the analysis is technical and the efficient market hypothesis states that it should not be possible to beat the market in the long run just through technical analysis.

It can be seen that the shape of the figures for the change factors for each day, figures 4.3-4.5 and A.1-A.6, are all very similar. The largest difference is a horizontal shift, where the graph is shifted to the left when the prediction accuracy is high and to the right when the prediction accuracy is low. This result is quite expected since it shows that most of the price changes are quite near 0. It is also expected that the difference between a network that has a large mean profit per stock and a network that has a small mean profit per stock is prediction accuracy.

6 Discussion

In this section assumptions will be questioned, and problems surrounding the method are discussed.

6.1 Assumptions

First of all it is assumed that no transaction costs are present. Transaction costs can be divided mainly into search costs and costs for reaching an agreement. (Byström, 2014) Identifying stocks that decrease drastically and running the analysis can be made by a computer, hence the search costs are close to none. The costs for reaching an agreement is in the purchase of stocks the brokerage fees that are associated with buying the stocks. This can be an fixed cost, which means that if a sufficiently large amount is invested then the relative cost could be very small. (Stockbrokers, 2018)

Secondly, it is assumed that it is possible to short sale stocks at no extra costs. The results would most likely be a bit different without assuming that short sales are possible, but I have not investigated this issue further.

Thirdly stock prices are assumed to never more than double its price in one day. This is quite reasonable shortly after a crash however for one stock and one day in the analyzed data set, the price more than doubled for one day - hence the assumption is not waterproof. Therefore it is important to be extra careful when short sales are used, and all the value of the portfolio should not be invested in a short sale transaction.

6.2 Arithmetic mean vs geometric mean performance

In section 2.8 it is described that arithmetic and geometric means are quite different but still both are widely used, although for different purposes. The arithmetic mean should be used if the same amount is invested into different stocks simultaneously, while geometric mean should be used if one amount is invested sequentially in a number of stocks. In section 2.5 something very similar to a geometric mean has been used, since it has been assumed that the same amount is invested sequentially in different stocks.

The difference between the two measurements is how an investor decides to invest the money in the portfolio into different stocks. In this thesis both strategies have been evaluated and combined. First a sort of geometric mean has been used in order to get an estimation of the performance of the investment strategy. Thereafter an arithmetic

mean has been used in order to estimate mean performance if the investment strategy is used. Also in the analysis of individual stocks an arithmetic mean is used to measure performance.

6.3 Ethical aspects of day trading

It is an usual point of discussion whether day trading is ethical or not. These investments are a part of making the financial market more efficient, and makes it reflect the underlying values in a better way. Still there are various disadvantages of day trading. Usually when someone profits from day trading, someone else loses. It is possible that the profiters of day trading generally are huge companies, while the losers are private small investors.

7 Conclusions

From the results and analysis the only possible conclusion is that a neural network may predict increases and decreases in stock prices shortly after drastic decreases in price, at least better than what chance does. The result is clear, however not necessarily large enough to contradict the efficient market hypothesis - at least not with the specific neural networks that have been constructed in the work of this thesis. Still, the result is clear enough to give a positive conclusion to the problem formulation.

8 Thoughts for further investigations

There are of course numerous ways to deepen the understanding of the financial market using neural networks. Therefore more analyses like the one presented in this thesis most likely will be presented within the next years. There are many further investigations that can be conducted, that are somewhat similar to this one.

First of all the type of network could be varied; different number of nodes and layers, and other neural network models can be used. This can be done by varying the parameters in the model and the exact same problem formulation can be further investigated. Using more nodes and layers would need a longer training period which takes time, on the other hand the network could possibly be more successful.

Secondly other characteristics could be analyzed. For example other time periods, apart from shortly after large decreases in price, can be investigated. The first and most obvious example is shortly after a large increase in price. Another interesting hypothesis to test if a neural network can predict price changes of stocks better than chance on general data.

Thirdly, other underlying assets could be investigated, with a similar approach. Possibly it is also feasible to do better than chance when predicting prices of derivatives and futures.

Apart from these very obvious continuations of the results from this thesis, there are other similar problems that can be investigated. One very interesting application could be to analyze the variance of the prices. An analysis of variance could then be compared to the commonly used ARCH- and GARCH-models. Also the hypothesis that a neural network could predict variances more accurately than statistical methods does not contradict any widely accepted hypothesis, like the one of the efficient markets.

Finally it could be investigated how transaction costs might affect the results presented in this report. One approach would be to find a trading platform and calculate the brokerage fees. Then this information could be used to calculated how large the investments would have to be in order to still have a positive mean performance. For example on the Swedish bank and trading platform Avanza, the brokerage fee can be a fixed cost of 99 Swedish crowns. (Avanza, 2018) The largest arithmetic mean of the performance for one stock is 0.2363% increase. If the arithmetic mean of increase should be at least 0.2%, then this would give that an investment of about 300000 Swedish crowns, or about 35000 USD, is needed to obtain an arithmetic mean of at least 0.2%. This number is of course not a valid estimation, since among other things only stocks on the Stockholm Stock Exchange market can be traded on Avanza. Therefore a more thorough investigation is needed.

9 Reference list

3blue1brown, 2017. *Deep learning.* https://www.youtube.com/watch?v=aircAruvnKk&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi [2018-05-13]

Avanza, 2018. *Så funkar våra courtageklasser.* https://www.avanza.se/konton-lan-prislista/prislista/courtageklasser.html [2018-05-21]

Bishop, C.M., 2006. *Pattern Recognition and Machine Learning*

Blom, G., Enger, J., Englund, G., Grandell, J. & Holst, L., 2005. *Sannolikhetsteori och statistikteori med tillämpningar*

Byström, H., 2014. *Finance - Markets, Instruments and Investments*

Fama, E.F., 1970. *Efficient Capital Markets: A Review of Theory and Empirical Work*

Gündüz, H., Yaslan, Y., & Çataltepe, Z., 2018. *Stock market prediction with deep learning using financial news*

Kaggle, 2018. *S&P500 Stock Data.* https://www.kaggle.com/camnugent/sandp500 [2018-03-30]

Kamijo, K. & Tanigawa, T., 1990. *Stock price pattern recognition: A recurrent neural network approach*

Li, M., & Ma, W., 2010. *Applications of Artificial Neural Networks in Financial Economics: A Survey*

Malkiel, B., 2003. *The Efficient Market Hypothesis and Its Critics*

Malliaris, M.E., 1994. *Modeling the behavior of the S&P 500 index: a neural network approach*

Rogers, S. & Girolami, M., 2016. *A first course in Machine Learning*

Saad, E.W., Prokhorov, D.V., & Wunsch, D.C., 1998. *Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks*

Stockbrokers, 2018. *21 Most Common Online Broker Features & Fees.* https://www.stockbrokers.com/guides/features-fees#stocktradingfees [2018-05-12]

A Figures of predictions for each day

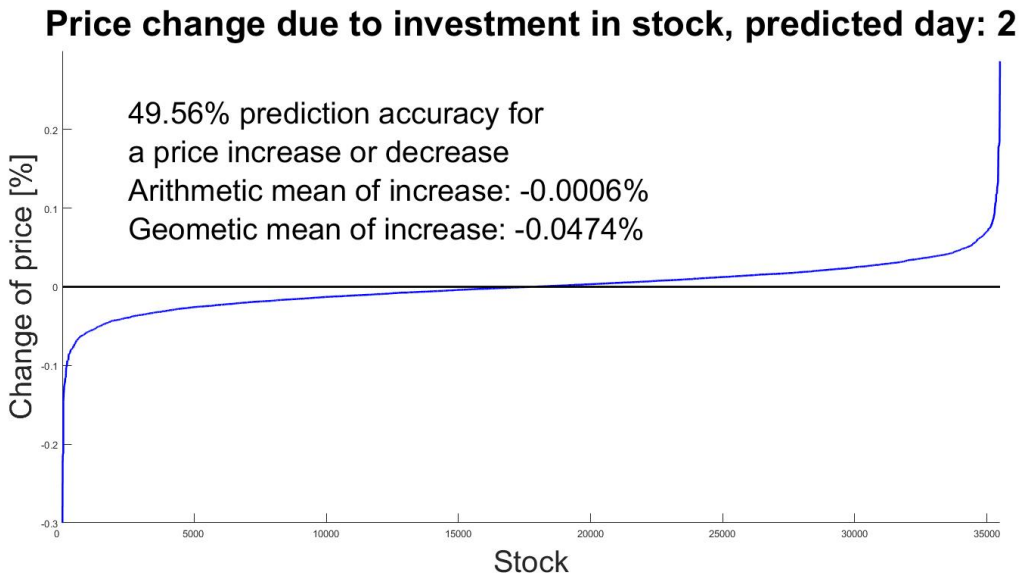

Figure A.1: Change factors given by the investment strategy for each stock.

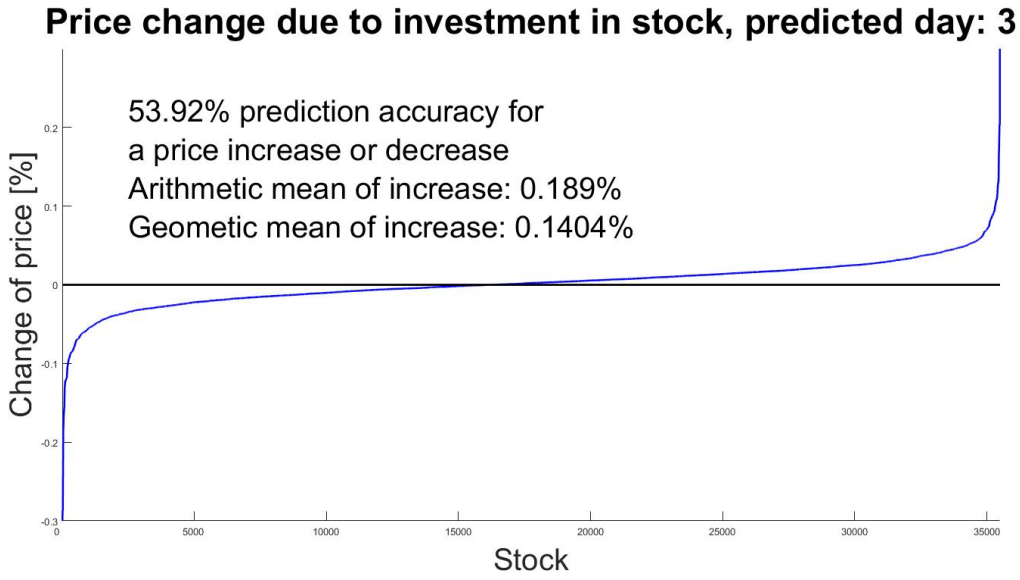

Figure A.2: Change factors given by the investment strategy for each stock.

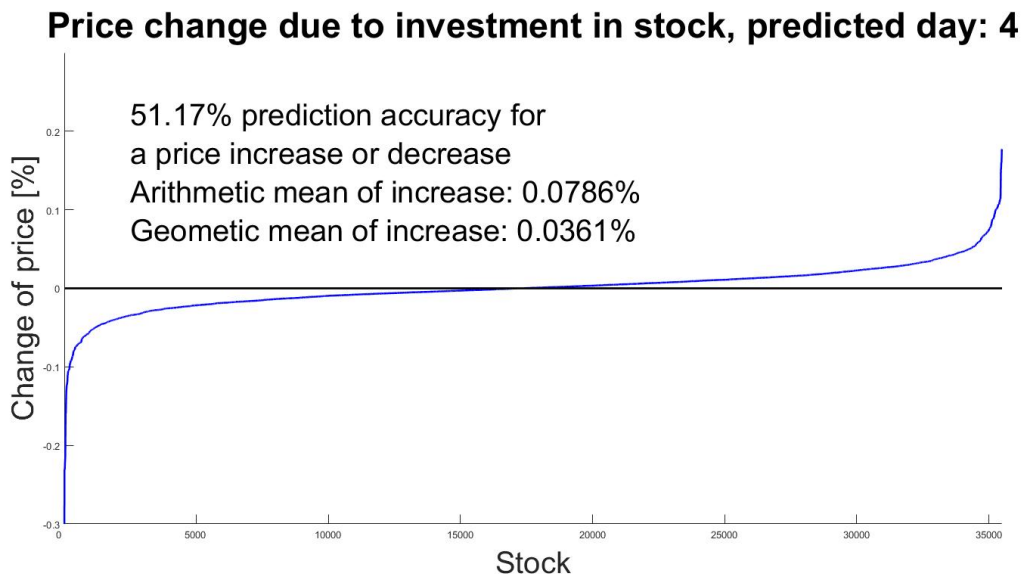

Figure A.3: Change factors given by the investment strategy for each stock.

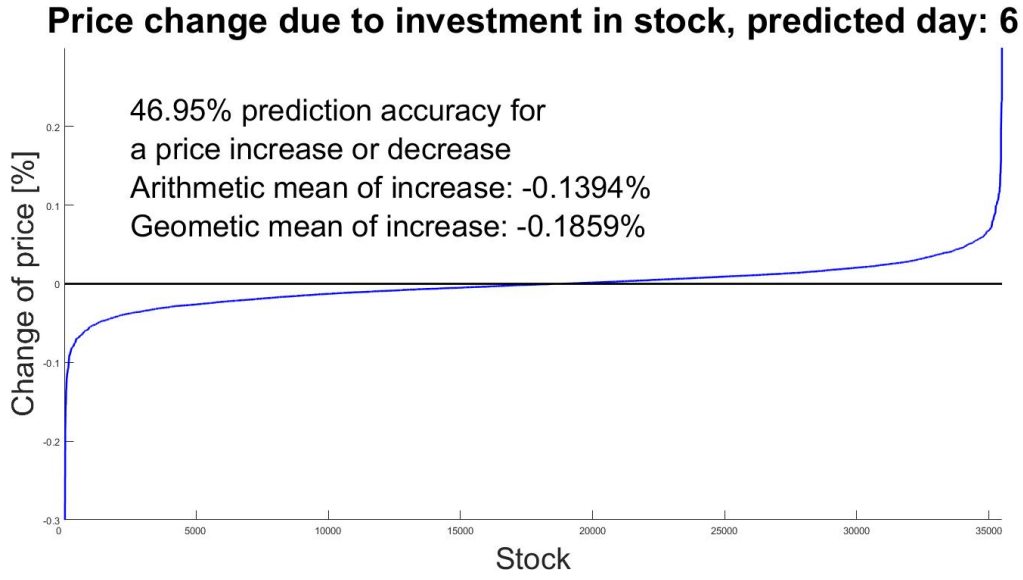

Figure A.4: Change factors given by the investment strategy for each stock.

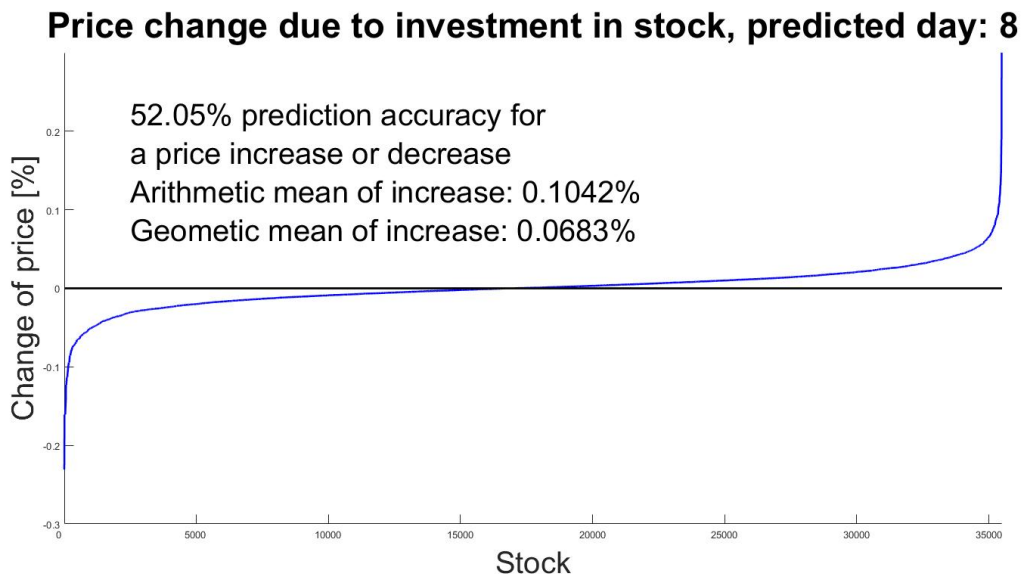

Figure A.5: Change factors given by the investment strategy for each stock.

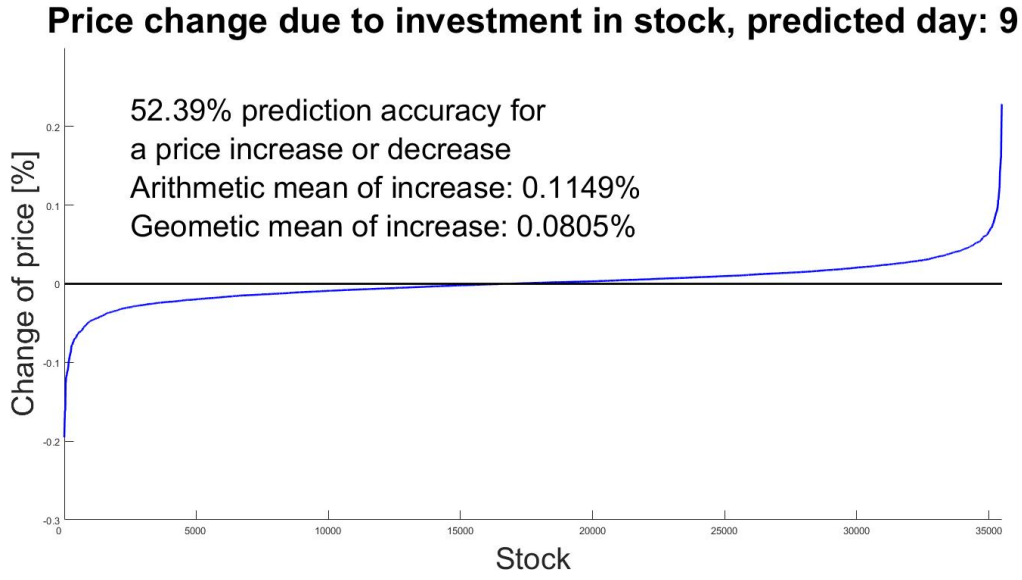

Figure A.6: Change factors given by the investment strategy for each stock.

B Python-code for the neural networks

```
from __future__ import division
import tensorflow as tf
import numpy as np
import math
import pandas
from keras.models import Sequential
from keras.layers import Dense, LSTM, Embedding, Dropout, TimeDistributed
    ↪ , Activation
from keras import losses
from random import shuffle

#Returns data as a list.
def inputs():
    csv = pandas.read_csv(file_path, header = None)
    data = []
    for i in range(len(csv)): data.append(csv[i:i+1].values.tolist()[0])
    return data

#Transfering vector to binary, if increase or decrease.
def to_binary(vec):
    if binary_answer:
        ret = []
        for i in range(len(vec)):
            if vec[i] >= 1: ret.append(1)
            else: ret.append(0)
        return ret
    return vec

#Returns a list of train_data, ans_train_data, test_data, ans_test_data.
def create_batches(data):
    ret = []
    test_size = int(no_data*test_proportion)
    if test_size % batch_size != 0: test_size -= test_size % batch_size
    test_data = data[:test_size]
    training_data = data[test_size:(no_data - no_data % batch_size)]
```

```
    ret.append(np.array([trd[:inp_to_guess] for trd in training_data]).
        ↪ reshape(len(training_data), inp_to_guess,1))
    ret.append(np.array([trd[inp_to_guess] for trd in training_data]).
        ↪ reshape(len(training_data),1))
    ret.append(np.array([ted[:inp_to_guess] for ted in test_data]).reshape
        ↪ (len(test_data),inp_to_guess,1))
    ret.append(np.array([ted[inp_to_guess] for ted in test_data]).reshape(
        ↪ len(test_data),1))
    return ret

#Give result of a test, given predicted and answers.
def test(predicted, answers):
    ret = 1
    val = [1]
    for ak in range(len(predicted)):
        if predicted[ak][0] > 1: ret *= answers[ak][0]
        else: ret *= (2-answers[ak][0])
        val.append(ret)
    return ret, val

print('-----------Start␣of␣file--------------')
#-----------------Config------------------
file_path = 'csv95.csv' #File name of csv.
batch_size = 355 #Size of batches that are analysed together.
no_layers = 1 #Number of hidden layers
inp_to_guess = 7
data = inputs()
input_size = len(data[0])
state_size = [20]*no_layers #Number of nodes in each layer. Can be
    ↪ different sizes for different layers.
test_proportion = 0.1
spread_factor = 2
spread_factor2 = 2
output_size = 1
no_data = len(data)

#Parameters to change!
epochs_of_training = 200
dropout_rate = 0
```

```
activation_hidden = 'relu'
activation_output = 'linear'
loss_function = 'mae'
optimizer_method = 'Adam'
binary_answer = False
shuffle_train_data = True
no_of_rounds = 100

#Runs the same test multiple rounds, and storing the results.
allRes = []
allPreds = []
allAnsws = []
allAllVals = []
results = []
preds = []
answs = []
all_values = []
for i in range(1,no_of_rounds+1):
    print 'Round ' + str(i) + ' started'
    shuffle(data)
#----------------Graph Building-----------
    model = Sequential()
    for lay in range(no_layers):
        model.add(LSTM(state_size[lay], batch_input_shape = (batch_size,
            ↪ inp_to_guess, 1), dropout = dropout_rate, activation =
            ↪ activation_hidden))
    model.add(Dense(1,activation = activation_output))

#-------------------Compiling-------------
    model.compile(loss = loss_function, optimizer = optimizer_method,
        ↪ metrics=['mae'])

#---------------Training--------------------
    training_data, ans_training_data, test_data, ans_test_data =
        ↪ create_batches(data)
    model.fit(training_data, ans_training_data, verbose = 0, batch_size=
        ↪ batch_size, epochs=epochs_of_training, shuffle=
        ↪ shuffle_train_data)
```

```
#--------------Testing-----------------
# score, acc = model.evaluate(test_data, ans_test_data, batch_size =
    ↪ batch_size)
    predicted = model.predict(test_data, batch_size = batch_size)
    score, values = test(predicted, ans_test_data)
    preds.append(predicted)
    answs.append(ans_test_data)
    results.append(score)
    print 'Round', str(i), 'finished. The score was', str(score)
    #all_values.append(values)

print results

resFile = open('res.txt','w')
resFile.write("%s\n" % str(results))
predFile = open('preds.txt','w')
predFile.write("%s\n" % str(preds))
answsFile = open('answs.txt','w')
answsFile.write("%s\n" % str(answs))

print('-----------End of file--------------')
```