UiA University of Agder

Master

Market timing using artificial neural networks

An Application of Machine Learning in Finance

DANIEL DEBBASU MESELU

JEFFREY BENJAMIN CHARLES

SUPERVISOR

Jochen Jungeilges & Valeriy Ivanovich Zakamulin

University of Agder, 2022

Faculty of Business and Law

Department of Economics and Finance

Acknowledgements

As a part of the Master's degree in Business Administration with a specialization in Analytical Finance at the University of Agder, we have written this master's thesis. During the spring semester of 2022, the work has been carried out at the Department of Economics and Finance. We want to show our gratitude and thank our supervisors Professor Jochen Jungeilges and Professor Valeriy Ivanovich Zakamulin for their expertise, guidance and patience. They have guided us through the master's thesis over the last semester, providing us with critical, informative, and, when required, deeply welcomed constructive feedback. We also want to express our appreciation to our family and friends for all of their support over the years.

Abstract

The emergence of artificial neural networks has given us some of the most impressive technological tools. Inspired by the human brain, these networks consist of interconnected artificial neurons that can detect patterns invisible to the human eye. These qualities have caught the attention of investors seeking ways to beat the market. In this thesis, we explore how artificial neural networks can be used to construct an active trading strategy and evaluate the strategy's performance against two benchmark strategies. Two stock indices were used to train neural networks using the lagged return as input to predict the market state. By using the networks' predictions, an active trading strategy was constructed. To evaluate the network-based strategy, we test if the Sharpe ratio differs significantly from the Sharpe ratio of a simple moving average strategy and the buy and hold strategy. Additionally, we estimate the alpha in the capital asset pricing model. The results show that the network strategy performs similar to the benchmark strategies in terms of Sharpe ratio and fails to generate significant alphas. Overall, our results contribute to the previous literature seeking to apply neural networks to finance and should serve as a reminder of the shortcoming of financial data for machine learning and the importance of statistical testing.

Keywords: Artificial neural networks, Forecasting, Active trading

JEL Classification: C45, C53, G17

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The only way to make money in financial markets is to buy low and sell high. With this comes the need to identify the times when the price of a commodity is low or high. Primarily, there are two methods of studying price movements: fundamental- and technical analysis. Fundamental analysis assumes that the stock price sometimes moves away from its natural value. A stock with a price under its natural value is said to be undervalued, and one should buy the stock, while a stock with a price above its natural value is overvalued, and it is time to sell the stock. The technical analysis seeks to forecast the future price movements by studying the historical price of data and discovering recurring patterns in price dynamics (Zakamulin, 2021a).

A well-known hypothesis in finance is the efficient market hypothesis. The hypothesis states that a security always "fully reflects" all available information in the market (Fama, 1970). The weak form efficient market hypothesis states that it is impossible for investors to consistently beat the market through the use of historical data. Despite this, investors attempt to beat the market using historical data of the underlying asset. Constructing effective forecasting models is extremely challenging due to the complex nature of stock market data (Kara et al., 2011).

Machine learning is a branch of Artificial Intelligence and includes different algorithms and techniques. Like traditional technical analysis, machine learning algorithms in finance seek to uncover underlying patterns in price dynamics. Technical analysis is based on the assump-

tion of specific patterns existing in financial prices. Therefore, technical analysis is tailored to exploit these patterns, which the human brain can often detect. The different algorithms used in machine learning have the capability to map continuous non-linear functions using the historical price data. This allows them to discover patterns that are not visible to the human eye (Zakamulin, 2021a).

Within the realm of machine learning algorithms, artificial neural networks (ANNs) are widely used due to their flexibility and scalability, allowing for a wide range of predictive problems to be solved (Cansiz, 2020). Inspired by the human brain, ANNs consist of interconnected units of large numbers of neurons. The neurons receive input signals, process them, and send them to an output signal. An artificial neural network consists of a set of weighted synapses, an adder for summing the input data weighted by the respective synaptic strength, and an activation function limiting the amplitude of the neuron's output (Zhang, 2018).

The capability of artificial neural networks (ANNs) to model and predict financial time series has been shown (Tealab et al., 2017). Return prediction is the most important task underlying the portfolio construction problem that lies at the heart of the investing industry (Israel et al., 2020). Existing literature on applying machine learning for return prediction shows that ANNs have the capability to model market behaviour (Episcopos & Davis, 1996; Yildiz et al., 2008). It is worth mentioning that these studies primarily focus on the machine learning algorithms instead of the results of the trading strategies. Consequently, this leads to the lack of statistical testing of performance measures.

Multiple studies have highlighted the limits and shortcomings of the application of machine learning algorithms to financial data due to the amount of noise, dimensionality and non-stationarity (Kara et al., 2011). This is reflected in a subsequent strand of literature showing that trading strategies built with ANNs are often outperformed by naive financial models like the random walk or the buy and hold strategy (Olson & Mossman, 2003).

The characteristics of financial markets therefore raise the need for a more sophisticated

approach when applying machine learning algorithms. Thus, instead of predicting returns or their sign, we seek to predict the market state. Specifically, we use a neural network to predict if the market will be in a bull or bear state. We use two major indices in the US, namely Standard and Poor's Composite (S&P composite) and Dow Jones Industrial Average (DJI).

Building on the existing literature, we present a trading strategy based on the forecast of our network and provide a statistical justification for our results. To do this, we compare the Sharpe ratios of the network-based strategy to a benchmark passive buy-and-hold (BH) strategy and a trading strategy based on a simple moving average (SMA). Additionally, we estimate the alpha in the capital asset pricing model (CAPM) by regressing the network strategy's excess returns on the market's excess return.

Our results show that neural networks correctly predict the market state with an average accuracy of 77% and 74% for the S&P Composite and DJI, respectively. The trading strategies based on these predictions showed NN to generate a higher Sharpe ratio than the passive BH strategy for the S&P Composite. However, the statistical tests show that NN outperforms the benchmarks in only one period. Furthermore, the NN strategy shows weak performance for the DJI, with Sharpe ratios lower than both benchmarks for all periods tested. For both indices, the NN did not generate any significant alphas.

Similar to the existing literature exploring ANNs in financial markets, our results show that an ANN strategy can give a higher Sharpe ratio and generate positive alpha. The main difference lies in the statistical testing of the results. Therefore, our findings should stress the importance of statistical tests and not be deceived by a favourable realization of a test statistic. The rest of the thesis is organized as follows. In chapter 2, we present a review of the literature regarding the application of neural networks to financial time series, chapter 3 dives into the theory of artificial neural networks, in chapter 4, we present the data used for the study, the methodology used is presented in chapter 5, chapter 6 shows our empirical results, and we elaborate on these in chapter 7. Finally, we present the conclusion of our study in chapter 8.

Chapter 2

Literature review

The usage of a neural network to predict financial markets has been actively researched since the late 1980s (Pan et al., 2005). For example, Yildiz et al. (2008) trained a three-layered neural network using mean square error (MSE) and revised backpropagation algorithms to predict the direction of the Istanbul stock exchange. The results showed that the model forecasted the direction of the respective index with an accuracy of 74.51%. Kara et al. (2011) attempted a similar study to predict the direction of movements in the daily Istanbul stock exchange. A three-layered feedforward artificial neural network was trained using root mean square error (RMSE) to forecast the direction of the price index. The statistical performance of the ANN model was compared to the performance of a model based on a support vector machine (SVM) approach for the same index. Empirical results showed a significant performance of 75.75% in predicting stock price movement with the ANN model, while the performance of the SVM model was 71.52%.

Neural networks also showed good performance in the study conducted by Yildirim et al. (2011) where they aimed to predict the monthly stock yields of 14 different paper companies using a multi-layered ANN. The forecast performance was compared to that of a multiple linear regression model using RMSE and mean absolute percentage error (MAPE). The ANN model correctly predicted stock yields to account for 95% of the data variability. Similarly, Pan et al. (2005) predicts the Australian stock market index using a multi-layer neural network. The results show that the model exhibits up to 80% directional prediction correctness. Addai (2016) compared different techniques such as artificial neural network,

Logit model, Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), and K-Nearest Neighborhood classification (KNN) to find the best predicting model by comparing the accuracy of their forecasts. The results showed that the ANN model performed relatively better than the other models in classifying up and down movements of returns. ANN predicted the direction correctly in 61% of the time considered.

Several authors have also tested the results of neural networks against traditional financial forecast methods. For instance, Olson and Mossman (2003) compared the average directional accuracy of ANNs to the ordinary least square regression and logistic regression for the Canadian stock market returns. The results showed that the ANN outperformed the best regression alternatives. Similarly, Mostafa (2010) conducted an empirical study to forecast the Kuwait stock exchange closing price movements. They used multi-layer perceptron neural networks and generalized regression neural networks to predict the closing movements. In addition, RMSE and mean absolute error (MAE) were used to evaluate the forecasts. The results showed that the neural network models are expected to outperform traditional statistical techniques such as regression and ARIMA in forecasting price movements.

Further, Qi (1999) examines the non-linear predictability of the excess returns. For this, a non-linear neural-network model was used. The author demonstrates that the non-linear NN model gives a better out-of-sample forecast and in-sample fit than the linear regression model using RMSE, MAE, MAPE, Pearson's correlation coefficient and the directional forecast accuracy. The results indicate that neural-network forecasts generate higher profits and lower risk than the buy-and-hold market portfolio. Similarly, Kanas and Yannopoulos (2001) conducted a study to forecast the out-of-sample performance of monthly returns for the two indices, Dow Jones and FTSE All-Share. A non-linear artificial neural network model is used to generate out-of-sample forecasts, and the RMSE of the forecasts were evaluated. The results show that the out-of-sample artificial neural network forecast is significantly more accurate than the linear forecasts.

In addition to the comparison against financial forecast models, studies have been conducted to compare network-based trading strategies against the random walk model. Among these,

Episcopos and Davis (1996) tried predicting returns on the Canadian exchange rates using neural networks and Egarch-M models and compared the forecast ability with the random walk model as the benchmark. In addition, the author used MAE to evaluate the performance of the models. The results showed that neural networks and Egarch-M models outperformed the random walk model. Additionally, Chen et al. (2003) attempted to predict the direction of return on the Taiwan stock exchange. Their model is rooted in the probabilistic neural network (PNN) paradigm, and the statistical performance of the PNN was compared to the generalized methods of moments with the Kalman filter. The empirical results show that the probabilistic neural network model demonstrated a stronger predictive power in terms of accuracy than random walk and generalized methods of moments forecasting models.

Israel et al. (2020) suggests machine learning algorithms should be an addition to existing forecasting techniques rather than a replacement. This adaptation is seen in a study by Wang et al. (2019) where a combination of complex network and ANN techniques were used to predict copper spot price. They first transformed the original time series into a price volatility network (PVN). Afterwards, three different ANNs techniques were used; the back propagation neural network, the radial basis function neural network (RBFNN), and the extreme learning machine (ELM). The results proposed that the PVN-ANN method had a favourable effect on directional and level predictions compared to the traditional ANNs methods. MAPE and RSME were used to evaluate the models. Furthermore, Pany and Ghoshal (2015) uses a local linear wavelet neural network (LLWNN) to predict the electricity price of the Ontario and Indian electricity markets. The performance was compared to the MLR, NN, heuristic, and wavelet network models using MAPE. The results show that the proposed model demonstrates a high degree of efficiency and effectiveness and predicts the price trend better than competing models.

To summarize, ANNs have proved to be a good tool for forecasting financial time series. However, although the results of the studies listed here are promising, their shortcomings are evident. Most of the empirical results are revolved around ANNs and their ability to forecast financial time series and neglect the performance of the trading strategies. Specifically, they do not test for the statistical significance of trading strategies based on their network

forecasts. This motivates the use of statistical tests to assess the performance of ANN-based trading strategies against traditional benchmark strategies.

Chapter 3

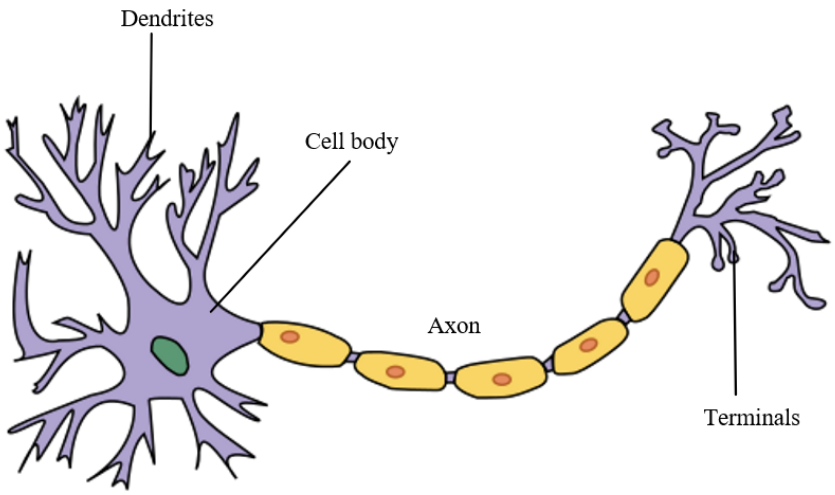
Theory

3.1 Neural networks

Throughout time, nature has inspired many inventions. For example, birds inspired us to fly, and whales inspired us to invent the submarine. Therefore, drawing inspiration from the brain when trying to build artificial intelligence seems logical. This is the idea that started artificial neural networks (ANNs). However, despite being inspired by birds, planes don't flap their wings. Similarly, despite being inspired by the human brain, ANNs differ from their biological counterparts.

Figure 3.1 illustrates simplified anatomy of a biological neuron. A biological neuron is a type of cell found mainly in animal brains. It is composed of a cell body, dendrites and an axon. At the end of the axon lies the terminals, which are the connection to other neurons. Signals in biological neurons are passed on as electric impulses from one neuron to the other through these terminals. When a neuron receives enough signals from other neurons, it fires off its own signal. Individually, this is a rather simple concept, but biological neurons are organized in a deep network of billions of neurons, with each neuron connected to thousands of other neurons. The result is a network that allows for highly complex computations to be performed by the brain.

Figure 3.1: The anatomy of a biological neuron (Wikipedia, 2008)



Artificial adaptation

Artificial neurons were first introduced in a paper by McCulloch and Pitts (1943). This research contained a simplified computational model of how biological neurons might work together to perform complex computations in animal brains. This model would later become known as an artificial neuron. Like the biological neuron, it consisted of one or more binary (on/off) inputs and one binary output. Furthermore, the neuron would activate if more than a certain number of inputs were active. Even with such a simple model, McCulloch and Pitts (1943) showed that it was possible to build a network of artificial neurons to compute any proposition. However, this type of network had no adaptive connection between neurons and was therefore limited to one-way operations. In other words, it could not learn.

The Perceptron

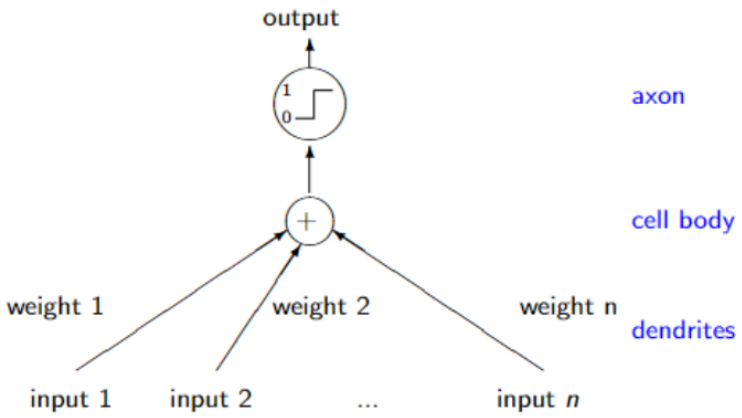
One of the most basic ANN architectures is called the perceptron and was invented by Rosenblatt (1958). It is based on an artificial neuron slightly different from the one presented by McCulloch and Pitts (1943). Instead of the binary on/off values, the inputs and outputs are numbers. Additionally, each connection was now being weighted. This introduced an adaptive connection between neurons, opening up the possibility of learning. This artificial neuron is called a threshold logic unit (TLU). The TLU takes the weighted sum of the inputs, applies a step function, and outputs the result. The step function is shown in (3.1)

$$step(x) = \begin{cases} 0, & \text{if } x < 0 \\ 1, & \text{if } x \geq 0 \end{cases} \tag{3.1}$$

.

If the weighted sum of inputs exceeds a threshold, the perceptron gives the value 1, else it gives 0 (Jungeilges, 2022b). The learning process of a perceptron is achieved by altering the weights between neurons. More specifically, the perceptron is fed one training example at a time to make a prediction (1 or 0). For every wrong prediction, the weights are altered until all training instances are correctly classified. The biological inspiration is reflected in the elements of the perceptron as they all are related to a component of the biological neuron and can be seen in figure 3.2.

Figure 3.2: The anatomy of the perceptron (Jungeilges, 2022b, p.5)



A single TLU can be used for linear binary classification, and early adaptations showed promising results as it was able to recognize letters and spoken words. This led to high expectations by scientists who sought to utilize this technology. However, the fame would be short-lived as a publication of Minsky and Papert (1969) highlighted the limitations of the perceptron and Rosenblatt's algorithm. In particular, they showed that the perceptron was limited to learning linear boundaries. For instance, it could not solve the "exclusive-OR"[1] classification problem. Minsky and Papert (1969) suggested introducing layers of intercon-

[1]A logical operator that results in a positive outcome if exactly one (but not both) of two conditions is positive

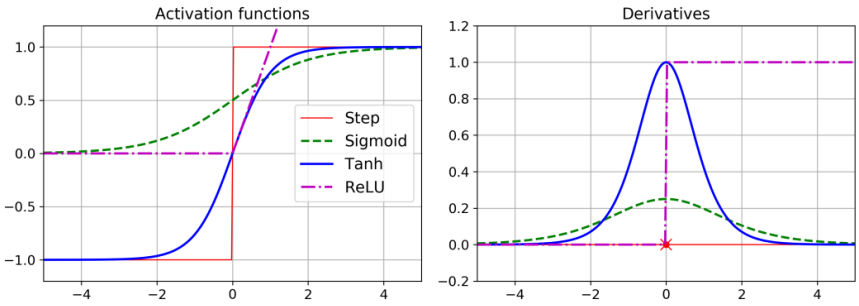
nected neurons and suggested that this construction should be able to learn more than linear boundaries. This was the start of the multi-layer perceptron (MLP). An MLP consists of one input layer, one or more layers of TLUs called the hidden layer, and one final layer of TLUs called the output layer.

Activation function

The introduction of MLPs meant more neurons and more connections. However, there was no efficient way for MLPs to learn at the time, and the development in the direction of neural networks stopped. This would last until 1986 when Rumelhart et al. (1986) published an important paper for the future of neural networks. In their paper Rumelhart et al. (1986) introduced the backpropagation algorithm to train MLPs. In short, backpropagation uses gradient descent to compute the gradient of the MLP's error with respect to every model parameter. A more detailed explanation of gradient descent and the backpropagation algorithm is given in chapter 5. For this algorithm to work, a crucial change had to be made to the architecture of the MLP. The step function in the TLUs of MLPs contained only flat segments, prohibiting gradient decent to move in the direction local minima. To overcome this problem, the authors introduced the logistic function, $\sigma(x) = \frac{1}{1+e^{-x}}$, also called the sigmoidal function. This change was essential as this gave the algorithm a derivative to work with as the derivative of the logistic function is non-zero and defined for all values. The BP algorithm also works well with other activation functions in addition to the logistic function.

Two other popular functions are the hyperbolic tangent function $tanh(x) = 2\sigma(2x) - 1$, and the Rectified Linear Unit Function: $ReLU(x) = max(0, x)$. Similar to the logistic function, the tanh function is continuous and differentiable. The difference between the tanh and the logistic function is that the tanh function outputs values ranging from -1 to 1, while the logistic function outputs values from 0 to 1. In contrast, the ReLU function is not differentiable at $x = 0$ and its derivative is equal to 0 for $x < 0$. However, the practical application of the ReLU function has shown to work very well for ANNs. A visual presentation of the activation functions and their derivatives can be seen in figure 3.3.

Figure 3.3: Activation functions and their derivatives (Géron, 2019, p.289)



But why do we need these activation functions? The downfall of the perceptron can mainly be explained by its inability to learn boundaries other than linear ones. The suggestion of Minsky and Papert (1969) to stack layers of interconnected perceptrons did not change this problem. The reason is that stacking several linear transformations results in a linear transformation. These activation functions all carry non-linear properties, and stacking many layers of such non-linear functions has left us with some of the most amazing products. For instance, voice recognition like Apple's Siri, Google images, and video recommendations on YouTube show some of the capabilities of ANNs. These inventions represent some most impressive technological advances in the last two decades. Therefore it may not be far-fetched to think ANNs have the capability to be used for financial data. However, as we will see, the financial world brings a turbulent environment that ANNs are not accustomed to.

3.2 Shortcomings of financial data in machine learning

The conditions that have allowed ANNs and other machine learning algorithms to perform well ultimately narrow down to two key factors. Firstly, they all have huge amounts of data to work with. Big data sets allow machine learning algorithms to perform well, giving room for a flexible and rich parameterization. Secondly, these environments have a high "signal to noise" ratio. The signal-to-noise ratio describes how much predictability there is in a system. Some systems have clear patterns with a high signal-to-noise ratio and are clearly predictable. Others are characterized by randomness and are difficult to predict, thus having a low signal-to-noise ratio (Israel et al., 2020).

Unfortunately, financial data lack these properties that machine learning benefits from the

most. Firstly, financial data is small compared to other research fields. The notion of "big data" in finance lies in predictors. However, for a machine learning algorithm to learn, it needs sufficient amounts of independent observations. Time is the limiting factor for financial data, as the only way to get more observations is to simply let time pass by. Most fields where machine learning has shown success can easily generate mode data as they see fit (take more pictures for the Google images algorithm, record more videos for YouTube, etc.) This problem depends to a degree on the frequency we are counting our data. The frequency should be based on the frequency the investor will be able to re-balance without acquiring too high transaction costs. A monthly frequency is a sensible frequency for the average investor, as anything more frequent quickly increases transaction costs and leads to a bigger gap between what a model predicts and what can be implemented in practice (Israel et al., 2020).

Secondly, financial data suffer from a low signal-to-noise ratio. Financial markets are difficult to predict, partly due to the growing nature of the economy. The low signal-to-noise ratio in financial data results from economic forces like profit maximization and its competitive nature. Investors who carry information that consistently allows them to predict price movements will exercise upon this information. Consequently, this will push prices up, which will happen until the information is exhausted, and the price will eventually adjust to their predictions. This leaves very little predictability on the table. This is the foundation for the efficient market hypothesis (Israel et al., 2020).

When forecasting financial time series, one usually tries to predict the returns or signs of return, that is if one should expect positive or negative returns. However, in light of what has been discussed in the previous paragraphs, high signal-to-noise ratio levels should be prioritized for machine learning algorithms. Therefore, in efforts to increase the signal-to-noise ratio in our data set, a reasonable approach may be not to predict the returns or their sign but rather to look at the market state. That is if the market is on the rise or fall, denoted by a general increase or decrease in prices. A market where prices are on the rise is referred to as a bull market, whereas a market where prices are declining is referred to as a bear market (Pagan & Sossounov, 2003).

3.3 Efficient market hypothesis

A market where the price of a security always "fully reflects" all available information is said to be efficient (Fama, 1970). Furthermore, Fama (1970) divides the efficient market hypothesis into three subcategories. Namely, weak form, semi-strong form, and strong form market efficiency. In weak form efficient markets, the available information is set to the historical prices. Semi-strong efficient markets assume the price of a security is adjusted efficiently in regard to public information. Finally, the strong form efficient markets, where prices in addition to reflecting historical prices and publicly available information, also reflect all private information. According to the weak form efficient market hypothesis, it is not possible for investors to consistently beat the market through the use of historical data of the underlying security. Therefore, the efficient market hypothesis motivates using a passive buy-and-hold strategy. In a buy-and-hold strategy, an investor will buy a security and hold it for a long period of time without concerns for short-term price fluctuations.

Chapter 4

Data

4.1 Description of data

This thesis uses the monthly returns on the stock market index Standard and Poor's Composite index (S&P Composite), Dow Jones industrial index and the risk-free rate of return. Robert Shiller introduced this S&P Composite index to extend the returns on the S&P Composite index back to 1860. The raw data set used for this study was gathered from two sources. William Schwert [1] provides the first part of the returns from January 1860 to December 1925. The second part of index returns from January 1926 to December 2018 are retrieved from the Center for Research in Security Prices (CRSP)[2]. The returns on the Dow Jones industrial index are provided by Dow Jones Indices LLC[3] and cover the period from January 1897 to December 2020. The total return in both indices includes dividends. The risk-free rate of return from January 1920 to December 2020 is the Treasury bill-rate. The risk-free rate prior to 1920 is constructed using commercial paper rates from New York as suggested by Welch and Goyal (2008).

To avoid arbitrary split-point, we divide the testing and training sample into 3 different split points (periods). By doing this, we can generalize the results. For S&P Composite, the training sample is divided depending on the testing sample. For each index, a time window is held out for testing the predictions of the network, while the remaining sample is used

[1] https://www.billschwert.com/gws data.htm
[2] http://www.hec.unil.ch/agoyal/
[3] https://www.spglobal.com/en/

for training. This time window covers approximately 25% of the data. A training/testing split of 70/30 and 80/20 is suggested by most practitioners in machine learning (Dixon et al., 2020; Géron, 2019). The model needs a sufficient amount of data to learn while also having enough data to generalize the predictions. Since data is scarce in this thesis, we find a 75/25 split to be a satisfactory middle ground. Thus, for S&P Composite, 1951-1980, 1980-2010 and 1990-2018 are used for testing, while 1951-1980, 1980-2010 and 1990-2020 are used for DJI. The time periods will hereafter be referred to as period 1, period 2 and period 3, respectively. The periods start on January 1^{st} and end on December 31^{st}.

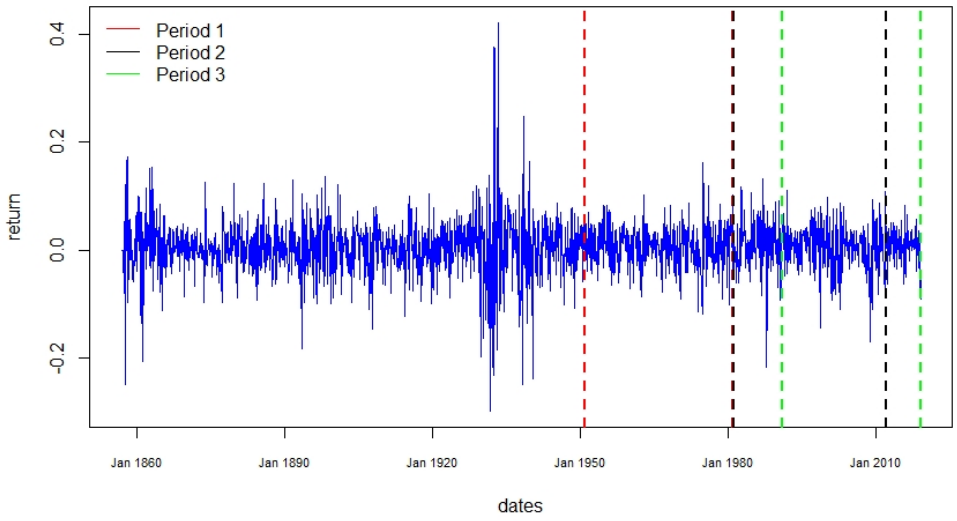


Figure 4.1: S&P Composite historical returns sample period and frequency

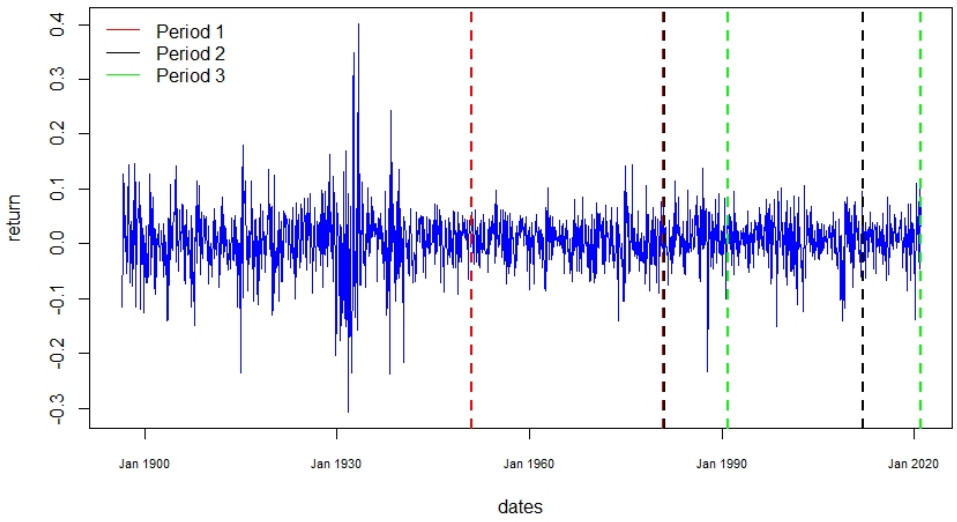


Figure 4.2: DJI historical returns sample period and frequency

4.2 Descriptive statistics

The skewness of S&P Composite is positive for all training periods and negative for all testing periods. In contrast, the DJI shows negative skewness for all periods except for the second training period. The conditions for a normal distribution call for a skewness equal to zero. Therefore, a positive skewness indicates a skewed distribution to the right, while a negative skewness indicates a skewed distribution to the left. A positive skewness could be an indicator of more small losses and fewer bigger gains in the returns. On the other hand, a negative skewness could indicate that we are dealing with longer or fatter distributions.

A normal distribution has a kurtosis of exactly 3. We observe a kurtosis larger than 3 for all training and testing periods. A distribution with a kurtosis larger than 3 is referred to as leptokurtic. Such distributions are characterized to have a higher probability of extreme realizations or heavier tails. Knowing the distribution is leptokurtic, we can expect fluctuations resulting in months with greater potential for very high or very low returns relative to the normal distribution (Brooks, 2019).

Table 4.1: Descriptive statistics of S&P Composite and DJI

S&P	Periods	Min	Max	Mean	Std	Skewness	Kurtosis
	1860-1950 & 1981-2018	-0.299	0.422	0.005	0.052	0.200	11.528
Training	1860-1980 & 2011-2018	-0.299	0.422	0.005	0.051	0.311	12.068
	1860-1990	-0.299	0.422	0.005	0.051	0.264	11.598
	1951-1980	-0.119	0.163	0.006	0.040	-0.070	3.769
Testing	1981-2010	-0.218	0.132	0.007	0.045	-0.668	5.187
	1991-2018	-0.169	0.112	0.007	0.041	-0.642	4.399
DJI	Periods	Min	Max	Mean	Std	Skewness	Kurtosis
	1897-1950 & 1981-2020	-0.307	0.402	0.006	0.057	-0.092	8.716
Training	1897-1980 & 2011-2020	-0.307	0.402	0.005	0.056	0.030	9.079
	1897-1990	-0.307	0.402	0.005	0.056	-0.006	8.921
	1951-1980	-0.140	0.144	0.005	0.039	-0.084	3.763
Testing	1981-2010	-0.232	0.138	0.008	0.044	-0.698	5.697
	1991-2020	-0.151	0.118	0.008	0.042	-0.565	4.337

4.3 Autocorrelation in returns

The degree of correlation between two consecutive time intervals is referred to as autocorrelation. It assesses the relationship between the lagged version of a variable's value and the original version in a time series (Brockwell & Lindner, 2012). Studying the linear relationship between input and output might be sufficient if the returns are autocorrelated. An estimator of the autocorrelation in returns a time series was proposed by Ljung and Box (1978) with a null hypothesis stating all L autocorrelations to be equal to zero simultaneously. The alternative hypothesis states that at least one lag differs from zero.

$$H_0 : \rho_1 = \rho_2 = \rho_3 ... = \rho_L = 0, \qquad HH_1 : \text{At least one} \neq \tag{4.1}$$

The test estimator can be written as follow:

$$\tilde{Q} = T(T+2) \sum_{T+1)}^{L} \frac{\hat{\rho}_\tau^2}{T-\tau} \ \chi_L^2 \tag{4.2}$$

where T is the sample size, τ is the number of lags, and $\hat{\rho}_\tau$ is the estimated correlation coefficient. When $T \to \infty$, the estimator has a χ^2 density with L degrees of freedom (Jungeilges, 2022a, p.7). The test was conducted using τ=12 for both indices. The results showed a test statistic of 36.98 and 30.28 for the S$P Composite and DJI, respectively, with corresponding p-values of 0.0002 and 0.003. In light of the null-hypothesis, we have a highly unlikely observation. Therefore, the null hypothesis is rejected, and one can expect autocorrelation in at least one of the lags. The correlograms for the S&P Composite and DJI shown in figure 4.3 and figure 4.4 indicate that the S&P Composite has significant autocorrelation at lat 1 and 5, while the DJI has a significant autocorrelation at lag 5.

Figure 4.3: Correlogram for the S&P Composite

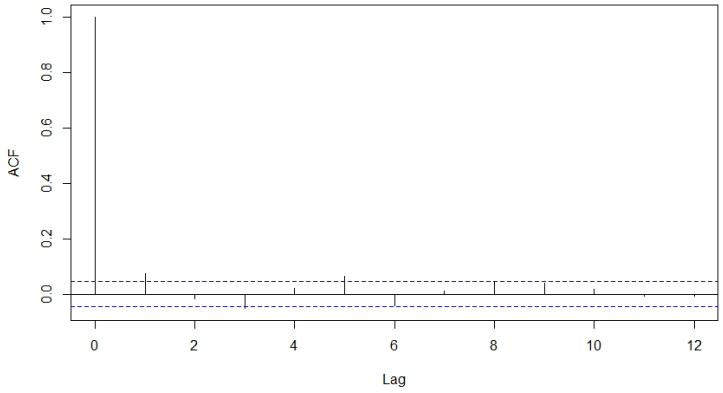
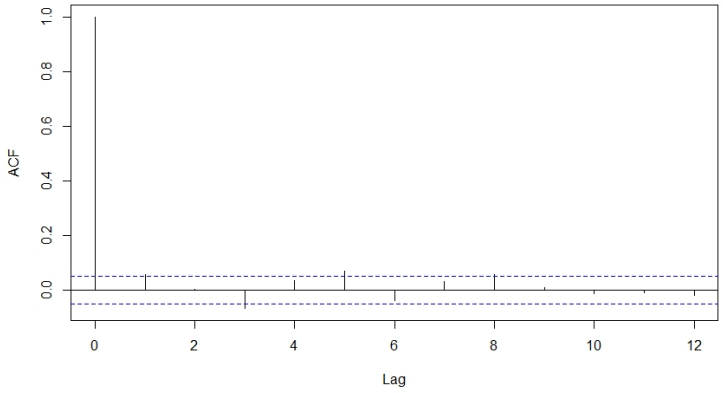


Figure 4.4: Correlogram for DJI



4.4 Labeling Bull and Bear Markets

There is no common agreement on the dating of bull and bear markets (Zakamulin, 2017a). Two distinct groups disagree regarding the timing of bull and bear markets. One group argues that there needs to be a substantial increase (decrease) in the price for it to qualify as a bull (bear) market. The other group argues that the stock market price needs to increase (decrease) for a substantial amount of time to qualify as a bull (bear) market. In this study, we implement the algorithm proposed by Bry and Boschan (1971) to transform our data from monthly returns to bull and bear markets by using the R package provided by (Zakamulin, 2017a). The algorithm requires prices as input, so the transformation is done in two steps. First, we calculate the historical prices of the indices using the monthly returns by calculating the cumulative product of the returns. The algorithm has two major stages: establishing the initial turning points in the price (P) and censoring operations. The turning points in P are chosen by first choosing a time window of length τ_{window} on either side of the date to identify a peak (trough) in P higher (lower) than other points in the window. Second, we select the highest of multiple peaks and lowest of multiple troughs to enforce the alternation of turning points.

The censoring operations require the removal of peaks and troughs in the first and last months of τ_{censor}, eliminating cycles that last less than τ_{cycle} months, and eliminating phases that last less than τ_{phase} months with the exception of instances where the relative change in P over a month surpasses the threshold θ. The censoring operations are usually repeated many times before satisfying all constraints. In our implementation of the algorithm, the following parameters are used:

$$\tau_{window} = 8,\ \tau_{censor} = 6,\ \tau_{cycle} = 16,\ \tau_{phase} = 4,\ \theta = 20\% \quad .$$

The algorithm's output gives us a vector containing TRUE if the market is in a bull state and FALSE in a bear state. To fit with our classification problem, we transform the signals to BULL and BEAR such that

$$Y = \begin{cases} BULL, & \text{if Bull} = TRUE \\ BEAR, & \text{if Bull} = FALSE \end{cases} \tag{4.3}$$

Table 4.2: Bull and Bear descriptive statistics

S&P Composite 1860-2018	Bull	Bear
Number of phases	53	53
Minimum duration	2	1
Average duration	24	12
Median duration	23	12
Maximum duration	74	44
Minimum amplitude	11	-4
Average amplitude	62	-24
Median amplitude	51	-21
Maximum amplitude	318	-82

DJI 1897-2020	Bull	Bear
Number of phases	39	38
Minimum duration	7	3
Average duration	27	13
Median duration	23	9
Maximum duration	73	34
Minimum amplitude	8	-5
Average amplitude	73	-25
Median amplitude	64	-21
Maximum amplitude	338	-89

The table reports the number of bull and bear markets, their duration, and amplitude for S&P Composite and DJI for the whole sample. The duration of market states is measured in the number of months and the amplitude in measured in percentages.

Figure 4.5: Bull and Bear States in the S&P Composite Index

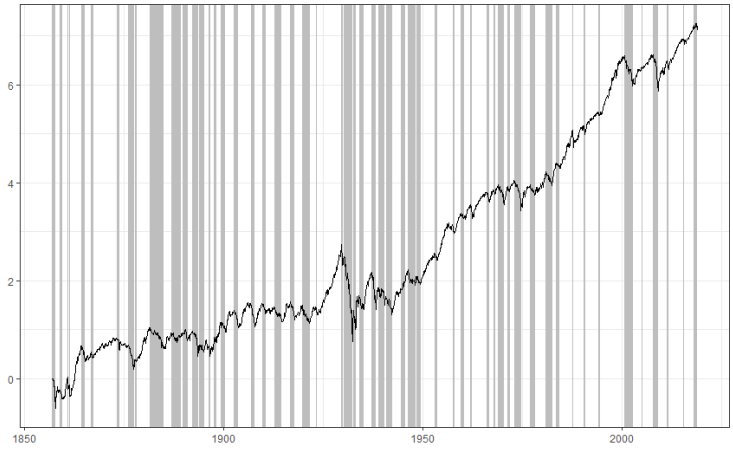


Figure 4.6: Bull and Bear States in the DJI

The plots visualize the distinct bull and bear markets as shown in table 4.2. The white lines indicate distinct bull markets, while the grey lines indicate distinct bear markets. Figure 4.1 shows the market states for the S&P Composite, while figure 4.2 shows the market states for DJI.

4.4.1 Results for S&P Composite

Table 4.2 illustrates that the S&P Composite has 53 phases of bull states and 53 phases of Bear states. This means that from 1860 to 2020, the market experience equal number of bull and bear states. The duration of states is given in the frequency of months. This means that the minimum duration of a bull state is two months, while the minimum duration of a bear state is one month. The average duration of a bull state is 24 months, while the average duration of a bear state is 12 months, meaning that one can expect bull states to last longer than bear states. The maximum duration of a bull state is 74 months and is 30 months longer than the longest bear state of 44 months. The amplitude is given in percentage and tells us something about the magnitude of the bull/bear state. The average amplitude of a bull state is 62%, meaning the stock market index on average increases by 62% over a bull market. Meanwhile, the average amplitude of a bear state is -24%, meaning the stock market index on average decreases by 24% over a bear market. This means that the bull state is much stronger than the bear state in terms of fluctuations. Again, we experience that the average amplitude of a bull state is larger than the amplitude of a bear state. The maximum amplitude of the bull state is 318%, and the largest amplitude of a bear state is -82%.

4.4.2 Results for DJI

Table 4.2 describes the different duration and amplitudes of the bear and bull states of DJI. From 1897 to 2020, we experience 39 phases of bull states and 38 phases of bear states. The average duration of a bull state is 27 months, equal to the S&P Composite. The average duration of the bear state is 13 months, which is a bit below the S&P Composite average. The maximum duration of a bull state is 73, while the maximum duration of a bear state is 34 months. This means that the maximum bear state duration of DJI was 10 months shorter than of the S&P Composite. The minimum duration of a bull state is seven months, while the minimum duration of a bear state is three months. The average amplitude of a bull state is 73%, while the average amplitude of a bear state is -25%, similar to the average amplitude of S&P Composite. The maximum amplitude of a bull state is 338% and -89% of a bear state. The minimum amplitude of a bull state is 8% and -5% of a bear state.

Chapter 5

Methods

5.1 Model estimation

5.1.1 Entropy and Cross entropy

Entropy measures the information content or uncertainty of a probability distribution. Let E_i stands for an event, and p_i denotes the probability of the event E_i to occur. If there are n events $E_1 \ldots .., E_n$ with probabilities $p_1, \ldots . p_n$ adding up to 1, the occurrence of events with a smaller probability will give us more information since they are less expected. Therefore, it is reasonable that a function of information h should be a decreasing function of the probability, p_i (Hanusch & Pyka, 2007). The function proposed by Shannon (1948) expresses information $h(p_i)$ through a logarithmic function. $h(p_i) = log_2(\frac{1}{p_i})$. Because of the properties of the logarithmic function, this function decreases from infinity to 0 for p_i ranging from 0 to 1. This reflects the thought process behind lower probabilities giving higher information. From the n number of information values $h(p_i)$, the expected information content of a probability distribution (entropy) is derived by weighing the information values $h(p_i)$ by their respective probabilities.

$$H = -\sum_{i=1}^{N} p_i log_2(\frac{1}{p_i}) \tag{5.1}$$

where H is the entropy measured in bits. More interestingly, we can measure the average number of bits we need to explain data coming from a distribution p when we use model q by using the cross-entropy between the two. The intuition is, given an underlying probability distribution p, and an estimate of the target distribution q, the cross-entropy of Q from

P is the number of additional bits we need to represent an event using Q instead of P. This cross-entropy can be formally stated as $H(P, Q)$ where H is the cross-entropy, P is the underlying distribution, and Q is the estimate of the target distribution. Thus, $P \in (y, 1-y)$ and $Q \in (\hat{y}, 1-\hat{y})$ we can calculate the cross-entropy as

$$H(P, Q) = -\sum P_i log Q_i = -ylog\hat{y} - (1-y)log(1-\hat{y}). \tag{5.2}$$

Cross entropy is widely used as a loss function in the optimization of classification models, mainly due to the cross-entropy leading to faster training and improved generalization for classification problems as opposed to the mean square error (Bishop & Nasrabadi, 2006, p.235).

5.1.2 Gradient descent

Gradient descent is a common algorithm with the ability to find optimal solutions for various problems (Géron, 2019). The general idea behind Gradient Descent is to change parameters iteratively to minimize a cost function. The key to minimizing a cost function is to travel in the direction of descending gradient. We reach a minimum once the gradient is equal to zero. Intuitively, we start by randomly initializing the parameter vector containing the weights and gradually improve it by taking small steps, each time trying to reduce the cost function (in our case, the cross-entropy loss), until the algorithm converges to a minimum. The size of the steps we take when undergoing Gradient Descent is an important parameter. This is determined by the learning rate. Choosing a sensible learning rate is essential for us to reach a minimum within a given time frame. A learning rate that is too small will require more iterations before converging, which can be time-consuming. In contrast, a learning rate that is too big may lead to skipping the minimum altogether and end up even higher than we started. The biggest shortcoming of the gradient descent algorithm is the existence of local minima. This can cause the algorithm to stop before reaching the global minimum. This happens when the learning rate is set too small. (Géron, 2019, p.121).

5.1.3 Activation function

The goal of activation in the hidden layers of a neural network is to introduce non-linearity to the model. In instances where the activation functions are linear, the transformation is just a linear regression, and the number of hidden layers and the neurons within is irrelevant (Dixon et al., 2020, p. 114). The reason is that a chain of linear transformations results in nothing but another linear transformation. Consequently, even the deepest networks act as a single layer and will be incapable of solving complex problems (Géron, 2019). A commonly used activation function for neural networks is the logistic activation function, the sigmoid function. The sigmoid function is given by

$$F(x) = \frac{1}{1 + e^{-x}}. \tag{5.3}$$

For a binary classification task, a single output neuron carrying the sigmoid activation function is sufficient. The output will be a number between 0 and 1 which can be interpreted as the estimated probability of our positive class (Bull). The final output of our model will contain estimated probabilities between 0 and 1 for the positive class. Thus, we set a threshold of 0.5 for the estimated probability to be of class 1. By doing so, in light of our problem, we can label each estimated value as belonging to class 1 (bull) or class 0 (bear).

5.1.4 Back-propagation

Although neural networks have the power to model complicated, nonlinear functions, the weights have to be learned through trial and error. ANNs do this through something called backpropagation. Backpropagation is a form of stochastic gradient descent. The stochastic nature of the gradient stems from the random initiation of weights. Descending gradient refers to the reduction of errors from each node. Intuitively, this means that we have a set of random weights at the start of our training, which is gradually tuned to give fewer errors. With the sample from the training set, the network first assigns different weights and runs through the network until we get an output. The data used is labelled in pairs, so we already know the target output. By comparing the output from the network with the target output, we can calculate the error $e_i = \hat{y}_i - y_i$. When we have obtained the output error, we can

propagate the error backwards from the last layer and through the network, hence the name backpropagation. Specifically, we calculate how a neuron contributed to the error and adjust the associated weights accordingly. The contribution is measured by how steep the gradient of the error is. This is calculated by finding the partial derivative of the error function with respect to a weight w_{ij} (Aamodt, 2015).

A simple network is shown in figure 5.1. The network has an input layer with two features, $x1$ and $x2$, a hidden layer with two neurons, $h1$ and $h2$ and one layer with a single output feature $\hat{p}_i$. The colour schemes represent the information that the respective neuron receives, and the weights are labelled accordingly. The superscripts on the weights represent the numbers of the layers the weights are connecting. The subscripts represent which neurons are connected together. For instance, w_{11}^1 is a connection from the input layer to the hidden layer between neuron 1 in the input layer to neuron 1 in the hidden layer.

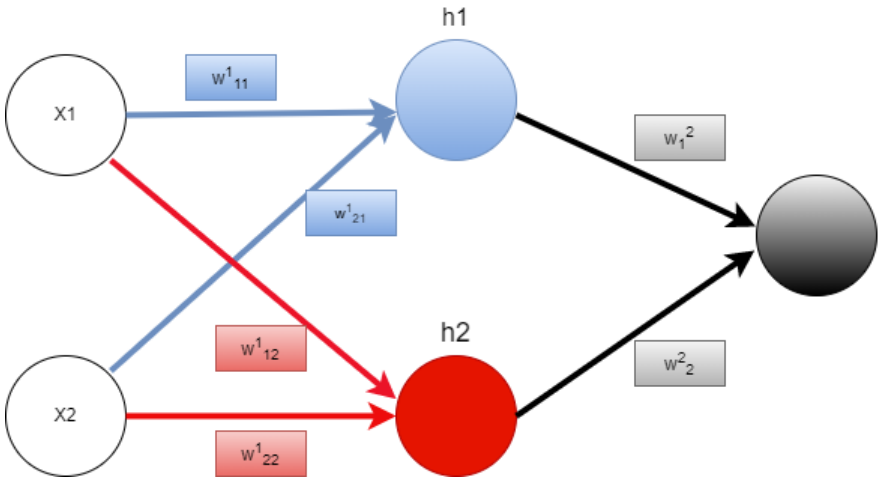


Figure 5.1: Example of network architecture

The current value of the information a neuron receives from the previous layer can be calculated as the sum of the weighted input from the previous layer. Specifically, for the neurons in the hidden layer, we can represent them as the equations in (5.4) and (5.5). This is equivalent to the blue and red connections in figure 5.1 respectively. After applying the sigmoid activation function, the value of the neuron can simply be written as (5.6) and (5.7). Similarly, for the output layer, (5.8) represents the information from the hidden layer, and (5.9) represents the estimated probability after the application of the sigmoid function, $\hat{p}_i$. The loss function used to update the weights is shown in (5.10). The estimated probability

$\hat{p_i}$ is a product of weights and input and will therefore change if we alter the weights.

$$Z_1^1 = w_{11}^1 x_1 + w_{21}^1 x_2 + b_1^2 \tag{5.4}$$

$$Z_2^1 = w_{12}^1 x_1 + w_{22}^1 x_2 + b_2^2 \tag{5.5}$$

$$h_1 = \sigma(Z_1^1) \tag{5.6}$$

$$h_2 = \sigma(Z_2^1) \tag{5.7}$$

$$Z^2 = w_1^2 h_1 + w_2^2 h_2 + b^2 \tag{5.8}$$

$$\hat{p_i} = \sigma(Z^2) \tag{5.9}$$

$$L = -\hat{y} log \hat{y} - (1 - \hat{y}) log(1 - \hat{y}) \tag{5.10}$$

Recall the activation function we are using, the sigmoid function, as presented in section 5.3.1. The partial derivative of the sigmoid function has properties used in backpropagation. Specifically, if we let

$$P = \sigma(x) = \frac{1}{1 + e^{-x}} \tag{5.11}$$

be the value of a neuron, the partial derivative of the neuron with respect to x is given as

$$\frac{\partial P}{\partial x} = (1 + e^{-x})^{-2} e^{-x} = \frac{1}{1 + e^{-x}} * \frac{e^{-x}}{1 + e^{-x}} = P(1 - P) \tag{5.12}$$

As mentioned, updating the weights works backwards from the last layer in the network. We demonstrate the backpropagation by taking the partial derivative of the error function with respect to w_1^2. That is the weight between $h1$ and $\hat{p_i}$. So, the question to be answered is what effect a change in w_1^2 has on the error function. Formally, this equation is given by

$$\frac{\partial L}{\partial w_1^2} \tag{5.13}$$

The error is the sum of all the errors derived from all the outputs of the network and their respective targets.

$$\frac{\partial L}{\partial w_1^2} = \sum_{i=1}^{N} \frac{\partial L_i}{\partial w_1^2} \tag{5.14}$$

We can calculate the partial derivative of the ith error with respect to w_1^2 using the chain rule. Intuitively, from (5.8), we can see that a change in w_1^2 leads to a change in $\hat{p}_i$. Consequently, a change in $\hat{p}_i$ will lead to a change in the error. As we can see, a change in the w_1^2 changes the error, but it happens over several steps. Thus, we can apply the chain rule and multiply all the small changes in order to get the final change on the error. The equation for the application of the chain rule is given in (5.15)

$$\frac{\partial L_i}{\partial w_1^2} = \frac{\partial Z^2}{\partial w_1^2} * \frac{\partial \hat{p}_i}{\partial Z^2} * \frac{\partial L_i}{\partial \hat{p}_i}. \tag{5.15}$$

From (5.8) we can see that the partial derivative of Z^2 with respect to w_1^2 gives us $h1$. Furthermore, we apply the logic behind the derivative of the sigmoid function showed in (5.11) and (5.12) to solve the partial derivative of $\hat{p}_i$ with respect to Z^2 giving us $\hat{p}_i(1-\hat{p}_i)$. And finally, the partial derivative of L_i with respect to $\hat{p}_i$ gives us $(\hat{p}_i - y_i)$

$$\frac{\partial Z^2}{\partial w_1^2} = h_1, \frac{\partial \hat{p}_i}{\partial Z^2} = \hat{p}_i(1-\hat{p}_i), \frac{\partial L_i}{\partial \hat{p}_i} = (\hat{p}_i - y_i). \tag{5.16}$$

We can cluster the last two terms together under and call it δ such that the partial derivative of the ith error function with respect to w_1^2 is

$$\frac{\partial L}{\partial w_1^2} = \delta_{p_i}. \tag{5.17}$$

Following the same principle, we can work our way one layer back and look at the effects of changing w_{21}^1 on the error function. This is the connection between $x2$ and $h1$. We again apply the chain rule and obtain

$$\frac{\partial L_i}{\partial w_{21}^1} = \frac{\partial Z_1^1}{\partial w_{21}^1} * \frac{\partial h_1}{\partial Z_1^1} * \frac{\partial Z^2}{\partial h_1} * \frac{\partial \hat{p}_i}{\partial Z^2} * \frac{\partial L_i}{\partial \hat{p}_i}. \tag{5.18}$$

We can solve this for each term in the equation and obtain

$$\frac{\partial L_i}{\partial w_{21}^1} = x_2 * h_1(1-h_1) * w_1^2 * \hat{p}_i(1-\hat{p}_i) * (\hat{p}_i - y_i). \tag{5.19}$$

We notice that the last two terms are identical to those of (5.16), and we can again cluster these in the same manner as in (5.17). So for this connection, the partial effect of the partial

derivative of the error function with respect to the respective weight is a product of the error calculated in the previous layer. Thus, the gradient calculated between the output and the hidden layer is propagated backwards through the network. This allows neural networks with many layers to be trained efficiently. The process presented in these steps is repeated for all weights in iterations. Usually, one for each training sample until w converges or a time limit is reached.

5.2 Model selection strategy

There are several choices one must make when building a neural network. These choices can be divided into the distinct layers of a neural network, namely the input, hidden and output layers. The number of input and output layers is always 1 for neural networks. Additionally, a binary classifier only needs a single neuron in the output layer. This leaves us to choose the number of hidden layers and neurons within the input and hidden layer. For the input layer, we choose to have 8 input features consisting of lagged returns to the indices. As for the hidden layer, Heaton (2008, p.158) states that there are no reasons to use neural networks with more than two hidden layers and that most practical problems can be solved using a single hidden layer. Géron (2019, p.319) also supports this claim by saying that "MLPs with a single hidden layer can model even the most complex functions provided it has enough neurons". Therefore, we will use a single hidden layer for our network. As for the number of neurons in the hidden layer, there is no clear-cut answer. Géron (2019, p.320) suggests gradually increasing the number of neurons until the model starts to overfit. Heaton (2008, p.159) also says it ultimately boils down to trial and error. Considering these points, we perform a grid search to look for the best model. Introducing more neurons in the hidden layer increases the complexity of the model. At the same time, the risk of overfitting increases. To account for this, we add a penalty to this increased complexity in the form of weight decay. That is, we train several models with a varying number of hidden layer neurons and different levels of weight decay. Finally, we choose the one that performs the best under the cross-validation.

5.2.1 Cross validation

A popular method to avoid overfitting and generalising the results of the model is to train the model using cross-validation. Cross-validation is a strategy used in statistics to evaluate and compare different learning algorithms. The procedure involves dividing data into two segments, one used for training and one used for validation. Generally, training and data sets cross over in successive rounds such that each data point can be validated against. The most widely used form for cross-validation is called k-fold cross-validation. Here the data is split into k equal parts where one $k-1$ folds are reserved for training the model, while the remaining fold is left out for validating the model. This is repeated k times such that each fold gets to serve as the validation set and is included in the training set. In the end, the performance across the test sets is averaged to give us the cross-validated estimate of the error for each model based on pre-determined performance criterion. Thus, we train models using 10-fold cross-validation for every combination of hidden layer neurons and level of weight decay. Finally, we choose the model that, on average, performs best.

5.3 Trend following strategies

Consider an active trading strategy based on the forecasts of the neural network. Specifically, we interpret the forecast of the model at time t as a signal regarding the state of the market at time $t+1$. As mentioned in section (4.3), the output of the model at time t indicates a bull or bear market for the subsequent period. Thus, we try to beat the market by investing in the financial asset and holding it over the next period if the signal from the model is bull and investing at a risk-free rate if the signal is bear. The return to this strategy is therefore given by

$$R_{t+1} = \begin{cases} r_{t+1} \text{ if Signal} = \text{Bull} \\ r_{f,t+1} \text{ if Signal} = \text{Bear} \end{cases} \tag{5.20}$$

where r_{t+1} and $r_{f,t+1}$ are the total stock return and the risk free rate of return respectively at time t. We are here only considering a long-only strategy. Alternatively, one can opt for a long-short strategy to exploit the opportunities in both bull and bear markets. In this case, whenever a sell signal is generated after a buy signal, we sell all our shares of the stock and

sell short the same number of shares. The profits are then invested at the risk-free rate over the next period. Using the long-only strategy, we simulate the returns to the active strategy over the same time period as the testing segment of our data, upon which the forecasts of our model are conducted. Furthermore, we estimate the alpha in the CAPM framework by regressing our active strategy's excess returns against the market's excess returns. We also compute the Sharpe ratio of our active strategy and test the hypothesis of equal performance.

5.3.1 Simple moving average

The moving average trading rule is among the oldest and most widely used by practitioners. It is motivated by the belief in trends in stock prices and seeks to uncover the underlying pattern of the price movements by smoothing the stock price over time. Zakamulin (2017b) shows that the trading indicator to a moving average strategy can be formulated using past returns instead of prices. Formally, the indicator is given by

$$I_t(n) \approx \sum_{i=0}^{n-2} w_i X_{t-i} \tag{5.21}$$

Where X_{t-i} is the the capital gain return on a financial asset from time period $t-i-1$ to $t-i$. For a simple moving average (SMA) rule, the weighting function is $w_i = n - i$ when using past returns to approximate the indicator. Furthermore, a trading signal is generated based on the technical indicator.

$$Signal_{t+1(n)} = \begin{cases} \text{Buy if } I_t(n) > 0 \\ \text{Sell if } I_t(n) < 0 \end{cases} \tag{5.22}$$

The intuition is that by averaging the returns with appropriate weighting over the period n, the signal from the indicator will tell us if the returns will be positive or negative. With this information, the signal for the next period is generated. This signal tells us to buy (sell) the stock if the value of our indicator is greater (less) than zero. We use the indicator to build a trend-following strategy. Specifically, when a buy signal is generated, we buy the stock at time t and hold it over the subsequent period $t+1$. If a sell signal is generated, we sell the stock, invest at the risk-free asset at t and wait for the next signal.

5.4 Evaluation of trading strategy

5.4.1 Sharpe ratio

A popular performance measure of a trading strategy amongst investors is the Sharpe ratio, developed by William F.Sharpe. The Sharpe ratio measures the performance of a portfolio based on the mean-variance theory. It is a meaningful measure of portfolio performance when the standard deviation can be used to adequately measure risk (Zakamouline & Koekebakker, 2009). The Sharpe ratio is computed as the ratio of the mean excess return per unit of risk. Given the excess return of an on a risky asset x, the Sharpe ratio is computed as

$$\frac{E[R_t - r_{ft}]}{\sigma} \tag{5.23}$$

Where $R_t - r_{ft}$ is the excess return and σ is the standard deviation. Using the simulated returns, we test the hypothesis that the active strategy is better than the benchmark strategies (BM). Specifically, using the Sharpe ratios of both strategies, we want to test the following hypothesis.

$$H_0 : SR_{NN} = SR_{BM}, \qquad H_1 : SR_{NN} > SR_{BM} \tag{5.24}$$

To test this null hypothesis, Jobson and Korkie (1981) present a test statistic

$$z = \frac{SR_A - SR_{BM}}{\frac{1}{T}[2(1-\rho) + \frac{1}{2}(SR_A^2 + SR_{BM}^2 - 2\rho^2 SR_A SR_{BM}} \tag{5.25}$$

where SR_A, SR_{BH}, and ρ are the estimated Sharpe ratios and correlation coefficient between the two strategies being tested over a sample period of T. However, under the null, z is asymptotically normal. In section 4.4, we concluded that the returns violate the assumption of normality and independency. Thus, the assumptions of the parametric test are not satisfied. This motivates the use of a non-parametric test instead. Therefore, we use the bootstrap method.

5.4.2 Bootstrap

Bootstrapping is a statistical method used to estimate the distribution of an estimator or test statistic by resampling the data (Zakamulin, 2021b). The bootstrap type depends on

whether the observations are assumed to be dependent or independent. When bootstrapping dependent observation, one needs to preserve the dependency when resampling. Therefore, one resamples a block of observations instead of single observations. This is known as a block-bootstrap and is conducted as follows: Firstly, we create a test statistic as shown in (5.26) and wish to test the null hypothesis in (5.25)

$$f = SR_{NN} - SR_{BM} \tag{5.26}$$

The null hypothesis in (5.25) states that the Sharpe ratio of the active strategy is equal to that of the benchmarks strategies. We carry out the test by jointly resampling the excess return of the active strategy and the buy-and-hold strategy. $r^*_{j,NN}$ represents the resampled returns to the network-based active strategy, where j indicates the j-th repetition of the bootstrap. Similarly, $r^*_{j,BM}$ represents the resampled returns to the benchmark strategies. The bootstrap is repeated j times, each time computing the Sharpe-ratio to the active strategy $SR^*_{j,NN}$ and the Sharpe ratio of the benchmarks $SR^*_{j,BM}$. The outperformance is then computed as:

$$H_0 : f = 0, \qquad H_1 : f > 0 \tag{5.27}$$

As formulated in (5.25), the null hypothesis states that the active strategy and the benchmarks perform similarly. Therefore, we adjust the computed outperformance by subtracting the observed outperformance in order to comply with the logic behind the null hypothesis. This gives us

$$f^*_j = (SR^*_{j,NN} - SR^*_{j,BM}) - f \tag{5.28}$$

Finally, we compute the p-value of the test by counting the number of times f^*_j is larger than f. Specifically,

$$p = \sum_{j=i}^{J} \frac{1^*_{f_j > f}}{J} \tag{5.29}$$

where $1^*_{f_j > f}$ is the indicator function that takes the values of 1 if $f^*_j > f$ and 0 otherwise.

5.4.3 Alpha

Another popular performance measure is to estimate a portfolio's α in the capital asset pricing model. This is done by regressing the excess return of the trend following strategies on the excess return of the market.

$$R_t = \alpha + \beta(R_{Mt} - r_{ft}) + \epsilon_t \tag{5.30}$$

Where R_t is the excess return to the trend-following strategy, $R_{Mt} - r_{ft}$ is the excess return to the market index, and β is the associated risk. The intuition is that the α of a trend following strategy is the excess return of the strategy relative to the benchmark index. Obviously, an α of zero means the trend following strategy performs equally to the benchmark index. Based on this intuition, we can formulate the following test:

$$H_0 : \alpha = 0, \qquad H_1 : \alpha > 0 \tag{5.31}$$

Additionally, to account for potential time series dependency and heteroskedasticity in our regression, we compute the Newey and West (1986) standard errors with 12 lags and report the one-sided p-values.

Chapter 6

Results

In this section, we will present our results. Firstly, we present the neural network architectures that gave the best performance and present the accuracy of their forecasts. Secondly, we present the results of our network-based active trading strategy compared to two benchmark strategies, one based on a simple moving average and one based on the buy and hold strategy. To test for statistical significance between the strategies, a geometric bootstrap is conducted to test the null hypothesis of an equal Sharpe ratio between the network strategy and the benchmark strategies. Finally, the α to the CAPM is estimated by regressing the active trading strategies' excess return to the market's excess return.

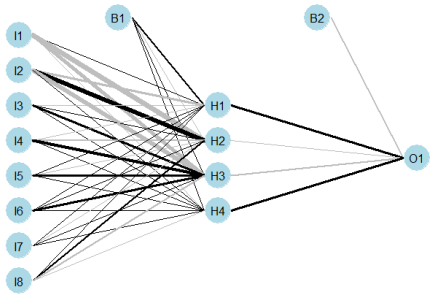
6.1 S&P Composite

6.1.1 Model architecture

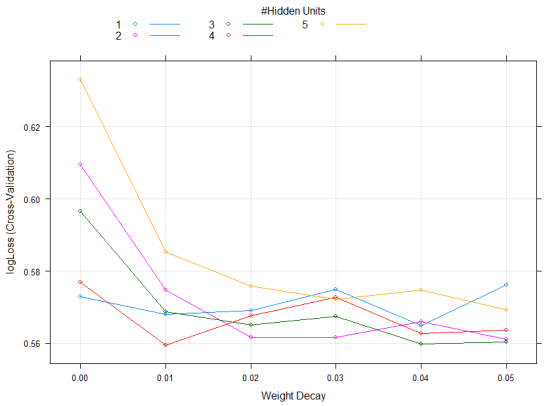
Figure 6.1 visualizes the network architectures using the training samples of S&P Composite returns for the testing periods identified on page 16. Specifically, (a), (c), and (e) show the network architecture that gives the lowest cross-validated log-loss. At the same time, (b), (d) and (f) show how different numbers of hidden layer units and levels of weight decay influenced the log loss during training. From the plots, we can see that 4, 5, and 3 hidden layer units combined with a weight decay of 0.01, 0.04 and 0.05 gives the lowest log loss for period 1, 2 and 3, respectively. These models gave an accuracy of 69%, 79% and 83% on their forecasts for periods 1, 2 and 3, respectively. The accuracy is calculated as the number of correct predictions divided by the total number of predictions.

Figure 6.1: Network architecture and corresponding tuning parameter for the S&P Composite

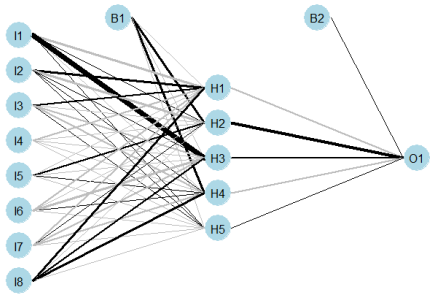
(a) Period 1



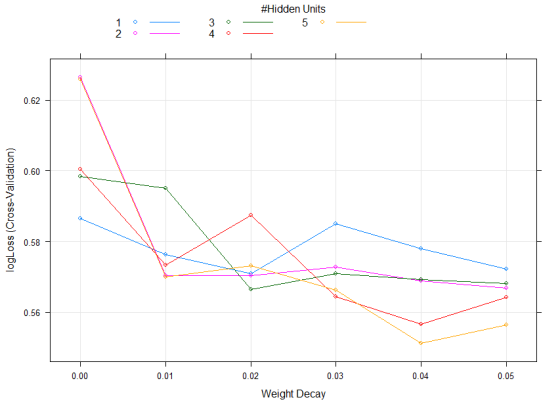
(b) Period 1



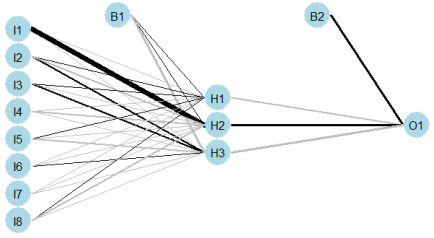
(c) Period 2



(d) Period 2



(e) Period 3



(f) Period 3

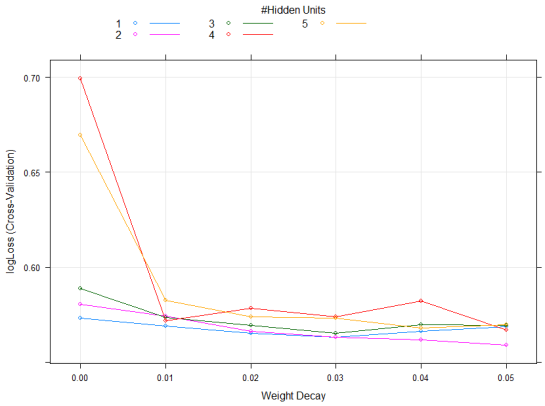


Figure a, c and e show the fitted models for the three testing periods for the S&P Composite. The thickness of the connection between the layers show the relative size of the weights. Positive weights are black while negative weights are grey. The horizontal axis in (b), (d) and (f) shows different levels of weight decay while the vertical axis shows the average log loss over the 10-fold cross validation.

6.1.2 Active strategy

The neural network trading strategy is based on the forecasts of the estimated model using the testing segment of the data. Specifically, the active trading strategy formulated in section 5.5 is applied to the S&P Composite index for all testing periods. The results are compared to a simple moving average trend following strategy using a window of 8 months to generate the weights. Figure 6.2 shows the descriptive statistics and performance of the network strategy, SMA(8) strategy, and the passive buy-and-hold strategy. Additionally, the cumulative log return to all strategies is visualized.

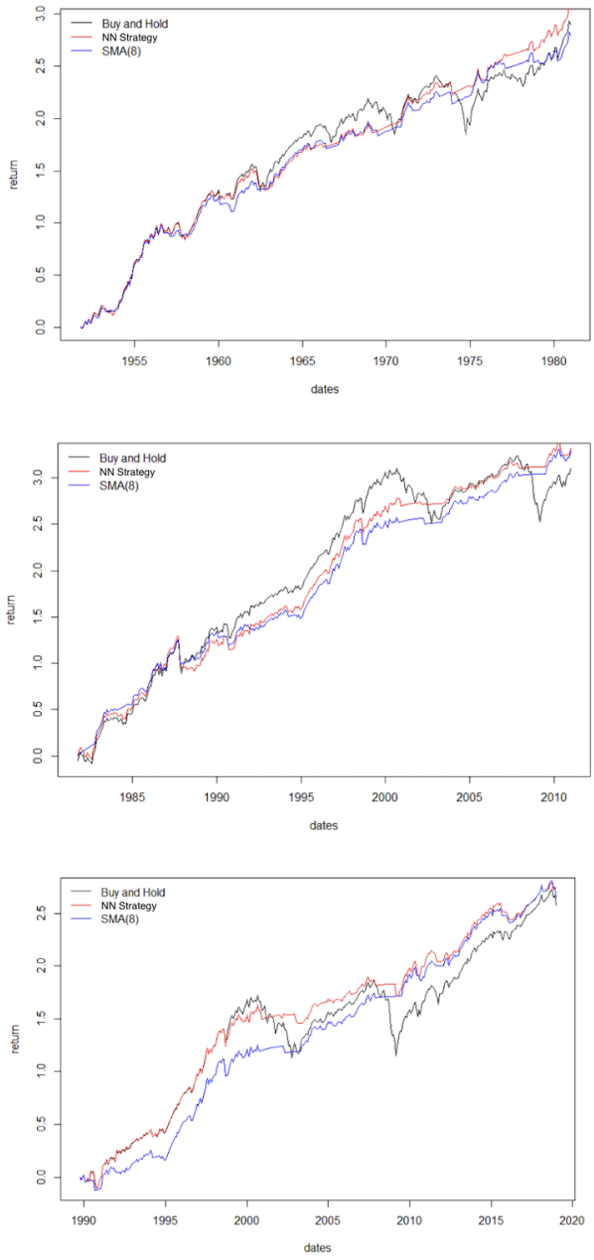
For period 1, all strategies generate similar mean returns, while the NN and SMA(8) strategies give noticeably lower standard deviation. Consequently, the Sharpe ratios of the trend following strategies are higher. The test for equal performance between the NN and benchmark strategies resulted in p-values greater than 0.05. Therefore, there is insufficient evidence to suggest that the NN strategy outperforms the benchmark. Additionally, both the NN and SMA(8) strategies gave positive α values. However, when accounting for time series dependency and heteroscedasticity in the regression, we do not observe significant p-values. Thus, there is insufficient evidence to suggest that the α to these trend-following strategies is greater than 0.

The network strategy gives a slightly higher mean return compared to the SMA(8) and BH strategy in period 2. The standard deviation is lower than the BH strategy but higher than the SMA strategy. Again, we observe a p-value greater than 0.05 for the Sharpe ratio difference between the NN strategy and the benchmarks. Therefore, we fail to reject the null hypothesis of equal performance. The NN and SMA(8) strategies generate a positive α for this period, with the SMA(8) having a significant p-value at the 5% level even when accounting for time series dependency and heteroscedasticity. The NN strategy shows a p-value of 0.14, and we fail to reject the null of zero α.

Period 3 shows similar results in terms of mean returns for all strategies. The standard deviation is significantly lower for the trend following strategies. Contrary to the preceding periods, we observe significant p-values between the NN strategy and the benchmarks

when testing for the Sharpe ratio difference. Therefore, we can reject the null hypothesis that states equal performance between the NN and the benchmarks. Both trend-following strategies generate positive α, with the SMA(8) being significant at the 5% level. The NN strategy shows a p-value of 0.16, and we fail to reject the null of zero α.

Figure 6.2: Descriptive statistics and performance of the trading strategies for the S&P Composite



S&P 1951-1980			
	BH	SMA	NN
Mean return	10.87	10.08	10.99
Std. deviation	13.76	10.13	10.59
Sharpe ratio	0.46	0.54	0.61
SR_{BH} vs SR_{NN}			(0.12)
SR_{SMA} vs SR_{NN}			(0.24)
Alpha CAMP		2.11	1.25
P-value		(0.14)	(0.37)

S&P 1981-2010			
	BH	SMA	NN
Mean return	11.86	11.93	12.17
Std. deviation	15.60	11.54	12.69
Sharpe ratio	0.45	0.61	0.58
SR_{BH} vs SR_{NN}			(0.20)
SR_{SMA} vs SR_{NN}			(0.67)
Alpha CAPM		3.24	2.66
P-value		(0.04)	(0.14)

S&P 1991-2018			
	BH	SMA	NN
Mean return	9.83	9.58	9.99
Std. deviation	14.17	10.16	11.00
Sharpe ratio	0.50	0.67	0.66
SR_{BH} vs SR_{NN}			(0.05)
SR_{SMA} vs SR_{NN}			(0.05)
Alpha CAPM		3.17	2.97
P-value		(0.04)	(0.16)

The figures show the cumulative log return over the testing periods for S&P Composite. The table reports the performance of the trading strategies. One sided p-values are reported in brackets.

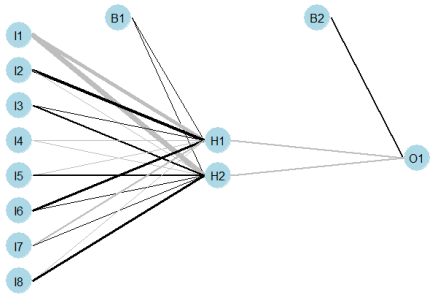
6.2 Dji

6.2.1 Model architecture

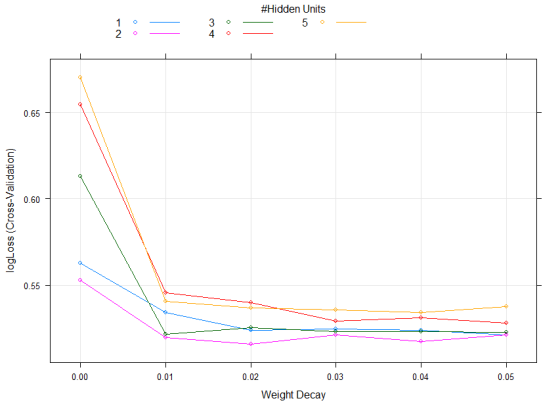
In this section, we briefly repeat the study in the previous section using the data for the DJI index for the whole sample period from January 1897 to December 2020. Figure 6.3 visualizes the fitted models using the DJI returns for the testing periods identified on page 16. On the left-hand side, (a), (c), and (e) show the network architectures that give the lowest cross-validated log loss. On the right-hand side, (b), (d), and (f) show how different numbers of hidden layer neurons and levels of weight decay influenced the log-loss during training. The plots show that a network with 2, 1 and 1 hidden layer neurons gave the best performance for periods 1, 2 and 3, respectively. These models were trained using a weight decay of 0.02, 0.05 and 0.02. These models gave an accuracy of 75%, 75%, and 73% for the respective periods.

Figure 6.3: Network architecture and corresponding tuning parameter for the DJI

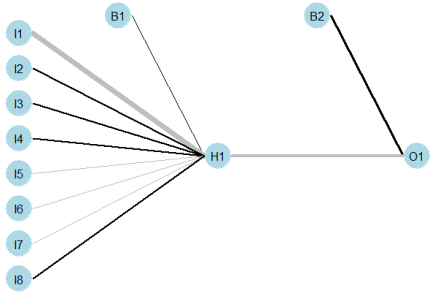
(a) Period 1



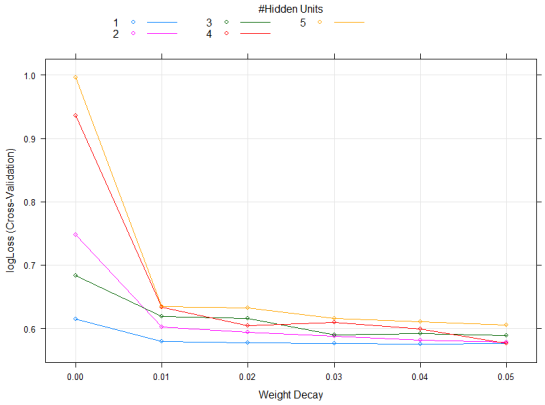
(b) Period 1



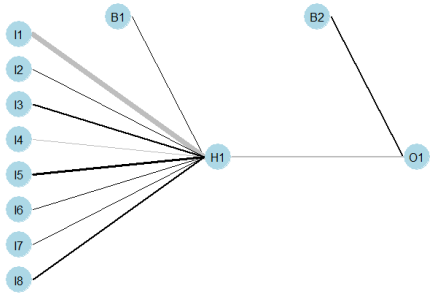
(c) Period 2



(d) Period 2



(e) Period 3



(f) Period 3

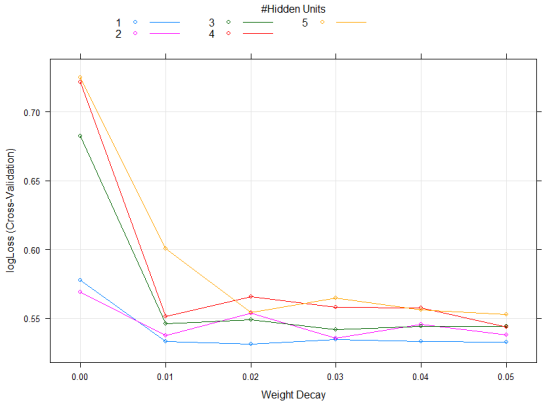


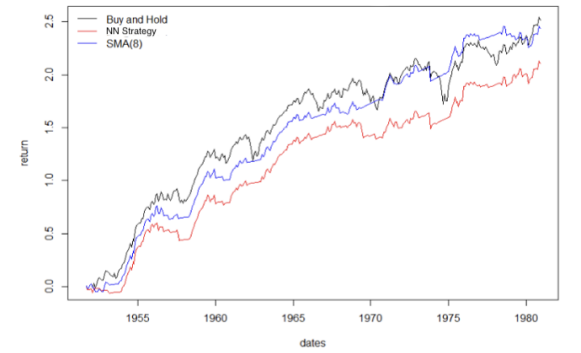
Figure a, c and e show the fitted models for the three testing periods for the DJI. The thickness of the connection between the layers show the relative size of the weights. Positive weights are black while negative weights are grey. The horizontal axis in (b), (d) and (f) shows different levels of weight decay while the vertical axis shows the average log loss over the 10-fold cross validation.

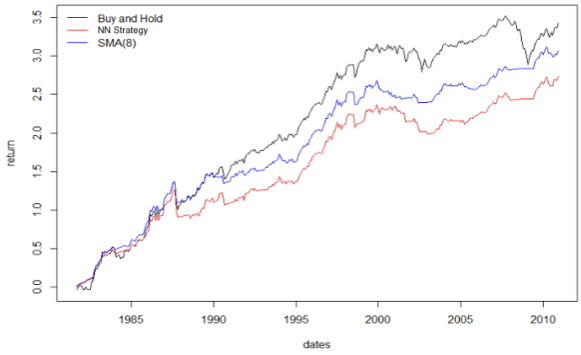
6.2.2 Trading strategy

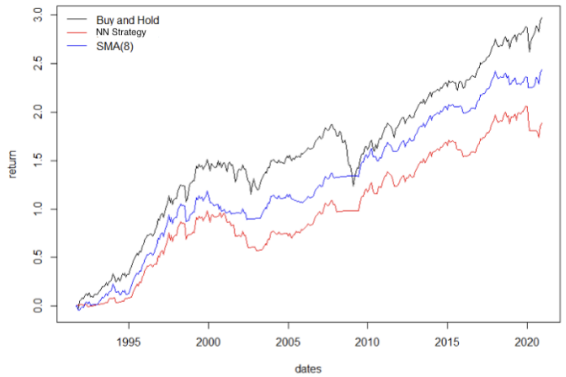
Again, the network-based active trading strategy formulated in section 5.4 is applied to the DJI index for all testing periods. Figure 6.4 shows the descriptive statistics and performance of the NN, SMA, and BH strategies. The cumulative log return for all strategies for all periods is also visualized. The results of the NN strategy are compared to a simple moving average benchmark and a passive buy and hold benchmark.

The NN strategy generates the lowest mean return for all periods, while the BH strategy generates the highest. The BH strategy also gives the highest standard deviation. The Sharpe ratios of the different strategies are similar, with the Sharpe ratio of SMA(8) being slightly higher. We observe p-values greater than 0.05 when testing for equal performance between the NN strategy and the benchmarks. Therefore, we fail to reject the null hypothesis of equal performance. Furthermore, both trend-following strategies show a positive α with the exception of the NN strategy in period 3. None of these is significant when accounting for serial dependency and heteroscedasticity. Therefore, we fail to reject the null hypothesis of zero α.

Figure 6.4: Descriptive statistics and performance of the trading strategies for the DJI







DJI 1951-1980			
	BH	SMA	NN
Mean return	9.53	8.80	7.69
Std. deviation	13.60	9.81	9.92
Sharpe ratio	0.38	0.45	0.34
SR_{BH} vs SR_{NN}			(0.45)
SR_{SMA} vs SR_{NN}			(0.72)
Alpha CAPM		1.75	1.01
		(0.25)	(0.36)

DJI 1981-2010			
	BH	SMA	NN
Mean return	12.93	11.26	10.18
Std. deviation	15.46	12.10	12.67
Sharpe ratio	0.53	0.53	0.43
SR_{BH} vs SR_{NN}			(0.73)
SR_{SMA} vs SR_{NN}			(0.76)
Alpha CAPM		1.50	0.18
		(0.38)	(0.91)

DJI 1991-2020			
	BH	SMA	NN
Mean return	11.21	8.91	7.10
Std. deviation	14.47	10.79	11.39
Sharpe ratio	0.61	0.61	0.42
SR_{BH} vs SR_{NN}			(0.93)
SR_{SMA} vs SR_{NN}			(0.94)
Alpha CAPM		1.66	-0.68
		(0.32)	(0.70)

The figures show the cumulative log return over the testing periods for the DJI. The table reports the performance of the trading strategies. One sided p-values are reported in brackets.

Chapter 7

Discussions

Our findings show interesting results. The first task in our study was to fit a network to our data and make predictions of the future market state. An input consisting of 8 neurons was manually defined, and the results show network configurations that differ across all periods for both indices. An interesting find shows that the S&P Composite periods' models are more complex than those for DJI. Specifically, the models fitted for periods 1, 2 and 3 for the S&P Composite have 4, 5 and 3 neurons in the hidden layer, while the same periods for DJI have 2, 1 and 1 neurons in the hidden layer. The models for the DJI data are thus more simple than those for the S&P Composite in terms of hidden layer neurons. Furthermore, several models have big weights for higher-order lags, indicating that they are being prioritized in modelling the pattern in the data.

The results of the different trading strategies showed that the Sharpe ratio of the NN strategy was statistically insignificant from the Sharpe ratios of the BH and SMA(8) strategies. An exception to this was for S&P Composite period 3, which did show a significant Sharpe ratio difference between the NN and both benchmark strategies. This indicates equal performance between the NN strategies and the benchmark strategies. Although the Sharpe ratio is a popular performance measure among practitioners, we should highlight some of the shortcomings. Firstly, it follows the assumption that investors only care for the mean and variance of returns. Therefore, higher-order moments such as skewness and kurtosis are neglected (Kadan & Liu, 2014). Furthermore, section 4.1 showed that the data at hand strongly deviates from a normal distribution. Therefore, these aspects should be considered

when constructing a portfolio or trading strategy. Secondly, the standard deviation is not a good performance of risk from the viewpoint of an investor since it treats both downside risk and upside return potential equally.

An important aspect of the data is the labelling of bull and bear markets. As mentioned in section 4.4, this transformation follows the work of Bry and Boschan (1971)). Since our target value, Y is a result of this transformation, one must be aware of the shortcomings. Firstly, the parameters in the algorithm, τ_{window}, τ_{censor}, τ_{phase}, τ_{cycle} and θ are chosen arbitrarily. As a result, the algorithm sometimes fails to correctly identify bull and bear markets (Zakamulin, 2017a). This gives us another aspect of uncertainty in our data. One can mitigate this by reducing τ_{window}, and see if prices generally increase (decrease) when the algorithm classifies a bull (bear) market. However, we see this introduction of uncertainty to our target values as a reasonable trade-off allowing for a higher signal-to-noise for the network.

From table 4.2, we can see that both indices had a higher proportion of bull markets in terms of their duration on all the descriptive statistics. Consequently, using the data at a monthly frequency gives us uneven amounts of bull and bear markets. One should be aware of this when evaluating the forecasts of the model. For instance, if we look at a testing set where 80% of the data is classified as a bull market, a dumb model that always predicted bull would have an accuracy of 80%, which generally is a good result. In reality, an active strategy based on the signals from such a model would be the same as the buy-and-hold strategy.

As a second performance measure, we estimated the alpha in the CAPM with the null hypothesis stating the trend following strategies to have zero alpha. For both indices, the NN strategy generated positive alphas for all periods except for DJI period 3. However, they were not significant when accounting for serial dependency and heteroscedasticity. We therefore fail to reject the null hypothesis of zero alpha. Like the Sharpe ratio, the CAPM also assumes that investors only care about expected return and volatility. Additionally, the model assumes the beta to be the only risk factor in the market (Womack & Zhang, 2003).

As mentioned earlier, the data in this study strongly violates the assumption of normal distribution. Thus, only focusing on the expected return and volatility leads to higher-order moments being neglected when they should be accounted for. In addition, researchers believe that other risk factors significantly impact the expected return and should therefore be considered. (Womack & Zhang, 2003).

One can argue that a testing period of around 30 years, as used in this study, might be too short of a window to evaluate the various trading strategies. However, training a neural network requires sufficient amounts of data, meaning enough observations of the dependent variable. At the same time, one needs enough data in the testing sample to generalize the network's output on unseen data. Splitting the data into a training and testing set of roughly 75 and 25% aimed to find a reasonable middle ground. To put this in perspective, for a data set containing 10 million observations, using 1% of the data to test would probably be enough, as 100 000 data points used to forecast would tell us something about the models ability to generalize.

We would also like to shed some light on the existing literature and its results. One should be aware that most of the studies focus on the properties of the algorithms used and lack a financial justification when suggesting that machine learning algorithms perform better than other strategies. In most cases, the accuracy of a machine learning-based forecaster was evaluated in isolation with little to no regard for the financial and statistical performance of such forecasters. In the process of exploring existing literature, we quickly noticed optimistic results. However, there was often a lack of statistical procedures to justify these results. The absence of such statistical procedures can be misguiding in the evaluation of network-based trading strategies. For instance, our results are aligned with Kara et al. (2011) and Pan et al. (2005) in terms of accuracy of forecasts. However, our results also show that the NN-based trading strategy is no better than a passive BH strategy or a simple trend-following strategy based on moving averages. Furthermore, Qi (1999) showed that a network-based strategy gave higher profits and lower risk than the BH strategy. At first glance, our results also show that the NN strategy gives higher profit and lower risk for the S&P Composite. However, 2/3 of the results were not significant when statistically tested.

Chapter 8

Conclusions

The main goal of this thesis was to explore the capability of neural networks to model financial data. To do this, we trained several networks using two major stock indices in the US, namely the S&P Composite and the DJI. In standard machine learning fashion, the data was split into a training and testing sample. To better generalize the results, three different testing periods were chosen. Furthermore, the shortcomings of financial data in machine learning pointed out by Israel et al. (2020) were to a degree overcome by predicting bear and bull markets as opposed to the traditional approach of predicting returns or signs of returns. The results showed that the neural networks, on average, correctly predicted the market state with an accuracy of 75%.

In addition, an active trading strategy was constructed using the networks' predictions. To evaluate the profitability of the predictions, we compared the Sharpe ratio of the network strategy to the Sharpe ratio of two benchmark (BM) strategies. Using a block-bootstrap, we estimated the Sharpe ratio difference between the NN strategy and the BM strategies. The results showed that the NN strategy did not generate a Sharpe ratio significantly larger than the BM strategies in two of three testing periods for the S&P Composite. For the DJI, the network strategy gave a lower Sharpe ratio than both BM strategies for all testing periods. Furthermore, the α in the capital asset pricing model was estimated by regressing the network strategy's excess return to the market's excess return. The results showed that the NN strategy could not generate a statistically significant α for any of the testing periods for both indices.

To summarize, neural networks have proven to be a powerful tool for forecasting financial time series. The networks trained in this thesis managed to predict the state of the market with a high level of accuracy, similar to the results of Kara et al. (2011) and Pan et al. (2005). However, a trading strategy based on these predictions was not superior to other traditional trading strategies such as the passive BH strategy and an active SMA strategy. Our results should highlight the importance of statistical tests to evaluate trading strategies. Furthermore, following the notion of Israel et al. (2020) we believe that machine learning, hereunder neural networks, should serve as a tool, supplementary to existing financial models and not as a replacement.

Appendix A

Confusion matrices

Table A.1: S&P Composite period 1

		Predicted	
		Bear	Bull
Actual	Bear	42	60
	Bull	49	201

Table A.2: S&P Composite Period 2

		Predicted	
		Bear	Bull
Actual	Bear	40	40
	Bull	32	240

Table A.3: S&P Composite Period 3

		Predicted	
		Bear	Bull
Actual	Bear	41	29
	Bull	26	232

Table A.4: DJI period 1

		Predicted	
		Bear	Bull
Actual	Bear	52	33
	Bull	54	213

Table A.5: DJI period 2

		Predicted	
		Bear	Bull
Actual	Bear	33	31
	Bull	58	230

Table A.6: DJI period 3

		Predicted	
		Bear	Bull
Actual	Bear	26	24
	Bull	71	231

Appendix B

R code

listings

```
rm(list=ls(all=TRUE))
library(tibble)
library(sandwich)
library(zoo)
library(tseries)
library(neuralnet)
library(xts)
library(TTR)
library(Rcpp)
library(bbdetection)
library(boot)
library(lmtest)
library(caret)
library(gbm)
library(nnet)
library(NeuralNetTools)
library(ggplot2)
library(NeuralSens)
library(tidyverse)
library(moments)
library(xtable)
library(pROC)
library(MLmetrics)
source("tfutils.r")
```

```
ws.lma <- function(n) {
  # the weighting scheme of the LMA rule
  ws <- (1:n)*(2:(n+1))/2
  return(ws)
}

sim_tfrule <- function(cap, mkt, rf, ws, short=FALSE) {
  # this function simulates the returns to a trend-following strategy
  # Inputs:
  # cap - capital gain return
  # mkt - total return
  # rf  - risk-free rate of return
  # ws  - return-weighting function; the weight of the most recent return ...
     is in the last element of ws
  # short - if TRUE, simulate the Long-Short strategy; otherwise - ...
     Long-only strategy

  nobs = length(cap) # compute the length of vector cap
  n = length(ws) # compute the window size in the trend-following rule
  r = rep(NA, nobs) # to hold simulated returns

  # for-loop to simulate the returns
  for(j in n:(nobs-1)) {
    # compute the value of the trading indicator
    signal = sum( cap[(j-n+1):j]*ws )

    # compute the next period return
    if(signal > 0) {
      # if signal is Buy, invest in the stocks
      r[j+1] = mkt[j+1]
    }
    else {
      # if signal is Sell
      if (short==FALSE) {
        # if Long-only, invest in the risk-free
        r[j+1] = rf[j+1]
```

```
      } else {
        # if Long-short, sell short the stocks
        r[j+1] = 2*rf[j+1] - mkt[j+1]
      }
    }
  }
  # return the returns to a trend-following strategy
  return (r)
}


Sharpe <- function(ex) {
  # computes the (annualized) Sharpe ratio
  # ex - vector of excess returns
  return(mean(ex, na.rm=TRUE)/sd(ex, na.rm=TRUE)*sqrt(12))
}

srdiff <- function(data)  {
  # this function computes the test statistics for each bootstrapped ...
     resample of data
  mktmrf <- data[,1] # vector that contains the excess returns to the market
  retmrf <- data[,2] # vector that contains the excess returns to the nn ...
     strategy
  # compute the f statistics
  f <- Sharpe(retmrf) - Sharpe(mktmrf)
  return(f)
}

AlphaTest <- function(X, retmrf) {
  # compute the z-statistic in a multi-factor model and other quantities ...
     of interest
  # Inputs:
  # X is the matrix of factor returns
  # retmrf is the excess returns to a mutual fund /  trading strategy
  # Outputs:
  # alpha - alpha of the Trend-following strategy
  # z  - z statistic (t-statistic) in the test
```

```
  # pval - p-value of alpha
  # yhat - fitted values
  # resid - residuals

  # to improve the robustness, conversion to required types
  X <- as.matrix(X)
  retmrf <- as.vector(retmrf)
  # perform required computations
  X <- cbind(1,X) # add vector of ones as the first column
  Y <- retmrf
  n <- length(Y)  # number of observations
  k <- ncol(X)    # number of independent variables
  xxi <- solve(t(X) %*% X)
  xy <- t(X) %*% Y
  beta <- as.vector(xxi %*% xy)          # regression coefficients
  yhat <- as.vector(X %*% beta)          # fitted values of Y
  resid <- Y - yhat                      # model residuals
  rss <-  as.numeric(resid %*% resid)    # residual sum of squares
  se <- sqrt(diag(rss/(n-k)*xxi))        # standard errors of estimation of ...
     beta
  tstat <- as.vector(beta/se)            # t-statistics of beta
  pval <- pnorm(-tstat)
  return(list(alpha=beta[1],z=tstat[1],pval=pval[1],
              yhat=yhat,resid=resid))
}

#Create a function to simulate the returns using NN predictions
sim_nn <- function(cap, mkt, rf, short=FALSE) {
  nobs = length(cap) # compute the length of vector cap
  r = rep(0, (nobs)) # to hold simulated returns

  # for-loop to simulate the returns
  for(j in 1:(nobs-1)) {
    signal = Ypred
    if(signal[j] == "Bull") {
      r[j+1] = mkt[j+1]
    }
```

```
    else {
      if (short==FALSE) {
        r[j+1] = rf[j+1]
      } else {
        r[j+1] = 2*rf[j+1] - mkt[j+1]
      }
    }
  }
  return (r)
}

lagpad <- function(x, k) {
  if (k>0) {
    return (c(rep(NA, k), x)[1 : length(x)] );
  }
  else {
    return (c(x[(-k+1) : length(x)], rep(NA, -k)));
  }
}



sp500<-read.zoo("sp500.txt",header=TRUE)
sp500<-window(sp500,start=as.Date.character("1857-01-01"),
end=as.Date("2018-12-31"))
dates<-index(sp500)
data<-coredata(sp500)
return<-data[,"CAP"]
price<-cumprod(1+return)
bull<-run_dating_alg(price)

bb.summary.stat(price,bull)
bb.plot(price,bull,dates)
train<-window(sp500,start=as.Date.character("1860-01-01"),
end=as.Date("1990-12-31"))
test<-window(sp500,start=as.Date.character("1991-01-01"),
end=as.Date("2018-12-31"))
```

```
#Make a Y vector containing the sign of market state
dates.train <- index(train)
data.train <- coredata(train)
ret.train <- data.train[,"CAP"]
Price.train<-cumprod(1+ret.train)
Bull.train<-run_dating_alg(Price.train)
Y.train<-ifelse(Bull.train==TRUE,"Bull","Bear")
Y.train<-as.factor(Y.train)

dates.test <- index(test)
data.test <- coredata(test)
ret.test <- data.test[,"CAP"]
Price.test<-cumprod(1+ret.test)
Bull.test<-run_dating_alg(Price.test)
Y.test<-ifelse(Bull.test==TRUE,"Bull","Bear")
Y.test<-as.factor(Y.test)

bb.summary.stat(Price.train,Bull.train)
bb.summary.stat(Price.test,Bull.test)
bb.plot(Price.train,Bull.train,dates.train)
bb.plot(Price.test,Bull.test,dates.test)

#Create lagged values for training sample
X1<-lagpad(x=Price.train,1)
X2<-lagpad(x=Price.train,2)
X3<-lagpad(x=Price.train,3)
X4<-lagpad(x=Price.train,4)
X5<-lagpad(x=Price.train,5)
X6<-lagpad(x=Price.train,6)
X7<-lagpad(x=Price.train,7)
X8<-lagpad(x=Price.train,8)

#Create lagged values for testing sample
t.X1<-lagpad(x=Price.test,1)
t.X2<-lagpad(x=Price.test,2)
t.X3<-lagpad(x=Price.test,3)
t.X4<-lagpad(x=Price.test,4)
```

```
t.X5<-lagpad(x=Price.test,5)
t.X6<-lagpad(x=Price.test,6)
t.X7<-lagpad(x=Price.test,7)
t.X8<-lagpad(x=Price.test,8)


#create data frame containing state of market and lagged returns
df.train<-data.frame(Y.train,X1,X2,X3,X4,X5,X6,X7,X8)

#trim the data frame
df.train<-df.train[-c(1:8),]

df.test<-data.frame(Y.test,t.X1,t.X2,t.X3,t.X4,t.X5,t.X6,t.X7,t.X8)

df.test<-df.test[-c(1:8),]

colnames(df.test)=colnames(df.train)

fitcontrol<-trainControl(method = "cv", number = 10,
                         summaryFunction = multiClassSummary,
                         classProbs = TRUE,
                         savePredictions = TRUE)

nngrid<-expand.grid(size = seq(from = 1, to = 5, by = 1),
                    decay = seq(from = 0, to = 0.05, by = 0.01))

set.seed(100)
nnfit1<-train(Y.train ~., data=df.train, #Train using cross validation ...
   with accuracy as measurement
              method="nnet",
              metric="logLoss",
              trControl=fitcontrol,
              tuneGrid=nngrid,
              linout=FALSE)

mlp_pred<-predict(nnfit1,df.train)
train.acc<-confusionMatrix(data=mlp_pred,reference = df.train$Y.train, ...
```

```
    positive = "Bull")$overall[1]
mlp_pred<-predict(nnfit1,df.test)
test.acc<-confusionMatrix(data=mlp_pred,reference = df.test$Y.train, ...
    positive = "Bull")$overall[1]

plot(nnfit1)
plotnet(nnfit1$finalModel,var_labs=FALSE)

varImp(nnfit1)
df<-nnfit1$results
SensAnalysisMLP(nnfit1)

Ypred <-mlp_pred
Y.test<-df.test$Y.train

length(Ypred)
length(Y.test)



#Confusion matrix
accuracy<-table(Reference=Y.test,Prediction=Ypred)
xtable(accuracy)

################# Simulate SMA ##################################
n<-8

sp500<-read.zoo("sp500.txt",header=TRUE)
sp500 <- window(sp500, start=as.Date("1991-01-01"),end=as.Date("2018-12-31"))
dates <- index(sp500)    # get dates
data <- coredata(sp500) # get data
mkt <- data[,"MKT"] # the total return
cap <- data[,"CAP"] # the capital gain return
rf <- data[,"RF"] # the risk-free rate of return

ws <- ws.lma(n)
r.sma <- sim_tfrule(cap, mkt, rf, ws, short=FALSE)
nobs <- length(mkt)
```

```
m <- length(ws)
r.sma <- r.sma[(m+1):nobs]
mkt <- mkt[(m+1):nobs]
rf <- rf[(m+1):nobs]
dates <- dates[(m+1):nobs]



################ Simulate NN ##########################
n<-8
sp500<-read.zoo("sp500.txt",header=TRUE)
sp500 <- window(sp500, start=as.Date("1991-01-01"),end=as.Date("2018-12-31"))
# get the dates and data
dates <- index(sp500)   # get dates
dates<-dates[-c(1:n)]
data <- coredata(sp500) # get data
mkt <- data[,"MKT"] # the total return
mkt<-mkt[-c(1:n)]
cap <- data[,"CAP"] # the capital gain return
cap<-cap[-c(1:n)]
rf <- data[,"RF"] # the risk-free rate of return
rf<-rf[-c(1:n)]

NN_returns<-sim_nn(cap,mkt,rf,short=FALSE)
r.nn <- NN_returns

#Compute the cumulative returns to NN strategy and BH strategy
ind.r   <- cumprod(1 + r.nn)
ind.mkt <- cumprod(1 + mkt)
ind.sma <- cumprod(1 + r.sma)

plot(dates,log(ind.mkt),type="l", main = "Cumulative return of trading ...
    strategies",
     ylab = "return", ylim=c(0,3))
lines(dates,log(ind.r),type="l",col=("red"))
lines(dates,log(ind.sma),type = "l",col=("blue"))
legend("topleft",legend=c("Buy and Hold","NN ...
    strategy","SMA(8)"),col=c("black","red","blue"),lty=1,bty = "n")
```

```
# compute the descriptive statistics and the performance
mean.mkt <- mean(mkt)*12*100
mean.sma <- mean(r.sma)*12*100
mean.nn <-mean(r.nn)*12*100
sd.mkt <- sd(mkt)*sqrt(12)*100
sd.sma <- sd(r.sma)*sqrt(12)*100
sd.nn <- sd(r.nn)*sqrt(12)*100
SR.mkt <- Sharpe(mkt-rf)
SR.sma <- Sharpe(r.sma-rf)
SR.nn <- Sharpe(r.nn-rf)
pval <- SharpeTest(mkt-rf, r.sma-rf)$pval
pval


###################### Alpha ########################
mktmrf <- mkt - rf #Excess return to the BH strategy
retmrf <- r.nn - rf #Excess return to the NN long only strategy

Alpha.nn<-AlphaTest(mktmrf,retmrf)
alpha.nn<-(Alpha.nn$alpha)*100*12
pval.alpha.nn<-Alpha.nn$pval

mktmrf <- mkt - rf #Excess return to the BH strategy
smamrf <- r.sma- rf #Excess return to the NN long only strategy

Alpha.sma<-AlphaTest(mktmrf,smamrf)
alpha.sma<-(Alpha.sma$alpha)*100*12
pval.alpha.sma<-Alpha.sma$pval

############################## bootstrapped sharpe ...
    ##########################################
mktmrf <- mkt - rf #Excess return to the BH strategy
retmrf <- r.nn - rf #Excess return to the NN long only strategy

srdiff <- function(data)  {
  # this function computes the test statistics for each bootstrapped ...
```

```
    resample of data
  mktmrf <- data[,1] # vector that contains the excess returns to the market
  retmrf <- data[,2] # vector that contains the excess returns to the nn ...
    strategy
  # compute the f statistics
  f <- Sharpe(retmrf) - Sharpe(mktmrf)
  return(f)
}

fstat <- Sharpe(retmrf) - Sharpe(mktmrf)
data <- cbind(mktmrf, retmrf)

bLen = 5
R <- 5000
# bootstrap time series
results <- tsboot(tseries=data, statistic=srdiff,
                  R=R, l=bLen, sim="geom")



# The result is a matrix that contains bootstrapped f-statistics
fstat.sim <- results$t - fstat
# compute the p-value of the test
pval.boot.sr <- length(which(fstat.sim > fstat))/R
pval.boot.sr

################################## SMA vs NN ...
   ###################################
smamrf <- r.sma - rf #Excess return to the BH strategy
retmrf <- r.nn - rf #Excess return to the NN long only strategy

srdiff <- function(data)  {
  # this function computes the test statistics for each bootstrapped ...
    resample of data
  smamrf <- data[,1] # vector that contains the excess returns to the market
  retmrf <- data[,2] # vector that contains the excess returns to the nn ...
    strategy
  # compute the f statistics
```

```
  f <- Sharpe(retmrf) - Sharpe(smamrf)
  return(f)
}

fstat <- Sharpe(retmrf) - Sharpe(smamrf)
data <- cbind(smamrf, retmrf)

bLen = 5
R <- 5000
# bootstrap time series
results <- tsboot(tseries=data, statistic=srdiff,
                  R=R, l=bLen, sim="geom")

# The result is a matrix that contains bootstrapped f-statistics
fstat.sim <- results$t - fstat
# compute the p-value of the test
pval.boot.nnsma <- length(which(fstat.sim > fstat))/R
pval.boot.nnsma


df <- data.frame(BH=c(mean.mkt, sd.mkt, SR.mkt, NA,NA,NA),
                 SMA=c(mean.sma, sd.sma, SR.sma, ...
                    pval.boot.sma,alpha.sma,pval.alpha.sma),
                 NN=c(mean.nn,sd.nn,SR.nn,pval.boot.sr,alpha.nn,pval.alpha.nn))
rownames(df) <- c("Mean return", "Std. deviation", "Sharpe ratio", ...
    "P-value","Alpha","P-value alpha")
xtable(df)

######################## Newey west ######################################
retmrf<-r.nn-rf
mktmrf<-mkt-rf

reslm <- lm(retmrf ~ mktmrf)
res <- coeftest(reslm, df = Inf, vcov = NeweyWest(reslm, lag = 12, ...
    prewhite = FALSE))
res
```

```
smamrf<-r.sma-rf
mktmrf<-mkt-rf
reslm <- lm(smamrf ~ mktmrf)
res <- coeftest(reslm, df = Inf, vcov = NeweyWest(reslm, lag = 12, ...
    prewhite = FALSE))
res

Sys.setlocale("LC_TIME", "English") # you need this command on computers ...
    with Norwegian Windows
DJ <- read.zoo("djia.txt", header=TRUE, FUN=as.yearmon)
train<-window(DJ, index. = index(DJ), start = as.character.Date("Jan ...
    1897"), end = as.character.Date("Dec 1990"))
test<-window(DJ, index. = index(DJ), start = as.character.Date("Jan ...
    1991"), end = as.character.Date("Dec 2020"))

dates<-index(DJ)
dates<-as.yearmon(dates)
data<-coredata(DJ)
return<-data[,"CAP"]
price<-cumprod(1+return)
bull<-run_filtering_alg(price)

bb.summary.stat(price,bull)
bb.plot(price,bull,as.yearmon(dates))

#Make a Y vector containing the sign of market state
dates.train <- index(train)
data.train <- coredata(train)
ret.train <- data.train[,"CAP"]
Price.train<-cumprod(1+ret.train)
Bull.train<-run_filtering_alg(Price.train)
Y.train<-ifelse(Bull.train==TRUE,"Bull","Bear")
Y.train<-as.factor(Y.train)



dates.test <- index(test)
data.test <- coredata(test)
```

```
ret.test <- data.test[,"CAP"]
Price.test<-cumprod(1+ret.test)
Bull.test<-run_filtering_alg(Price.test)
Y.test<-ifelse(Bull.test==TRUE,"Bull","Bear")
Y.test<-as.factor(Y.test)



#Create lagged values for training sample
X1<-lagpad(x=Price.train,1)
X2<-lagpad(x=Price.train,2)
X3<-lagpad(x=Price.train,3)
X4<-lagpad(x=Price.train,4)
X5<-lagpad(x=Price.train,5)
X6<-lagpad(x=Price.train,6)
X7<-lagpad(x=Price.train,7)
X8<-lagpad(x=Price.train,8)



#Create lagged values for testing sample
t.X1<-lagpad(x=Price.test,1)
t.X2<-lagpad(x=Price.test,2)
t.X3<-lagpad(x=Price.test,3)
t.X4<-lagpad(x=Price.test,4)
t.X5<-lagpad(x=Price.test,5)
t.X6<-lagpad(x=Price.test,6)
t.X7<-lagpad(x=Price.test,7)
t.X8<-lagpad(x=Price.test,8)



#create data frame containing state of market and lagged returns
df.train<-data.frame(Y.train,X1,X2,X3,X4,X5,X6,X7,X8)
df.train<-df.train[-c(1:8),]

df.test<-data.frame(Y.test,t.X1,t.X2,t.X3,t.X4,t.X5,t.X6,t.X7,t.X8)
df.test<-df.test[-c(1:8),]
```

```
fitcontrol<-trainControl(method = "cv", number = 10,
                         summaryFunction = multiClassSummary,
                         classProbs = TRUE,
                         savePredictions = TRUE)

nngrid<-expand.grid(size = seq(from = 1, to = 5, by = 1),
                    decay = seq(from = 0, to = 0.05, by = 0.01))

colnames(df.test)=colnames(df.train)
set.seed(100)
nnfit2<-train(Y.train ~., data=df.train, #Train using cross validation ...
   with accuracy as measurement
              method="nnet",
              metric="logLoss",
              trControl=fitcontrol,
              tuneGrid=nngrid,
              linout=FALSE)

mlp_pred<-predict(nnfit2,df.train)
accuracy.train<-confusionMatrix(data=mlp_pred,reference = ...
   df.train$Y.train, positive = "Bull")$overall[1]
mlp_pred<-predict(nnfit2,df.test)
accuracy.test<-confusionMatrix(data=mlp_pred,reference = df.test$Y.train, ...
   positive = "Bull")$overall[1]

varImp(nnfit2)
plot(nnfit2)
plotnet(nnfit2$finalModel,var_labs=FALSE)

#make predictions on the testing sample using the model
Ypred <- mlp_pred
Y.test<-df.test$Y.train

accuracy<-table(Reference=Y.test,Prediction=Ypred)
xtable(accuracy)

############################## Simulate SMA ######################
```

```
n<-8
DJ<-read.zoo("djia.txt",header=TRUE, FUN=as.yearmon)
DJ<-window(DJ, index. = index(DJ), start = as.character.Date("Jan 1981"), ...
    end = as.character.Date("Dec 2010"))
dates<-index(DJ)
data<-coredata(DJ)
mkt <- data[,"MKT"] # the total return
cap <- data[,"CAP"] # the capital gain return
rf <- data[,"RF"] # the risk-free rate of return

ws <- ws.lma(n)
r.sma <- sim_tfrule(cap, mkt, rf, ws, short=FALSE)
nobs <- length(mkt)
m <- length(ws)
r.sma <- r.sma[(m+1):nobs]
mkt <- mkt[(m+1):nobs]
rf <- rf[(m+1):nobs]
dates <- dates[(m+1):nobs]

######################## Simulate NN ######################################
DJ <- read.zoo("djia.txt", header=TRUE, FUN=as.yearmon)
DJ<-window(DJ, index. = index(DJ), start = as.character.Date("Jan 1981"), ...
    end = as.character.Date("Dec 2010"))
# get the dates and data
dates <- index(DJ)    # get dates
dates<-dates[-c(1:n)]
data <- coredata(DJ) # get data
mkt <- data[,"MKT"] # the total return
mkt<-mkt[-c(1:n)]
cap <- data[,"CAP"] # the capital gain return
cap<-cap[-c(1:n)]
rf <- data[,"RF"] # the risk-free rate of return
rf<-rf[-c(1:n)]

NN_returns<-sim_nn(cap,mkt,rf,short=FALSE)
r.nn <- NN_returns
```

```
#Compute the cumulative returns to NN strategy and BH strategy
ind.r   <- cumprod(1 + r.nn)
ind.mkt <- cumprod(1 + mkt)
ind.sma <- cumprod(1 + r.sma)

ind <- cbind(ind.mkt,ind.r,ind.sma)
ind <- zoo(ind, order.by = dates)
plot(log(ind))



plot(dates,log(ind.mkt),type="l", main = "Buy and hold VS Neural network",
     ylab = "return")
lines(dates,log(ind.r),type="l",col=("red"))
lines(dates,log(ind.sma), type="l",col=("blue"))
legend("topleft",legend=c("Buy and Hold","Long only ...
   NN","SMA(8)"),col=c("black","red","Blue"),lty=1,bty = "n")

mktmrf <- mkt - rf #Excess return to the BH strategy
retmrf <- r.nn - rf #Excess return to the NN long only strategy

#Descriptive statistics of NN and BH strategy
mean.mkt <- mean(mkt)*12*100
mean.sma <- mean(r.sma)*12*100
mean.nn <-mean(r.nn)*12*100
sd.mkt <- sd(mkt)*sqrt(12)*100
sd.sma <- sd(r.sma)*sqrt(12)*100
sd.nn <- sd(r.nn)*sqrt(12)*100
SR.mkt <- Sharpe(mkt-rf)
SR.sma <- Sharpe(r.sma-rf)
SR.nn <- Sharpe(r.nn-rf)
pval <- SharpeTest(mkt-rf, r.sma-rf)$pval
pval

########################## Alpha ################################
mktmrf <- mkt - rf #Excess return to the BH strategy
retmrf <- r.nn - rf #Excess return to the NN long only strategy
```

```
Alpha.nn<-AlphaTest(mktmrf,retmrf)
alpha.nn<-(Alpha.nn$alpha)*100*12
pval.alpha.nn<-Alpha.nn$pval

mktmrf <- mkt - rf #Excess return to the BH strategy
retmrf <- r.sma- rf #Excess return to the NN long only strategy

Alpha.sma<-AlphaTest(mktmrf,retmrf)
alpha.sma<-(Alpha.sma$alpha)*100*12
pval.alpha.sma<-Alpha.sma$pval

#==================Bootstrap=========================
mktmrf <- mkt - rf #Excess return to the BH strategy
retmrf <- r.nn - rf #Excess return to the NN strategy

srdiff <- function(data)  {
  # this function computes the test statistics for each bootstrapped ...
     resample of data
  mktmrf <- data[,1] # vector that contains the excess returns to the market
  retmrf <- data[,2] # vector that contains the excess returns to the nn ...
     strategy
  # compute the f statistics
  f <- Sharpe(retmrf) - Sharpe(mktmrf)
  return(f)
}

fstat <- Sharpe(retmrf) - Sharpe(mktmrf)
data <- cbind(mktmrf, retmrf)

bLen = 5
R <- 5000
# bootstrap time series
results <- tsboot(tseries=data, statistic=srdiff,
                  R=R, l=bLen, sim="geom")


# The result is a matrix that contains bootstrapped f-statistics
```

```
fstat.sim <- results$t - fstat
# compute the p-value of the test
pval.boot.sr <- length(which(fstat.sim > fstat))/R
pval.boot.sr

################################ SMA vs NN ...
    ##########################################
smamrf <- r.sma - rf #Excess return to the BH strategy
retmrf <- r.nn - rf #Excess return to the NN long only strategy

srdiff <- function(data)  {
  # this function computes the test statistics for each bootstrapped ...
     resample of data
  smamrf <- data[,1] # vector that contains the excess returns to the market
  retmrf <- data[,2] # vector that contains the excess returns to the nn ...
     strategy
  # compute the f statistics
  f <- Sharpe(retmrf) - Sharpe(smamrf)
  return(f)
}

fstat <- Sharpe(retmrf) - Sharpe(smamrf)
data <- cbind(smamrf, retmrf)

bLen = 5
R <- 5000
# bootstrap time series
results <- tsboot(tseries=data, statistic=srdiff,
                  R=R, l=bLen, sim="geom")

# The result is a matrix that contains bootstrapped f-statistics
fstat.sim <- results$t - fstat
# compute the p-value of the test
pval.boot.nnsma <- length(which(fstat.sim > fstat))/R
pval.boot.nnsma

############################### Newey west ################################
```

```
retmrf<-r.nn-rf
mktmrf<-mkt-rf

reslm <- lm(retmrf ~ mktmrf)
res <- coeftest(reslm, df = Inf, vcov = NeweyWest(reslm, lag = 12, ...
    prewhite = FALSE))
res

smamrf<-r.sma-rf
mktmrf<-mkt-rf
reslm <- lm(smamrf ~ mktmrf)
res <- coeftest(reslm, df = Inf, vcov = NeweyWest(reslm, lag = 12, ...
    prewhite = FALSE))
res



df <- data.frame(BH=c(mean.mkt, sd.mkt, SR.mkt, NA,NA,NA),
                 SMA=c(mean.sma, sd.sma, SR.sma, ...
                     pval.boot.sma,alpha.sma,pval.alpha.sma),
                 NN=c(mean.nn,sd.nn,SR.nn,pval.boot.sr,alpha.nn,pval.alpha.nn))
rownames(df) <- c("Mean return", "Std. deviation", "Sharpe ratio", ...
    "P-value","Alpha","P-value alpha")
xtable(df)
```

Appendix C

Discussion Paper

C.1 Jeffrey Benjamin Charles

Introduksjon

Dette diskusjons papiret reflekterer over hvorvidt masteroppgaven vår er relevant i forhold til konseptet «responsible» (ansvarlighet), og hvordan vi sammen har skrevet, jobbet og diskutert oppgaven. Prosessen har vært lang, krevende, og vi har gang på gang møtt på motstand i form av ulike problemstillinger knyttet til det teoretiske samt det tekniske aspektet ved oppgaven. Samtidig har det også vært meget spennende å skrive om teamet maskinlæring i finans, og dette har gitt oss tverrfaglig kunnskap og dybde i et fagområde som er ansett som veldig attraktiv og etterspurt i dagens arbeidsmarked. For det første, er masteroppgaven vår relevant til konseptet «responsiable» ved at studiet er utført på globale indiser som Standard Poor 500 og Dow Jones Industrial. Det forventes av aksjonærene og samfunnet at selskapene på disse indeksene utøver tilstrekkelig ansvarlighet på alle aspekter. For det andre, er maskinlæring en form for kunstig intelligens som har som oppgave å lære seg selv opp ved hjelp av eksisterende data. Det er derfor spesielt viktig at man opptrer ansvarlig ved bruk at denne formen for teknologi og ikke stoler blindt på resultatet da dette kan ha konsekvenser for de involverte. I dette diskusjons papiret presenteres først et kort sammendrag av masteroppgaven, deretter defineres konseptet «ansvarlighet, så diskuteres hvordan oppgaven forholder seg til dette konseptet, og til slutt en konklusjon. For det tredje er oppgaven relevant til konseptet på grunn av alt maskinlæring kan brukes til. I vår oppgave har vi spesifikt tatt for oss det finansielle aspektet, men det er verdt å nevne at maskinlæring

brukes i så mye annet i mange forskjellige industrier. Det er derfor ekstremt viktig at man er ansvarlig ved bruk av dette.

Presentasjon av masteroppgaven

Den effektive markedshypotesen er en velkjent finansiell hypotese. Ifølge ideen, en sikkerhet alltid "fullstendig reflekterer" all tilgjengelig informasjon i markedet (Fama, 1970). I følge den svake formeffektive markedsteorien er det vanskelig for investorer å kontinuerlig overgå markedet ved å bruke tidligere data. Til tross for dette prøver investorer å utkonkurrere markedet ved å bruke tidligere data fra den underliggende eiendelen. På grunn av kompleksiteten til aksjemarkedsdata er det ekstremt vanskelig å utvikle effektive prognosemodeller (Kara et al., 2011).

"Kunstig intelligens brukt gjennom maskinlæring vil være en av de mest transformerende teknologiene i vår generasjon, som håndterer noen av menneskehetens mest utfordrende problemer, øker menneskelig ytelse og maksimerer produktiviteten. Ansvarlig bruk av disse teknologiene er nøkkelen til å fremme fortsatt innovasjon" - (Amazon, 2022).

Maskinlæring er et underfelt av kunstig intelligens som omfatter mange algoritmer og metoder. Maskinlæringsalgoritmer innen finans, som konvensjonell teknisk analyse, streber etter å finne underliggende mønstre i prisatferd. Teknisk analyse er basert på forestillingen om at visse mønstre eksisterer i finansiell prising. Som et resultat er teknisk analyse designet for å utnytte disse trendene, som den menneskelige hjernen ofte kan gjenkjenne. De ulike maskinlæringsmetodene har kapasitet til å kartlegge kontinuerlige ikke-lineære funksjoner ved bruk av tidligere prisdata. Dette gjør dem i stand til å identifisere mønstre som det menneskelige øyet ikke kan se. (Zakamulin, 2021a). I vår masteroppgave har vi valgt å ta for oss nettopp dette spennende dagsaktuelle temaet. Vi har som motivasjon å utforske om vi ved å benytte oss av maskinlæring, som benytter historisk data, klarer å lage en aktiv strategi som er i stand til å prestere likt eller bedre enn simple moving average og den tradisjonelle kjøp-og-hold strategien. Isteden for å predikere avkastning har vi valgt å se predikere oppgang og nedgang i markedet. Med andre ord, ønsker vi å se om vår strategi basert på «neural network» er i stand til å predikere opptrender og nedtrender i markedet, og om ved at vi handler

på signalene som gis fra modellen, klarer å prestere likt eller bedre enn markedet. I oppgaven valgte vi å se på to indekser i USA: SP Composite og Dow Jones Index. Den eksisterende litteraturen i dette emnet er sterkt fokusert på maskinlæringssiden, med en tydelig mangel på det økonomiske aspektet. Som et resultat har vi bestemt oss for å inkludere statistiske tester som Sharp ratio-testen og Alpha-testen for å se om den aktive strategien basert på "neural nettverk" er statistisk signifikant sammenlignet med det simple moving average og kjøp-og-hold. strategi. Algoritmen vår var i stand til å forutse opp- og nedturer i markedet med et høyt nivå av nøyaktighet som kan sammenlignes med tidligere studier. Sammenlignet med kjøp-og-hold-strategien ga en trendfølgende handelsstrategi basert på resultatene av modellen vår et akseptabelt Sharpe-forhold og en positiv alfaverdi for SP Composite. Disse funnene var imidlertid ikke banebrytende fordi evaluering av premissene for lignende ytelse som en kjøp-og-hold-strategi ikke ga noen statistisk signifikante resultater. For Dow Jones-indeksen presterte modellen vår dårlig, noe som resulterte i et redusert Sharpe-forhold for alle perioder og en negativ alfaverdi i to av tre perioder. Til slutt vil vi vise til studien av Israel et al. (2020) som mener at maskinlæring, spesielt "neural network", bør brukes som et verktøy for å supplere eksisterende økonomiske modeller i stedet for som en erstatning. Dette skyldes det faktum at funnene våre ikke er representert i den eksisterende litteraturen som indikerer at maskinlæring har høy grad av nøyaktighet og at det er lettere å forutse hvordan en aksje eller indeks vil prestere over tid.

Konseptet «Responsiability» (Ansvarlighet)

For å kunne drøfte masteroppgaven i lys av konseptet «Ansvarlighet» er vi først nødt til å definere dette konseptet. Vedantu (2022) definerer ansvarlighet som

«Ansvarlighet er en moralsk forpliktelse å fullføre det tildelte arbeidet. Med andre ord er det også definert som "Ansvar er en persons forpliktelse til å utføre tildelte aktiviteter etter beste evne".

Det å handle ansvarlig er noe som er i stort fokus i dagens samfunn. Bedrifter og intuisjoner opplever et sosialt press fra samfunnet om å handle etisk og moralsk ovenfor miljøet og samfunnet. Tidligere har mye av fokuset blant bedrifter vært å maksimere profitt uten å tenke på konsekvensen av dette, mens i nyere tid har dette tatt et skifte og bedrifter er mer bundet til å søke velferden til ulike deler av samfunnet og miljøet. Det er viktig å kunne anlegge et globalt perspektiv som omfatter hele verdens befolkning og fremtidige

generasjoner.

"ESG (Enviromantal, Social Governence) er et konsept som ofte blir bundet til det å handle ansvarlig. En litteraturgjennomgang av Liang and Renneboog (2020) forklarer at ved å ta investorperspektivet kan man forstå hvordan ESG og bærekraftig finans påvirker aktiva priser og porteføljeavkastning. PRI (The Principles of Responsible Investing), også kjent som det største globale nettverket av institusjonelle investorer som er forpliktet til å vurdere ESG-spørsmål i investeringene sine, oppmuntrer investorer til å investere ansvarlig for å øke avkastningen og bedre risikostyring (UNPRI, 2021). I 2016 introduserte Moriningstar ESG-rating, og fond med høy ESG-rating observerte positiv kontantstrøm mens fond med lavere ESG-rating hadde negativ kontantstrøm (Hartzmark & Sussman, 2019). ESG-informasjon har blitt viktigere i investeringsprosessen på grunn av det økende antallet ansvarlige investorer som vurderer ESG-informasjon i sine investeringsallokeringer (Amel-Zadeh & Serafeim, 2018). Konseptet «ansvarlighet» er bredt og kan derfor anvendes i ulike området innenfor finans. Videre skal jeg drøfte hvorvidt dette konseptet er relevant" - (Charles & Larsen, 2021, p.6).

Drøfting av «Ansvarlighet» i lys av masteroppgaven

I denne delen skal etiske utfordringer i lys av ansvarlighet knyttet til masteroppgaven diskuteres. Maskinlæring har mange suksesshistorier innenfor en rekke forskningsfelt. Gjenkjenne bilder og stemme, automatisering av kjøretøy, robotikk og strategisk spilling er noen av tingene maskinlæring har utmerket seg på. Maskinlæring er blitt mer og mer anvendt i finansbransjen ettersom denne teknologien innehar evnen til å oppdage mønstre som ikke er synlig for det menneskelige øyet samt løse komplekse problemstillinger. Maskinlæring fører til at mange arbeidsoppgaver både i og utenfor finans som drives av mennesker kan bli tatt over av denne teknologien. Det kan argumenteres for at teknologien i seg selv skaper arbeidsplasser, men når denne teknologien utvikles og blir tilstrekkelig nok, vil det bli mer lønnsomt for bedrifter å ta i bruk dette fremfor menneskelig arbeidskraft. Dette kan da føre til at det blir lavere arbeidstilbud på markedet. Dette reiser derfor en rekke spørsmål om hvorvidt det er ansvarlig å benytte seg av en slik teknologi som er på bekostning av arbeidskraft. Det kan tenkes at frem i tid så kan teknologien ha utviklet seg såpass at maskinlæring har evnen til å hente inn løpende finansielle data, og bruke dette til å ta aktive investeringsbeslutninger.

Det kan da tenkes at den kunstige intelligensen ikke nødvendigvis tar i betraktning hvilke selskaper det investeres i, men kun tenker på å maksimere profitt. Dette kan føre til at det investeres i selskaper som ikke er særlig bærekraftige og scorer lavt på ESG målinger. Det kan være at algoritmen beslutter å investere i våpenindustri og tobakksindustrien (to industrier som er boikottet fra de fleste verdens fond), og tar beslutninger som ikke er betenkt i et etisk og morals perspektiv. Ved bruk av maskinlæring er det viktig å trå ansvarlig. Det er viktig at brukeres tar ansvar for algoritmiske beslutninger, rettferdighet av utfall, åpenhet av beslutningene og hvordan man kommer frem til de, og muliggjøre byrå og algoritmiske valg. Under masterskrivingen har vi vært nødt til å ha en ansvarlig forskningsoppførsel. Dette innebærer at vi tar hensyn og respekterer andre forskers arbeid, ved å referere til deres arbeid på riktig måte. Det er også viktig at vi fremlegger resultatene og fremgangsmåten vår på en forståelig måte slik at leser kan forså måten vi har tenkt på. Vi har også vært ansvarlig i valg av metode i masteroppgaven vår. Det betyr at vi har benyttet oss av en metode hvor det er utført mye studier på, som er med på å erkjenne og godta metoden som en forsvarlig fremgangsmåte av eksisterende litteratur. Dette betyr at metoden vår støttes av sterk teori. Vi har hatt et ansvar overfor oss selv om å utføre en tilstrekkelig og gjennomtenkt jobb under skriveperioden av oppgaven. Vi har med andre ord måtte opptre ansvarlig i hele prosessen av masterskrivingen for å kunne levere en tilstrekkelig god masteroppgave etter de standardene vi har satt oss selv.

Konklusjon

Som diskutert i dette refleksjonsnotatet, forholder denne oppgaven seg til konseptet; ansvarlighet. Oppgaven har som diskutert dette konseptet på ulike områder, både i forhold til problemstillingen vår, måten vi har arbeidet med oppgaven på, og eventuelle problemstillinger knyttet til teamet. Det har vært en lærerik prosess og det føles som jeg har gjort det fått litt mer innsikt og reflektert over viktige temaer som jeg håper vil være nyttige når man går ut i arbeidslivet. Til slutt vil jeg uttrykke min takknemlighet til hele fakultetet ved UiA. Mer spesifikt ønsker jeg å takke alle foreleserne jeg har hatt gjennom mitt studieløp. De har veiledet og bidratt godt til at jeg har klart å tilegne mye nytt kompetanse, som jeg mener at jeg har et ansvar for å bruke på riktig måte både etisk og moralsk. Denne oppgaven tillot meg å demonstrere mine evner, presse meg selv og oppdage nye egenskaper. Det har også

vært en glede å jobbe med min partner Daniel Meselu, og jeg vil se tilbake på mine fem år ved UiA med stolthet og glede.

C.2 Daniel Debbasu Meselu

Discussion paper – International

This discussion paper is written as a mandatory part of the master's program in business and administration with specialization in analytical finance. This thesis was written together with Jeffrey Benjamin Charles, which under the whole process has been an encouraging and motivating writing partner. To get this master thesis approved, there has been a couple of requirements. Such as a mandatory meeting with supervisors, an oral presentation of our thesis to fellow co-students and writing a discussion paper. The purpose of this discussion paper is to shed light on the process we've gone through when writing this thesis, as well the knowledge we have accumulated during this two-year master's program. This paper should therefore give students an opportunity to reflect upon important subjects before graduating.

I am among the students who will discuss the topic "international" considering our thesis and the master's program. Specifically, how our topics, research questions, and findings can be influenced by international trends and forces and how actors may react to these trends. Our thesis is written within the topic of machine learning. The choice of topic came naturally after having the introductory course in machine learning the semester prior to the master thesis. This offered the opportunity to write about something innovating and fresh, with few similar studies. It has been a daunting task and we have faced many challenges, but with the help of our supervisors, we managed to overcome said challenges.

Despite the presence of the efficient market hypothesis, introduced by Fama (1970), investors constantly try to beat the market by trading actively, oppositely to what the efficient market hypothesis suggests. The efficient market hypothesis states that historical prices of an asset do not carry any information that can be used to predict the future price, and that the movement of prices therefore is random. As a result, investors should not be able to consistently beat the market using purely historical data of the underlying asset. This justifies a passive buy and hold strategy. The buy-and-hold strategy is a passive long-term investing strategy where investors will actively select a stock to buy and hold over a long-time horizon without worrying about short-term price fluctuations. Our thesis seeks to check how an

active trading strategy based on the forecasts of a neural network model performs against the passive buy-and-hold strategy. As a robustness check, we also compare the performance of a standard trend following strategy based on moving averages. Traditional forecasting methods to generate trading signals are often characterized by linearity. Machine learning, being a branch of artificial intelligence has given many success stories in fields, such as robotics, voice recognition, image recognition and much more. With this came the idea to apply it to financial data to find patterns in the data invisible to the human eye, and to exploit these patterns to achieve greater profits. Financial data however tends to be noisy and have little to no structure, which strengthens the efficient market hypothesis. Thus, there is a need to adjust the data to facilitate the use of machine learning algorithms. Several authors have attempted to apply neural networks to financial time series. The results are often characterized by high accuracy in forecasts and following trading strategies indicating superior performance. We quickly found these studies to lack the statistical depth required to evaluate such a trading strategy. Therefore, we wanted to build a neural network to predict the future market state, build a trading strategy and properly evaluate it in comparison to pre-determined benchmark strategies.

Motivated by this, we used the historical returns on two major stock market indices in the to predict the market state, that is if the market would be in a bull state or a bear state. A bull market is characterized by prices rising, whereas a bear market is the opposite where prices fall. The idea was to predict the future market sign such that we would know if it would be a bull or bear and thus invest in the market whenever the prices were on the rise and sell to invest at a risk-free interest rate whenever the prices were on the downfall. Neural network is a common name for algorithms where information is passed on through subsequent layers of non-linear function, allowing for non-linear mapping of functions (Dixon et al., 2020). These kinds of algorithms have been used in many scientific fields, where some of the results are astonishing. Picture recognition, voice recognition and robotics are some of the fields where machines learning has excelled. Our results are promising in the sense that our neural network strategy performs competitively to a simple moving average strategy. The simplicity of our network configuration leaves room for improvements and lays a path for future research.

Although our approach seems simple in the grad scheme of neural networks, we still found it challenging to construct a meaningful forecaster. One of the main reasons for our struggles was the lack of precise guidance in the construction of neural network. (Bishop & Nasrabadi, 2006; Géron, 2019; Heaton, 2008) all guided us on the right path, but they all focused on other tasks than financial modeling. Thus, we were ultimately left to ourselves to test many configurations before settling with an appropriate result. This experience really brought up the innovative side of the master's program.

International trends and forces can be though of as a general development or direction of a phenomenon across national borders. The topic of machine learning is highly present at the international level. In fact, developing smart algorithms to automate procedures has been a key factor to gain competitive advantage in numerous business fields (Attaran & Deb, 2018). An increasing number of everyday functions are being automated. These are often overlooked as everyday features but are in fact the result of machine learning algorithms. To set thing in perspective, one of the most useful tools, the spam filter is in fact just a classification algorithm trained to detect important email and separate it from spam. The advertisements we see on websites, which coincidentally happen to be something we've been looking for is also the result of a machine learning algorithm. With these success stories, machine learning has proven to be a powerful tool that can change the ways we do things completely. The results of our thesis showed that a neural network has the capacity to model a financial time series, given enough data points and sufficient structure in the data. The simplicity of the network architecture also motivates the use of more complex networks to model time series data and truly study if the prices and returns of the stock market are purely random. Our result gave similar results to other forecasting tools in terms of statistical significance, relative to the market portfolio. It should be considered as a tool to enhance already existing forecasting techniques, rather than a replacement.

I would also like to mention other aspects of the master's program where "international" has been a core theme. Firstly, we had a subject called sustainable capitalism in the first year of the program. This subject was divided into two, with the first part consisting of lectures and oral presentations and the last part consisting of individual work on a term paper discussing

environmental issues. Here we learned the importance of international collaboration to reduce the amount of pollution in our globe. The problem is deemed to be an international problem, as it threatens all life of earth equally. Consequently, international collaboration is also needed to overcome these challenges. The term paper also reviewed an interesting topic. One of the questions to be answered was if "green growth" was possible, meaning a capitalistic growth while remaining pollution free. One of our key findings indicated that the focus should be directed towards the green instead of growth. Additionally, suddenly reducing the use of fossil fuels would have ramification on developing countries, which could not be justified given most of developed countries used such fuels to build their wealth.

Another subject to frequently bring up the topic of international trends was supply chain management. This course quickly highlighted the transition of the supply chain term and highlighted that it no longer refers to the core business, but through the whole operation on an international level (Fujita & Hamaguchi, 2016). Businesses who quickly exploited the advantages of an international supply chain quickly gained a competitive advantage. The course was held at the same time as the outbreak of the COVID-19 virus. The virus quickly became a global pandemic, and the consequences were detrimental for many businesses. The trends and forces that developed because of the pandemic were of an international scale. As boarders closed, we quickly began to see how much businesses struggled to meet their demands. The adaptation of the supply chain to meet demands was shocking and showed how vital international cooperation had become.

To summarize, this thesis explores the use of machine learning in finance. Specifically, the use of neural networks to predict price movements in the market. Our findings are encouraging and show that simple configurations of neural networks have the capacity to perform similarly to traditional forecasting methods. As far as international trends and forces go, machine learning is a scientific discipline that has been booming in the last decade. It has quickly become one of the most important tools to gain competitive advantage and has replaced many automated jobs (Attaran & Deb, 2018). The school of business administration and law at the University of Adger has neatly incorporated the topic of international into the courses of this master's program. Most notable, we have seen the importance of interna-

tional cooperation to overcome the challenges we are facing regarding the environment. The outbreak of COVID-19 also showed how much businesses rely on international cooperation in their supply chain. The closing of borders resulted in huge deficiencies that would take a long time to recover from.

And finally, I would like to thank the School of Business and Law at the University of Agder and give them credit for the last five years of academic, social, and personal development I have experienced. The courses have been meaningful, and I feel well armed with knowledge to step into the working life, and I have no doubt the knowledge I have accumulated during these five years will help me in my future endeavors. There has been highs and lows during my stay, but at the core of my memory, it is a time I will look back at with pride and joy.

Bibliography

Aamodt, T. (2015). Predicting stock markets with neural networks (master's thesis).

Addai, S. (2016). *Financial forecasting using machine learning* (Doctoral dissertation). University of Cape Town.

Amazon. (2022). *Responsible use of artificial intelligence and machine learning*. https://aws.amazon.com/machine-learning/responsible-machine-learning/

Amel-Zadeh, A., & Serafeim, G. (2018). Why and how investors use esg information: Evidence from a global survey. *Financial Analysts Journal*, *74*(3), 87–103.

Attaran, M., & Deb, P. (2018). Machine learning: The new'big thing'for competitive advantage. *International Journal of Knowledge Engineering and Data Mining*, *5*(4), 277–305.

Bishop, C. M., & Nasrabadi, N. M. (2006). *Pattern recognition and machine learning* (Vol. 4). Springer.

Brockwell, P., & Lindner, A. (2012). Lévy-driven time series models for financial data. *Handbook of Statistics*, *30*, 543–563.

Brooks, C. (2019). *Introductory econometrics for finance (vol. 4th edition)*. Cambridge, UK: Cambridge University Press.

Bry, G., & Boschan, C. (1971). *Cyclical analysis of time series: Selected procedures and computer programs*. NBER.

Cansiz. (2020). *The math behind training of a neural network*. Retrieved May 5, 2022, from https://towardsdatascience.com/adventure-of-the-neurons-theory-behind-the-neural-networks-5d19c594ca16

Charles, J., & Larsen, E. (2021). How does esg create value for stakeholders? (term paper in be-422-1 21h environmental, social and governance (esg) metrics: Reshaping finance.

Chen, A.-S., Leung, M. T., & Daouk, H. (2003). Application of neural networks to an emerging financial market: Forecasting and trading the taiwan stock index. *Computers & Operations Research*, *30*(6), 901–923.

Dixon, M. F., Halperin, I., & Bilokon, P. (2020). *Machine learning in finance* (Vol. 1170). Springer.

Episcopos, A., & Davis, J. (1996). Predicting returns on canadian exchange rates with artificial neural networks and egarch-m models. *Neural Computing & Applications*, *4*(3), 168–174.

Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *The journal of Finance*, *25*(2), 383–417.

Fujita, M., & Hamaguchi, N. (2016). Supply chain internationalization in east asia: Inclusiveness and risks. *Papers in Regional Science*, *95*(1), 81–100.

Géron, A. (2019). *Hands-on machine learning with scikit-learn, keras, and tensorflow: Concepts, tools, and techniques to build intelligent systems*. " O'Reilly Media, Inc."

Hanusch, H., & Pyka, A. (2007). *Elgar companion to neo-schumpeterian economics*. Edward Elgar Publishing.

Hartzmark, S. M., & Sussman, A. B. (2019). Do investors value sustainability? a natural experiment examining ranking and fund flows. *The Journal of Finance*, *74*(6), 2789–2837.

Heaton, J. (2008). *Introduction to neural networks with java*. Heaton Research, Inc.

Israel, R., Kelly, B. T., & Moskowitz, T. J. (2020). Can machines' learn'finance? *Journal of Investment Management*.

Jobson, J. D., & Korkie, B. M. (1981). Performance hypothesis testing with the sharpe and treynor measures. *Journal of Finance*, 889–908.

Jungeilges, J. (2022a). Lecture notes in empirical finance. unit 6: Inference concerning autocovariance/orrelation.

Jungeilges, J. (2022b). Lecture notes in machine learning with applications in finance (se-507-1), unit 4.

Kadan, O., & Liu, F. (2014). Performance evaluation with high moments and disaster risk. *Journal of Financial Economics*, *113*(1), 131–155.

Kanas, A., & Yannopoulos, A. (2001). Comparing linear and nonlinear forecasts for stock returns. *International Review of Economics & Finance*, *10*(4), 383–398.

Kara, Y., Boyacioglu, M. A., & Baykan, Ö. K. (2011). Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the istanbul stock exchange. *Expert systems with Applications*, *38*(5), 5311–5319.

Liang, H., & Renneboog, L. (2020). Corporate social responsibility and sustainable finance: A review of the literature. *European Corporate Governance Institute–Finance Working Paper*, (701).

Ljung, G. M., & Box, G. E. P. (1978). On a measure of lack of fit in time series models. *Biometrika*, *65*(2), 297–303.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115–133.

Minsky, M., & Papert, S. (1969). Perceptrons.

Mostafa, M. M. (2010). Forecasting stock exchange movements using neural networks: Empirical evidence from kuwait. *Expert systems with applications*, *37*(9), 6302–6309.

Newey, W. K., & West, K. D. (1986). A simple, positive semi-definite, heteroskedasticity and autocorrelationconsistent covariance matrix.

Olson, D., & Mossman, C. (2003). Neural network forecasts of canadian stock returns using accounting ratios. *International Journal of Forecasting*, *19*(3), 453–465.

Pagan, A. R., & Sossounov, K. A. (2003). A simple framework for analysing bull and bear markets. *Journal of applied econometrics*, *18*(1), 23–46.

Pan, H., Tilakaratne, C., & Yearwood, J. (2005). Predicting australian stock market index using neural networks exploiting dynamical swings and intermarket influences. *Journal of research and practice in information technology*, *37*(1), 43–55.

Pany, P. K., & Ghoshal, S. P. (2015). Dynamic electricity price forecasting using local linear wavelet neural network. *Neural Computing and Applications*, *26*(8), 2039–2047.

Qi, M. (1999). Nonlinear predictability of stock returns using financial and economic variables. *Journal of Business & Economic Statistics*, *17*(4), 419–429.

Rosenblatt, F. (1958). Perceptron simulation experiments. *Proceedings of the IRE*, *48*(3), 301–309.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). *Learning internal representations by error propagation* (tech. rep.). California Univ San Diego La Jolla Inst for Cognitive Science.

Tealab, A., Hefny, H., & Badr, A. (2017). Forecasting of nonlinear time series using ann. *Future Computing and Informatics Journal*, *2*(1), 39–47.

UNPRI. (2021). *About the pri*. https://www.unpri.org/pri/about-the-pri

Vedantu. (2022). *Concept of responsibility*. https://www.vedantu.com/commerce/concept-of-responsibility

Wang, C., Zhang, X., Wang, M., Lim, M. K., & Ghadimi, P. (2019). Predictive analytics of the copper spot price by utilizing complex network and artificial neural network techniques. *Resources Policy*, *63*, 101414.

Welch, I., & Goyal, A. (2008). A comprehensive look at the empirical performance of equity premium prediction. *The Review of Financial Studies*, *21*(4), 1455–1508.

Wikipedia. (2008). Diagram of neuron with arrows but no labels [[Online; accessed June 10, 2022]]. https://commons.wikimedia.org/wiki/File:Derived_Neuron_schema_with_no_labels.svg?fbclid=IwAR0VT8GbSWCyGiR7aDv6kkRwnvl5g0EAyFYsxQBeW32BDyJ1pmE

Womack, K. L., & Zhang, Y. (2003). Understanding risk and return, the capm, and the fama-french three-factor model. *Available at SSRN 481881*.

Yildirim, I., Ozsahin, S., & Akyuz, K. C. (2011). Prediction of the financial return of the paper sector with artificial neural networks. *BioResources*, *6*(4), 4076–4091.

Yildiz, B., Yalama, A., & Coskun, M. (2008). Forecasting the istanbul stock exchange national 100 index using an artificial neural network. *An International Journal of Science, Engineering and Technology*, *46*, 36–39.

Zakamouline, V., & Koekebakker, S. (2009). Portfolio performance evaluation with generalized sharpe ratios: Beyond the mean and variance. *Journal of Banking Finance*, *33*(7), 1242–1254.

Zakamulin, V. (2017a). Introduction to the bbdetection package.

Zakamulin, V. (2021b). Lecture notes in se-507 machine learning with applications in finance, unit 8: Bootstrap methods in finance.

Zakamulin, V. (2017b). *Market timing with moving averages: The anatomy and performance of trading rules*. Springer.

Zakamulin, V. (2021a). Lecture notes in se-507 machine learning with applications in finance, unit 7: Trend-following rules.

Zhang, Z. (2018). Artificial neural network. *Multivariate time series analysis in climate and environmental research* (pp. 1–35). Springer.