MASTER'S THESIS

Course code: BE305E

Name: Einar Solberg

Machine Learning in Portfolio Construction—Empirical Evidence from the Norwegian Capital Market

---

Date: 25.5.2021

Total number of pages: 65

NORD University

www.nord.no

Sammendrag

Denne oppgaven undersøker gjennomførbarheten av å bruke maskinlæringsmetoder til porteføljekonstruksjon i det norske kapitalmarkedet. Jeg bruker maskinlæringsmetodene Long Short Term Memory networks og random forest til å konstruere aksjeporteføljer av forskjellige størrelser som måles opp mot hverandre for å finne den optimale kombinasjonen av porteføljestørrelse og maskinlæringsmetode.

Jeg utforsker også den fremtidige gjennomførbarheten av å bruke maskinlæringsmetoder i porteføljekonstruksjon ved å undersøke utviklingen av treffsikkerheten og avkastningen til de forskjellige metodene. Resultatene viser at både treffsikkerhet og avkastning er avtagende, og har vært avtagende i majoriteten av perioden 1.1.2009-26.11.2020. Dette er i henhold til resultatene av liknende forskning fra utenlandske markeder, der man har oppdaget en liknende nedgang i avkastning. Denne nedgangen har dog inntruffet tidligere i mer modne markeder, eksempelvis det amerikanske.

Når det kommer til den nåværende statusen til porteføljekonstruksjon med maskinlæring, har Long Short Term Memory nettverket den høyeste avkastningen, med en årlig Sharpe ratio på 5.69 etter transaksjonskostnader, samt en årlig information ratio med OSEBX som referanseindeks på 5.20 etter transaksjonskostnader.

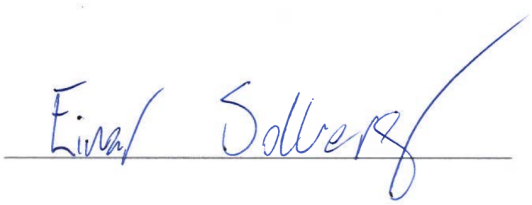
Jeg kjører en multippel regresjonsanalyse på avkastningsdataen fra Long Short Term Memory nettverket og random forest modellen, der jeg benytter meg av flere vanlige kilder til systematisk risiko som uavhengige variabler. Resultatet av den analysen viser at ingen av metodene har en høyt signifikant eksponering til disse faktorene, og klarer derfor å generere høye alfa-verdier i det norske kapitalmarkedet.

Preface

This thesis is written as a final part of my MSc in Business degree with the specialization Finance and Investment at Nord University Business School. Writing this thesis has been an exciting challenge from start to finish, and has given me ample opportunities to dive deeper into a variety of topics ranging from the foundations of portfolio theory to state-of-the-art programming in R and Python.

I wish to thank my supervisor, Oleg Nenadic, for helpful insights into the world of machine learning and advice on writing. I also want to thank my family for their assistance in proofreading the thesis and their continued support throughout my entire education. Finally, I wish to thank my fellow students for giving me an unforgettable time here at Nord University.

Bodø, 25.5.2021



Einar Solberg

Abstract

In this thesis, I examine the feasibility of using machine learning methods in the task of portfolio construction in the Norwegian capital market. I use Long Short Term Memory networks and random forests to construct stock portfolios of various sizes, which are measured against each other to determine the optimal method and portfolio size combination.

I also aim to shed light upon the future feasibility of machine learning in portfolio construction, by examining the development of accuracy and returns of the different methods. I find that both accuracy and overall returns are in decline, and have been in decline for the majority of the study period, which lasts from 1.1.2009 to 26.11.2020. This is consistent with similar studies from foreign markets, which have experienced a similar decline in profitability, although this has occurred sooner in more mature markets such as the U.S. market.

As for the current state of portfolio construction with machine learning, the Long Short Term Memory network achieves the highest returns of this thesis, with an annualized Sharpe ratio of 5.69 after transaction costs, and an annualized information ratio with the OSEBX as a benchmark of 5.20 after transaction costs.

Running a multiple regression analysis on the returns data from the Long Short Term Memory network and random forest, using several common sources of systematic risk as independent variables, I find that neither of the methods are exposed to these factors at a high significance level. Thus, both methods are able to generate high alphas in the Norwegian capital market.

Contents

List of Tables

List of Figures

1 Introduction

Using machine learning techniques in the field of stock return prediction is quickly becoming popular in academia, with several research articles being published on the subject, using several different methods and comparing them to see which one can produce the best return for a potential investor. Notable contributions within this field include Fischer & Krauss (2018), who constructed several models for stock return prediction (e.g. LSTM networks, random forest, logistic regression), and tested them against each other using data from the U.S. stock market. That paper and others will be examined in greater detail in the second chapter of this thesis.

The purpose of this thesis is to contribute to the existing literature within the field of machine learning in finance, more specifically stock return predictions, by examining state-of-the-art methods applied to the Norwegian stock market. Further, I will use the predicted returns to construct portfolios of different sizes and measure their performance against each other with a variety of financial benchmarks, making this thesis more than strictly theoretical, but also beneficial to financial practitioners.

This thesis also aims to shed light upon the feasibility of continued use of machine learning methods in stock return predictions, as several recent studies have reported declining profitability when deploying these methods, especially in mature financial markets like the U.S. market (Hsu et al., 2016).

Oslo børs has received relatively little attention from academia compared to other larger and more liquid markets, especially when it comes to stock return prediction. This implies that there may be a potential for significant returns on stocks using machine learning methods.

Therefore, the problem statement of this thesis is as follows: *Is it feasible to reliably predict returns in the Norwegian stock market using machine learning methods?*

This thesis will consist of six chapters, where in this first chapter I will present the problem statement and contribution of the thesis, while the second chapter will focus on existing theory. The second chapter also contains a brief introduction to the theory behind the machine learning methods used in this thesis, to give the reader as much context as possible.

In the third chapter, I will discuss the specifics of the data and software applied when writing the thesis, and in the fourth, I will elaborate on the different methods used for constructing the portfolios. In chapter five I will present my results and discuss my findings, before finally making my conclusion in chapter six.

2 Theory

In this chapter, I will introduce relevant literature on the topic of machine learning and return prediction, and give an introduction to the theory behind the machine learning methods I deploy in this thesis.

2.1 Fama (1970)

An immediate issue that arises when contemplating a stock market prediction method, is that of the efficient market hypothesis (EMH), written by Fama (1970), and published in the Journal of Finance. In the article, Fama presents evidence that capital markets are in fact efficient, and that stock prices reflect all information. This implies that an investor cannot consistently beat the market over a significant period.

Fama examines three forms of market efficiency, *weak form*, *semi-strong form*, and *strong form*. In the weak form, previous market prices cannot be used to predict future prices of a stock, as it suggests that all available information is already contained within the current prices, including the information given by the past prices of the stock. Therefore, in the weak form of efficiency, changes in prices can only come from new information relevant to the stock. This implies that certain methods such as technical analysis or momentum-based strategies are not useful in predicting future price movements.

In the semi-strong form of efficiency, the market is fast to absorb any new information about a stock and adjust the price thereafter, meaning that an investor has no chance to utilize this information to beat the market. An implication of this is that only private information unavailable to the public market may lead to an advantage and an opportunity to beat the market.

Finally, in the strong form, the present prices of stocks reflect all available information, both private and public, meaning that even investors with insider knowledge will not be able to beat the rest of the market. It should be noted that Fama himself considered this strong form to be best viewed as a benchmark against which deviations of market efficiency can be judged, as such an extreme model could not realistically be expected to be an exact description of

the world (Fama, 1970).

Since its publication in 1970, the EMH has become a central staple of modern financial theory but is not without criticism. Critics often point out that there are in fact investors that have been able to consistently beat the market over longer periods (e.g. Warren Buffett), and that large market crashes imply that prices of stocks can significantly deviate from their fair values as described in the EMH. The objective of this paper is to develop a method of predicting future returns of the Norwegian market with reasonable accuracy, this should not be possible according to the EMH.

2.2 Multi-Factor Models

In the results chapter of this thesis, I will present the different portfolios' exposure to systematic risk, by running a regression analysis with the returns of said portfolios as the dependent variable, and several factors as independent variables. In the following section, I will examine these factors and their backgrounds in the financial literature.

2.2.1 The CAPM

The Capital Asset Pricing Model (CAPM) is a staple of finance literature. Familiar to anyone in the field of finance, the CAPM is in many cases the first financial model a student is taught. First published by Sharpe (1964), and further expanded upon by Lintner (1965) and Mossin (1966).

The CAPM is based on several assumptions about individual behavior and market structure:

1. Individual behavior

- Investors are rational, mean-variance optimizers.
- Their common planning horizon is a single period.
- They all use identical input lists (homogeneous expectations), assumes all relevant information is publicly available.

2. Market structure

- All assets are publicly held and traded on public exchanges.
- Investors may borrow or lend at a common risk-free rate and take short positions on traded securities.
- No taxes or transaction costs. Under these assumptions, all investors will hold portfolios of identical variance (risk), and that in equilibrium, the market portfolio is the unique mean-variance efficient tangency portfolio (Bodie et al., 2018). This makes a passive strategy efficient according to the CAPM.

Mathematically, the CAPM can be expressed as:

$$E(r_i) = r_f + \beta_i[E(r_m) - r_f]$$

where $E(r_i)$ denotes the expected return of security i, r_f denotes the risk-free rate, and $E(r_m)$ is the expected return of the market portfolio. The beta coefficient (β_i) describes the relationship of the risk of security i and the market portfolio, and is expressed as:

$$\beta_i = \frac{Cov(r_i, r_m)}{\sigma_m^2}$$

Despite being considered a staple of financial theory, the CAPM is not without criticism, especially from academia. Many point to the assumption that all assets are publicly held and traded on public exchanges as particularly flawed. This would imply that the market portfolio included all risky assets in the entire economy, while in practice, investors cannot even observe all tradable assets, let alone account for those that are non-tradable. Thus, the theoretical market portfolio of the CAPM is not possible to observe in practice (Bodie et al., 2018).

To conclude, the single factor CAPM has long been considered outdated in financial academia, and much effort has been devoted to discovering other factors that may better explain security returns, factors like *SMB* and *HML* discovered by Fama & French (1992), which I will examine next.

2.2.2 Fama & French (1992)

The Cross-Section of Expected Stock Returns was published in The Journal of Finance in 1992. The goal of the paper was to identify the joint roles of several different variables, namely the market β, size, Earnings over Price (E/P), leverage, and book to market equity. These roles were evaluated based on cross-sectional returns of NYSE, AMEX, and NASDAQ stocks.

The data used in the paper are COMPUSTAT's annual industrial files of income statements, balance sheet data from the Center for Research in Security Prices (CRSP), and returns from the NYSE, AMEX, and NASDAQ (also from the CRSP).

The authors find that the older Sharpe-Lintner-Black model, in which the relationship between the market β and the average returns of the NYSE was a central element, was no longer there in the more recent period of 1960-1990 (the appendix of the paper also concludes that this relationship is weak for the 1941-1990 period as well).

Their main finding was that for the 1963 to 1990 period, size and book to market equity captured the cross-sectional variation in stock returns associated with *E/P*, book to market, and leverage, which became the foundation for what is known today as the Fama-French model. The Fama-French model consists of three factors, namely excess return on market, size (SMB), and book to market ratio (HML), and can be expressed through the following formula:

$$R_{it} - R_{ft} = \alpha_{it} + \beta_1(R_{Mt} - R_{ft}) + \beta_2 SMB + \beta_3 HML + \epsilon_{it}$$

This model has been highly influential in the world of finance, as it builds on the flawed *CAPM*, as well as the works of other academics in the field (e.g. Reinganum, Lakonishok, and Shapiro).

The authors state that the findings of the paper could be quite useful in e.g. portfolio formation and performance evaluation, where the investor in question is primarily interested in long-term average returns rather than short-term. What this means in practical terms, is that performance of different portfolios can be compared to the performance of a benchmark portfolio with similar values for *SMB* and *HML*. However, they also make a disclaimer that it is possible, though unlikely, that the *SMB* and *HML* variable described the cross-section of

average returns in their sample, but in reality, they were and are unrelated to the expected return.

2.2.3 Carhart (1997)

In his article *On Persistence in Mutual Fund Performance*, Carhart (1997) finds the returns of mutual funds may to some extent be explained by a *momentum* factor.

The author uses monthly data on diversified equity funds ranging from January 1962 to December 1993, making a dataset consisting of 1,892 funds in total. According to the author, it was the largest and most complete mutual fund database available at the time.

He uses two models to measure the performance of the mutual funds: the Capital Asset Pricing Model, and his own model, consisting of four factors (commonly known as the Carhart 4-factor model). Carhart constructs his model using Fama and French's 3-factor model, and an additional factor following Jegadeesh & Titman (1993), which published an article examining one-year momentum anomalies. In his paper, the Carhart 4-factor model is presented as follows:

$$r_{it} = \alpha_{iT} + b_{iT}RMRF + s_{iT}SMB + h_{iT}HML + p_{iT}PR1YR + e_{it}$$

The new momentum factor is here denoted as $p_{iT}PR1YR$, and is constructed as the equal-weight average of firms with the highest 30 percent eleven-month returns lagged one month, minus the equal-weight average of firms with the lowest 30 percent eleven-month returns lagged one month (Carhart, 1997).

The new momentum factor provided a monthly excess return of 0.82, a standard deviation of 3.49, and a t-statistic of 4.46, making it significantly different from 0.

The factors SMB and HML also achieved high returns, and the author stated that they could account for much of the cross-sectional variation in the mean returns of stock portfolios.

Further, he finds that the 4-factor model substantially improves on the average pricing errors of the CAPM and the 3-factor model by Fama & French (1992).

2.2.4 Pástor & Stambaugh (2003)

The final factor I will examine is the *liquidity factor*. Published in the Journal of Political Economy, the paper *Liquidity Risk and Expected Stock Returns* by Pástor & Stambaugh (2003) investigates if marketwide liquidity is a state variable important for asset pricing.

Data for this article was obtained from the Center of Research in Security Prices (CSRP) at the University of Chicago, where they got daily stock data from the New York Stock Exchange (NYSE), and American Stock Exchange (AMEX). The authors did not use data from NASDAQ, as NASDAQ data was only available for a part of their desired study period, which was from 1966 through 1999.

The term liquidity may be defined in several ways, but the authors use the following definition: "the ability to trade large quantities quickly, at low cost, and without moving the price" (Pástor & Stambaugh, 2003, p. 644).

The concept of a liquidity factor is based on the relatively intuitive notion that an investor would demand a premium for holding a stock with low liquidity. The investor would take on additional risk of sustaining larger implicit or explicit transaction costs, should he or she suddenly need to liquidate a position.

Since liquidity itself is not able to be traded, Pástor & Stambaugh propose a long-short portfolio, where you short the high-liquidity stocks and go long on the low-liquidity stocks (Pinto et al., 2015).

The authors present the following equation, building on the Fama-French 3-factor model:

$$r_{i,t} = \beta_i^0 + \beta_i^L L_t + \beta_i^M MKT_t + \beta_i^S SMB_t + \beta_i^H HML_t + \epsilon_{i,t},$$

here, the new liquidity factor is denoted by the letter L (also commonly denoted LIQ), $r_{i,t}$ denotes asset i's excess return, MKT denotes the excess return on a broad market index, and the other two factors, SMB and HML, are payoffs on long-short spreads constructed by sorting stocks according to market capitalization and book-to-market ratio (Pástor & Stambaugh, 2003).

2.3 Using Machine Learning for Capital Market Predictions

As mentioned in the introduction of this thesis, machine learning methods are being put into use as predictors of stock returns in foreign markets with impressive results. A quite recent example of this is Fischer & Krauss, which with their paper published by the European Journal of Operational Research in 2018, utilized techniques like Long Short Term Memory networks (LSTM), random forests, and logistic regression.

2.3.1 Fischer & Krauss (2018)

The paper makes three contributions to the literature, first by applying LSTM networks to a large, liquid, and survivor bias-free stock universe to assess its performance in large-scale financial market prediction tasks. As LSTM networks are state-of-the-art when it comes to machine learning, no such research had been done previously (Fischer & Krauss, 2018).

The second contribution is in the form of shedding additional light on the "black box" of artificial neural networks by examining how they may be a source of profitability, rather than just a purely academic tool.

Finally, the third contribution is using the findings from the paper to construct a simplified, rules-based trading strategy based on the patterns the LSTM uses for selecting stocks.

The data used when constructing the LSTM network was S&P 500 data sourced from Thomson Reuters, from December 1989 to September 2015. In order to remove any survivor bias, they consolidated the lists into a binary matrix which indicated whether a given stock was still in the S&P 500 the subsequent month. Using this method, the authors were able to approximately replicate the index at any given point in time between December 1989 and September 2015. They then acquired daily total return indices for the period January 1990 to October 2015, these returns were with dividends and adjusted for all relevant stock splits and corporate actions. The software used was a combination of Python 3.5, Tensorflow, and R.

Their methodology consisted of five key steps, where in the first step, they split their data into a training set (75%) and a testing set (25%), a procedure that is very common in the field of machine learning. They then constructed the input sequences for the LSTM network.

Thirdly, they give an in-depth discussion of LSTM networks. The fourth step is describing their benchmarking machine learning methods, before they finally in the fifth step developed their trading approach.

After running the LSTM network and benchmark models on the data, the authors found that the LSTM network achieved daily returns of 0.46%, an average predictive accuracy of 54.3%, and an impressive Sharpe ratio of 5.8 prior to transaction costs. They also found that the LSTM network outperformed the different benchmarking models. It also outperformed the general market, especially in the period 1992 to 2009, but this strong performance seems to fade from 2010 and beyond, as the authors suspect the excess returns have been arbitraged away in this period. Post 2010 the profitability seems to fluctuate around 0 after transaction costs have been applied (Fischer & Krauss, 2018). All in all, the results of this paper indicate that there is a strong case for using machine learning techniques to predict stock prices.

2.3.2 Chalvatzis & Hristu-Varsakelis (2020)

Another paper published even more recently than Fischer & Krauss, is a paper named "*High-performance stock index trading via neural networks and trees* (Chalvatzis & Hristu-Varsakelis, 2020)," published in *Applied Soft Computing* in the fall of 2020. In this paper, the authors examine the use of machine learning techniques for predicting future stock prices, over several different American indexes, not just the S&P 500.

The authors describe the contribution of their paper as a "novel, systematic, model-agnostic approach to exploiting the potential profitability of predictions. (Chalvatzis & Hristu-Varsakelis, 2020)." The term "model-agnostic" refers to the fact that the authors use different machine learning models to study the data while making no assumptions that the structure of the underlying data can be described accurately by the model because of its nature. This approach consists of two parts, developing a trading strategy that is effective in generating profits with a reasonable trading risk, and a price prediction model which was tuned for profitability rather than accuracy as typically done in the existing literature.

As for the data used in the paper, the authors used data from the S&P 500, Dow Jones Industrial Average, NASDAQ, and Russell 2000, from 4.1.2010 to 20.12.2019. The software

used for constructing the models was a combination of Python 3 and Tensorflow v1.8.

Their two LSTM networks were trained on a method called back-propagation through time, where they applied exponential decay to their learning rate and trained the model for 1600 iterations. The random forest and gradient boosted trees were modeled using a Python-package called scikit-learn and another one named XGBoost. The use of these packages led to very fast and scalable executions of the models (execution time for a complete back-test averaged at between 5-10 minutes, compared to an average of 14 hours for the LSTM network).

The findings of the paper show that both the LSTM networks were better suited to predict the price movements than the other methods in all four stock indexes at a 5% confidence level. At the same 5% confidence level, there was no significant difference in the prediction accuracy between the two LSTM models, or between the Gradient boosted trees and the random forest model.

As mentioned earlier, the authors were also interested in seeing how profitable their models could be in practice, so when presenting the return and risk statistics for the models, they also include the results of a hypothetical Buy and Hold strategy (BnH). This is then compared to the cumulative returns of the different models, given by: $CR = \Pi_2^N s_i/s_{i-1} - 1$ for a portfolio that has a value of s_i over trading days denoted by $i = 1, 2, ..., N$.

The results show that the model which produces the best CR over the 2010-2019 period is largely one of the two LSTM networks, with the notable exception of the S&P 500, where the Gradient Boosted trees gives a CR of 428%, whereas the highest overall CR was given by the LSTM on the NASDAQ index, with a CR of 497%. The BnH was outperformed by all models on all indexes. The authors also utilized the Information Ratio in a clever way, where they compared all the models to each other, creating a matrix where the active return and tracking error was calculated for the return of one model using another as the benchmark. By doing this they could see which models performed best on the different indexes.

Interestingly, they also compared their findings to other papers within the field of stock prediction (Fischer & Krauss (2018), Krauss et al. (2017), Baek & Kim (2018), Zhou et al. (2019)), to see which models produced the best results. In this comparison, the authors found

that all of their models outperformed the models in the other papers in terms of CR and Sharpe Ratio.

As for future work, the authors suggest further experimentation on even more sophisticated allocation and trading strategies (e.g. short sales), as well as using the trading strategies presented in the paper on other price prediction schemes to examine the possibilities of profitability.

2.3.3 Krauss, Do, and Huck (2017)

When it comes to the topic of using random forests for statistical arbitrage, few papers have been more influential than *Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500* by Krauss, Do, and Huck, published in the European Journal of Operational Research. The paper uses three machine learning methods, namely deep neural networks (DNN), random forests (RAF), and gradient boosted trees for statistical arbitrage in the U.S. market. These methods are also combined into ensembles (ENS) of different configurations to examine their collective effectiveness in market predictions (Krauss et al., 2017).

In the introductory chapter, the authors give an interesting example of how rapidly modern deep learning methods have developed. In 1997, a custom computer named Deep Blue was able to defeat the world champion, Garry Kasparov, in a chess game, which was considered a large feat at the time. But this pales in comparison to what was achieved only nine years later when a computer named Alpha Go (based on deep neural networks and Monte Carlo tree search) defeated a champion Go player. "Go" is a game with origins in Asia, with a much higher level of complexity than chess. In chess, the number of total possible moves are 35^{80}, while in Go, this number is 250^{150}, a more complex computer is needed for defeating a human Go player. The victory of Alpha Go was a showcase of the advantage deep learning methods had over the "shallow" machine learning models, such as support vector machines and older tree-based models (Krauss et al., 2017).

The data used in the paper is from the S&P 500, motivated by the computational feasibility, and the high liquidity and market efficiency the S&P 500 brings, as it accounts for roughly

80% of the US stock market capitalization. The data ranges from 1996 to 2015 and is adjusted for dividends, stock splits, and corporate actions (Krauss et al., 2017). All their preprocessing and handling of the data was done in R, with the help of packages *xts*, *TTR*, and *Performance Analytics*, while the deep neural networks, gradient-boosted trees, and random forests were implemented via *H2O*, a Java-based platform for fast, scalable, open-source machine learning (Krauss et al., 2017). The stock data was split into training and testing sets of 75% for training, and 25% for testing for each study period (one study period was approximately 4 years).

For the output of their models, they construct a binary response variable, which returns 1 if a given stock is predicted to outperform the cross-sectional median of all stocks in the next period, and returns 0 if not. This creates a classification rather than a regression problem, as this is shown to create better predictions on financial data by the existing literature (Krauss et al., 2017). As mentioned above, each study period initially consists of 1000 days, but 240 of these days are lost due to the feature calculation of the models, which calculate the return of the first 240 days.

When it came to forecasting the most profitable stocks for each period, the authors applied a method where they predict the probability of each stock to outperform the cross-sectional median one period ahead (t+1). The stocks were then ranked in descending order according to this probability, making the stocks highest on the list the most undervalued (according to the models).

Based on this ranking, they go long on the stocks with the highest probability of outperforming the cross-sectional median and go short on the stocks with the lowest probability. The amount of stocks in the short and long portfolio is dependent on the desired size of the total portfolio, and by not selecting the middle stock for either portfolio, the authors avoid the stocks with the highest directional uncertainty (Krauss et al., 2017).

In the results chapter, they analyze the performance of different sized portfolios, where the number of stocks in the long and short portfolios is denoted by k. The portfolio sizes they analyze are given by $k \in (10, 50, 100, 150, 200)$, which are compared with respect to returns per day prior to transaction costs, standard deviation, and daily directional accuracy.

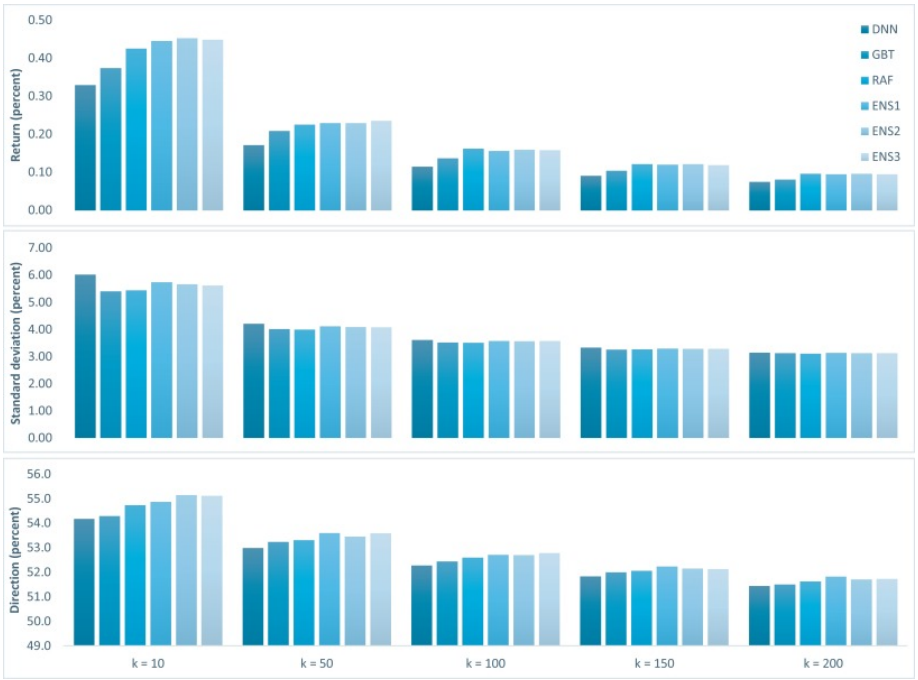


Figure 2.1: Differences between portfolio sizes, (Krauss et al., 2017)

The *k=10* portfolio outperforms the other, larger portfolios, and the authors point out that increasing k leads to a decrease in returns and directional accuracy, but also a decrease in standard deviations.

The authors choose to focus on the *k=10* portfolio for the remainder of the results chapter, where the different methods are compared to each other on the basis of different daily return characteristics, such as mean returns for the long and short portfolio, standard error, standard deviation, skewness, and kurtosis to mention a few. They find that all their strategy variants possess positive skewness, a desirable feature for investors. They also find that the returns are leptokurtic - driven by large outliers, which is in accordance with the theory. Further, following the Risk Metrics approach of Mina & Xiao (2001), the one percent value at risk is roughly twice the level of the general market, with the DNNs showing the highest risk, while RAFs show the lowest.

In conclusion, they find that returns, regardless of method, appear to be deteriorating over time, starting as early as 2001. They believe this is caused by the increase in popularity of

machine learning techniques, as well as a rise in computer power, which causes the excess returns to be arbitraged away. Despite this, the authors note that in times of significant market turmoil, such as the financial crisis or the subsequent European debt crisis, there still may be potential for successful relative-value arbitrage (Krauss et al., 2017).

2.4 Machine Learning Methods

This section aims to give the reader a general introduction to the theory behind the two main methods applied in this thesis: Long Short Term Memory networks and random forests. This introduction includes the methodology which they are based on (neural networks and decision trees) to give the reader more context.

2.4.1 Neural Networks

One of the most popular algorithms in machine learning are called *neural networks*, which cover a wide range of concepts and techniques. A neural network consists of several neurons and connections between those neurons. Neural networks start with unknown parameter values, which are estimated when we fit our neural network to a data set using what is known as *backpropagation*. The function of backpropagation is to optimize the weights and biases in the neural networks.

Neural networks consist of an *input layer* with at least one input neuron, one or more *hidden layers* with several neurons, and finally, an *output layer* with at least one output neuron, as illustrated in figure 2.2.

The hidden layer receives input from the input layer (or another hidden layer, depending on the network) and provides output to another layer, either an output or hidden layer.

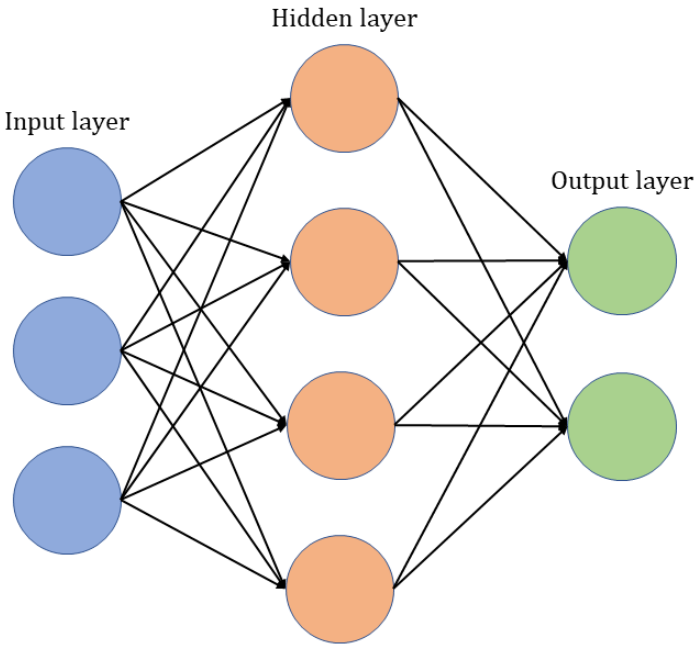


Figure 2.2: A simple neural network

All hidden layers must have an activation function in order to define how the weighted sum of the inputs are transformed into an output from a neuron (or multiple neurons) (Brownlee, 2021). Three common activation functions are the *ReLU*, *sigmoid*, and the *Tanh*, neither of which are linear. The reason for preferring to use non-linear activation functions is that it allows the network to learn more complex tasks. As an example, take the sigmoid activation function,

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

Here, x denotes the sum of weighted activations from the previous layer (input or hidden), and could in theory be any number. x is then transformed into a value between 0 and 1 by the sigmoid function, making the sum of weighted activations much simpler to interpret. In addition to this, the sum of weighted activations has a *bias term*, which tells you how high the weighted sum needs to be before a given neuron starts to get significantly active.

Finding the appropriate weights and biases for the network is an enormously important task, and is essentially how the network "learns." In that task, *gradient decent* plays a large role.

Gradient descent is the process of nudging an input of a function by some multiple of the negative gradient. It's a way to converge towards a local minimum of a *cost function* (a valley in a 3d graph).

This cost function takes the squared difference between the estimated value and the observed value (sum of squared residuals) and returns the quality of the model. In other words, we want the output of the cost function to be as small as possible.

When it comes to adjusting the actual behavior of the network, this can be done through three parameters:

- Changing the bias
- Changing the weights
- Changing the activations from the former layer

The fine-tuning of these three parameters is done through *backpropagation*, which changes the bias, weights, and activations based on the performance of the network in the previous iteration. Using backpropagation allows you to reduce the sum of squared residuals in the network by optimizing parameters (Starmer, 2020).

When a neural network has more than one output, it is common to use a *SoftMax* function. Regardless of how many raw output values there are, SoftMax output values will always be between 0 and 1.

Finally, epochs, batches, and iterations are central elements in neural networks. An epoch is one full forward and backward pass of the entire data set through the neural network. An epoch is further broken down into smaller batches, where the batch size is given as the total number of training samples in a given batch. An iteration is simply the number of batches you need to make up one epoch (Sharma, 2017).

2.4.2 LSTM Networks

Long Short Term Memory networks are a relatively new method in machine learning. Introduced by Hochreiter & Schmidhuber (1997), which found that LSTM networks achieved a much faster learning rate, and led to many more successful runs than other methods, such as

real-time recurrent learning, back propagation through time, recurrent cascade correlation, Elman nets, and neural sequence chunking. They also found that LSTM networks were able to solve complex, artificial long-time-lag tasks that had never been solved by any other recurrent network algorithm.

LSTM networks are special in that they are specifically designed to overcome the limitations of Recurrent Neural Networks (RNNs). Limitations like vanishing and exploding gradients, as proven by Sak et al. (2014), who compared LSTM networks to RNNs and DNNs by building speech recognition models. They also found that the LSTM models converged quickly, and gave state-of-the-art performance for relatively small-sized models.

Like other neural networks, the LSTM networks consist of different layers. First, we have the input layer. The number of neurons in this input layer depends on the feature space of the model, which is how many explanatory variables one wishes to have in said model.

Next is one or more hidden layers, which is where we find the feature that makes LSTM networks stand out from other RNNs; the *Memory Cells*. Every memory cell has three gates that maintain and adjusts its respective cell state s_t : a forget gate (f_t), an input gate (i_t), and finally, an output gate (Fischer & Krauss, 2018). I have illustrated an example of a memory cell in figure 2.3.

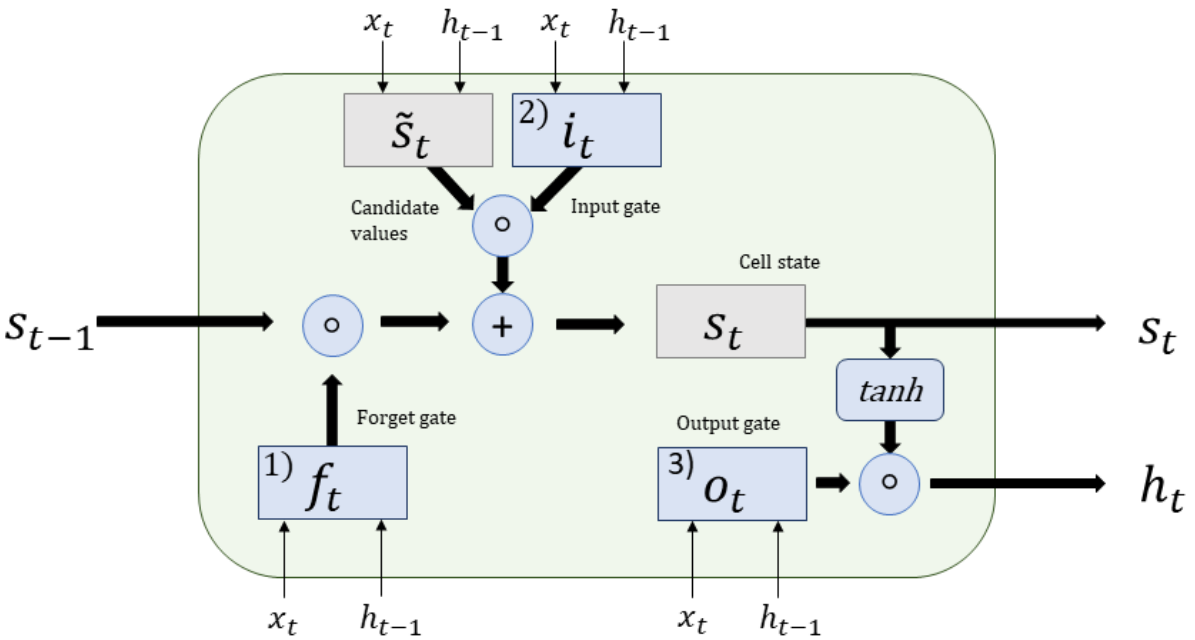


Figure 2.3: A LSTM memory cell following Fischer & Krauss (2018), Graves (2013), and Olah (2015)

Each of these gates functions as a filter, where at every timestep (t), all three gates are presented with input, denoted x_t, and the output of the cell at the previous timestep, denoted h_{t-1}. The forget gate defines what information to remove from the cell state, the input gate defines what information to add to the cell state, while the output gate defines what information from the cell state to use as output.

For the next equations, the following notation is used:

- x_t denotes the input vector at time t
- $W_{f,x}, W_{f,h}, W_{\tilde{s},x}, W_{\tilde{s},h}, W_{i,x}, W_{i,h}, W_{o,x},$ and $W_{o,h}$ are weight matrices.
- $b_f, b_{\tilde{s}}, b_i,$ and b_o denote bias vectors.

- $f_t, i_t,$ and o_t are vectors of the activation values for each gate.
- s_t and $\tilde{s}$ represent the vectors for cell states and the candidate values.
- Finally, h_t denotes a vector for the LSTM layer's output.

When the network performs a forward pass, the cell states s_t and outputs h_t for the LSTM layer are calculated as follows:

The current input x, the outputs of the memory cells h_{t-1} in the previous timestep, and the bias terms b_f of the forget gates are used to calculate the activation values for the forget gate f_t at the current timestep t. By doing this, the LSTM layer selects which data to remove from the previous timestep. A sigmoid function, ranging from 0 to 1, then scales the activation values to a range of 0, which means that value will be completely forgotten, to 1, meaning it will be remembered in its entirety (Fischer & Krauss, 2018):

$$f_t = sigmoid(W_{f,x}x_t + W_{f,h}h_{t-1} + b_f).$$

In the second step of the process, candidate values $\tilde{s}$ that could be added to the cell states are computed, and the activation values for i_t of the input gates are calculated. These two operations decide what data will be added to the networks cell state s_t. The $\tilde{s}$ is calculated using a *tanh* function ranging from -1 to 1, while the i_t function is calculated with a sigmoid function, similarly to f_t in the previous step:

$$\tilde{s} = tanh(W_{\tilde{s},x}x_t, +W_{\tilde{s},h}h_{t-1} + b_{\tilde{s}}),$$

$$i_t = sigmoid(W_{i,x}x_t + W_{i,h}h_{t-1} + b_i).$$

Next, the new cell states s_t are calculated using the previous equations. $\circ$ denotes the Hadamard (elementwise) product:

$$s_t = f_t \circ s_{t-1} + i_t \circ \tilde{s}.$$

Finally, in the last step, we derive the output h_t of the memory cell with the following

equation:

$$o_t = sigmoid(W_{o,x}x_t + W_{o,h}h_{t-1} + b_0),$$

$$h_t = o_t \circ tanh(s_t).$$

When the LSTM network is processing a given input sequence, the features of that input sequence is presented timestep by timestep. When the final element of the input sequence is processed, the final output of that sequence is returned. Throughout the training of the network, its bias and weight terms are adjusted according to a minimization function, which aims to minimize the values of a given metric. The metric used in this minimization function is selected by the creator of the network and depends on its desired use.

The amount of these bias and weight terms are calculated through the following equation:

$$4hi + 4h + 4h^2 = 4(hi + h + h^2) = 4(h(i + 1) + h^2),$$

where h denotes the number of hidden units of a given LSTM layer, and i denotes the number of input features (Fischer & Krauss, 2018). Thus, the dimensions of the four weight matrices which are applied to the inputs at each gate ($W_{f,x}, W_{\tilde{s},x}, W_{i,x},$ and $W_{o,x}$) are denoted as *4hi*, the dimensions of the bias vectors ($b_f, b_{\tilde{s}}, b_i,$ and b_o) are denoted as *4h*, and the dimensions of the weight matrices applied to the outputs of *t-1* ($W_{f,h}, W_{\tilde{s},h}, W_{i,h},$ and $W_{o,h}$) as $4h^2$.

2.4.3 Decision Trees

In this thesis, random forest models play a significant role in addition to LSTM networks. In this subsection, I will give an introduction to the theory behind decision trees, which are what random forests are built upon.

Decision trees consist of *internal nodes* and *terminal nodes* (also referred to as *leaves*). Internal nodes are where the predictor space is split, the *branches* are the segments of the tree that connect the nodes, and the terminal nodes are where a branch ends. Decision trees are typically split into regression trees and classification trees.

Regression trees are constructed through two steps. In the first step, all possible values for

$X_1, X_2, ..., X_p$ (the predictor space) are divided into J distinct and non-overlapping regions, $R_1, R_2...., R_J$, by using an approach known as *recursive binary splitting*. In the second step, we make the same prediction for every observation which falls into the region R_J. Thus, that prediction is the mean of the response values for the training observations in R_J. For more details on how the regions R_J are constructed, I refer to chapter 8 in James et al. (2013).

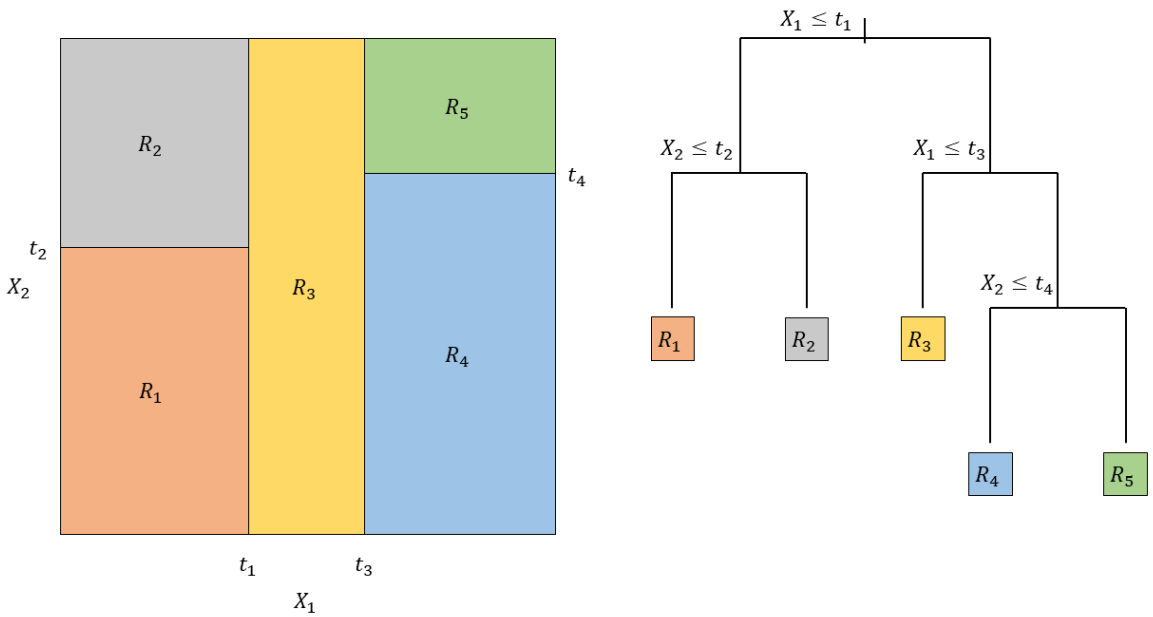


Figure 2.4: Left: The result of recursive binary splitting. Right: Illustration of a decision tree.

A concern with decision trees is the possibility of overfitting, where the tree becomes too large and complex, with too many regions $R_1, ..., R_J$ (too many splits). A solution to this is to only expand the tree as long as the decrease in the Sum of Squared Residuals (SSR) is higher than a relatively high threshold. This approach leads to smaller trees but may be too short-sighted. A poor split high in the tree which only makes a small reduction in SSR might be followed by an exceedingly good split further down. Should we then simply construct a much larger tree (T_0), and prune it down to obtain a *subtree*? The problem with this is that estimating cross-validation error for every possible subtree would be very computationally

intensive, as the number of possible subtrees is extremely high.

This is solved by what is know as *cost-complexity pruning* or *weakest link pruning*, where we consider a sequence of trees indexed by a tuning parameter (α) which is non-negative. This is much more computationally efficient than considering every possible subtree.

Thus, a regression tree may be constructed using the following algorithm:

1. First, use recursive binary splitting to make a large tree based on the training data, only stopping when each terminal node has fewer than a given number of observations.
2. Then, apply the cost complexity pruning to the large tree made in step 1 to obtain a sequence of best subtrees as a function of α.
3. Use K-fold cross-validation to determine α, then for each $k = 1, ..., K$:
    - Simply repeat steps 1 and 2 on all but the kth fold of the training data.
    - Evaluate the *mean squared prediction error* on the data in the left-out kth fold as a function of α.
4. Finally, return the subtree from step 2 which corresponds to the given value of α (James et al., 2013).

The second and final type of decision tree is a classification tree, which has many similarities with the regression trees I just discussed above. In the regression tree, we make predictions based on the mean response of the training observations which belong to the same terminal node, while in the classification tree, we base our predictions on the most occurring *class* of training observations.

There are also similarities when growing the classification tree, but instead of looking to the SSR, we use the *classification error rate*. We want to assign an observation in a given region to the most commonly occurring class of training observations in that region. The classification error rate is given by the fraction of training observations in that region that do not belong to the most common class (James et al., 2013).

In many cases though, the classification error rate is simply not sensitive enough for growing decision trees. An alternative is the *Gini index*, which measures the total variance across

the K classes. It takes on small values when all the proportion of training observations in the mth region from the kth class are close to zero or one. Thus, the Gini index is regularly referred to as a measure of node purity. When it returns a small value, it indicates that a node contains predominantly observations from a single class.

The final alternative to the classification error rate is *entropy*. Entropy takes on a value near zero if the proportion of training observations in the mth region from the kth class are close to zero or one- quite similar to the Gini index. This means that entropy is also a good method for measuring node purity, as it takes on small values when the mth node is pure. In practice, we typically use either the Gini index or entropy when constructing classification trees and evaluating the quality of a given split, as they are more sensitive to node purity than the classification error rate.

Clearly, there are several advantages to decision trees, advantages such as them being quite intuitive by nature and thus easy to explain, which is supported by the fact that they can be illustrated graphically. Some also believe that decision trees are quite similar to our own decision-making process, and can easily handle qualitative predictors.

But like most machine learning methods, decision trees have their weaknesses. The biggest being that they lack the predictive accuracy of other methods (e.g. multiple regression or logistic regression). Also, decision trees are known for being highly sensitive to small changes in the input data (non-robustness) (James et al., 2013). Fortunately, there are solutions to these issues, which will be examined in the next section.

2.4.4 Random Forests

Before examining random forests, we must first understand the basic components which make them a better alternative than simple decision trees. A large factor in this is what is known as *bagging*. Bagging is short for *bootstrap aggregation* and is a method for reducing the variance of a machine learning method. An issue with simple decision trees is that they suffer from high variance, meaning that the results of a decision tree could vary greatly depending on which parts of a dataset it is presented with.

Bagging is therefore a useful tool when working with decision trees. In statistics, averaging a set of observations reduces variance, when given a set of n independent observations $Z_1, ..., Z_n$ with variance denoted σ^2, the variance of the mean $\bar{Z}$ of the observations is given by $\frac{\sigma^2}{n}$. This is where bagging comes into the picture, as we can use *bootstrapping* by taking repeated samples from a single training data set. In doing this, we get B different bootstrapped training sets, which can be used to train a decision tree on the bth bootstrapped training set. This will give us $\hat{f}^{*b}(x)$, and when we average all the predictions we obtain

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x),$$

which is how bagging reduces variance in decision trees (James et al., 2013).

A random forest seeks to further improve upon decision trees and bagged trees by decorrelating them. Starting similarly to bagging, we use bootstrapped training samples to construct a number of decision trees. The difference is that each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors (James et al., 2013). A fresh sample of m predictors is taken at each of the splits, and the split is only allowed to use one of the m predictors. Usually, the value for m is chosen as the square root of the total number of predictors ($m \approx \sqrt{p}$). This means that for every split of the decision tree, the random forest algorithm is only allowed to consider a fraction of all the available predictors. The reason for this feature is that without it, many of the trees may look quite similar. If you have one exceedingly strong predictor alongside many moderately strong predictors in the data set, the vast majority of bagged trees will use this same strong predictor at the top split of the tree. This similarity between the trees leads to a high correlation, which is solved by the random forest algorithm examining only a subset of the predictors.

As we can see, the main difference between bagging and random forests is the size of the predictor subset, denoted by m. If we were to set the number of predictors equal to m, rather than $m \approx \sqrt{p}$, we would simply be left with a standard bagging model.

3 Data, Software, and Hardware

3.1 Data

I use data from the Norwegian stock exchange, Oslo Børs, more specifically, from the Oslo Børs Benchmark Index (OSEBX). The OSEBX consists of the largest and most traded stocks available on the Norwegian stock exchange, weighed by free-float market capitalization. At the time of writing, the index has 69 constituents, which are reviewed semiannually (Oslo Børs, 2021).

Data for the OSEBX constituents are available through the TITLON database, which is a collaborative effort by several Norwegian universities and academic institutions. It contains fully adjusted stock data from 1980 and onward, as well as bond data, accounting data, and calculated Fama-French factors which are useful for financial research (TITLON Team, 2020). In this paper, I have used daily adjusted closing prices of OSEBX constituents from January 2006 to November 2020, as well as Fama-French factors from January 2009, to November 2020.

The data is split into 4 year study periods, with 3 years (750 trading days) devoted to training, and 1 year (250 trading days) devoted to testing. This gives me a total of 12 non-overlapping trading periods, starting in 2009. The predictions for 2009 are trained on the data from the previous three years, which are then used to make out-of-sample predictions for the next day throughout 2009. Once the first study period is completed, the block moves on, making predictions for 2010 based on data from 2007 to 2009. This process continues until all 12 non-overlapping trading periods are completed. Please note that the reason for not acquiring data from after November 2020, is that TITLON's database was not updated beyond this point, due to Oslo børs's transition to Euronext.

3.2 Software and Hardware

All data collection is done via R version 4 (R Core Team, 2017), through TITLON's SQL-based portal, while the LSTM network and the random forest model are run through Python 3.8 (Van Rossum & Drake, 2009) with the help of several packages. These packages are Keras

(Chollet & others, 2015) on top of TensorFlow (Abadi et al., 2015), pandas (McKinney, 2010), and NumPy (Harris et al., 2020).

The LSTM network is trained on an NVIDIA GTX 960m GPU, while the random forest model is trained on an Intel i5 CPU. Training the LSTM network on a GPU rather than a CPU has a major advantage in that GPUs are much better suited for running several calculations at once, significantly reducing the time required for training (Dsouza, 2020).

3.3 Handling Missing Values

In data science, there are several ways to handle missing values in a data set, and selecting which method to use is often dependent on the usage and type of said data.

Common missing data mechanics are *Missing Completely at Random (MCAR)*, *Missing at Random (MAR)*, and *Missing Not at Random (MNAR)*.

The most common methods for handling missing values are *Row deletion*, *mean/median imputation*, *hot deck methods*, *Multiple Imputation*, and *Last Observation Carried Forward*, each method is briefly described below.

Row deletion is the simplest of the solutions, as you simply delete the entire row that contains a missing value. A problem with this approach is that it can cause bias in the data (if applied, data should be missing completely at random (MCAR)).

Mean/median imputation takes the mean or median of the whole data set and uses that as an estimate for the missing value. This method is also quite simple but may lower the variability of the data set depending on how many NA-values you need to replace with this exact same mean/median value.

Hot deck methods compute a missing value from similar values. It thus leads to more educated guesses but is more computationally intensive.

Multiple imputation draws a random sample from the data and uses this to make a regression. Using this regression, we can estimate the value of the missing data point. This process is then repeated a number of times with many different samples, in order to get the best

possible estimate. Again, this method allows us to make even better unbiased guesses but is quite computationally expensive.

Finally, we can carry forward the last non-missing value for a given observation. This is a sensible solution when dealing with missing values in daily financial time series data, which is what I am using in this thesis. The reason for this is that many of the values that were missing in my data set, were due to low liquidity—the missing value stock was simply not traded on that day, leading to a missing value. When working with financial data it is reasonable to assume that if a stock is not traded on a given day, the market value is unchanged from the day before. Thus, carrying forward the previous value represents the actual market value on a given day better than the other imputation methods described above. Therefore, this is the imputation method I implemented in my data set, using the *na.lockf()* function in the zoo package (Zeileis et al., 2014).

4 Methods

In this chapter I will describe the specifics of the LSTM network and random forest used in this thesis, and elaborate on how I construct the portfolios. I will also provide details of the metrics used to measure the performance of the portfolios.

4.1 The LSTM Network

My LSTM network is constructed with a maximum number of epochs of 1000, with an *early stopping* function built-in. This function stops the training of the model when the model has stopped improving, with a patience value of 10, meaning that when the model has not improved for 10 epochs, the training will be terminated. When the training is stopped, the model from the best epoch is restored. These features help in preventing overfitting and reduce the computational intensity of the model.

As for feature generation, following Fischer & Krauss (2018), I first generate simple returns for a stock s over a number of periods m given by the following equation:

$$R_t^{m,s} = \frac{P_t^s}{P_{t-m}^s} - 1,$$

where P_t^s denotes the price of stock s at time t. The one-day return for any given stock at any given time will thus be denoted as $R_t^{1,s}$.

I then take these one-day returns and construct a feature vector V with the dimension $n_i \times T_{study}$. Here, n_i denotes the number of stocks that are available to the model on the last day of a given training period, and T_{study} denotes the total number of days in the study period. All these returns are then standardized by subtracting the mean, denoted by μ_{train}^m and dividing by the standard deviation, denoted by σ_{train}^m shown in the equation below:

$$\tilde{R}_t^{m,s} = \frac{R_t^{m,s} - \mu_{train}^m}{\sigma_{train}^m}$$

The sequence length is selected to be 240, and the single feature used in the LSTM network will be the standardized one-day return ($\tilde{R}_t^{1,s}$). I then construct overlapping sequences of

length 240, consisting of these standardized one-day returns by first sorting the feature vector (V) by stocks (s) and date (t) from lowest to highest. Then I generate sequences as $\left(\tilde{R}^{1,s}_{t-239}, \tilde{R}^{1,s}_{t-238}, ..., \tilde{R}^{1,s}_{t}\right)$ for each day larger or equal to 240 ($t \geq 240$) and each stock of the study period. Using the first stock (s_1) as an example, its first sequence will look like: $\left(\tilde{R}^{1,s_1}_{1}, \tilde{R}^{1,s_1}_{2}, ..., \tilde{R}^{1,s_1}_{240}\right)$, while its second sequence will be $\left(\tilde{R}^{1,s_1}_{2}, \tilde{R}^{1,s_1}_{3}, ..., \tilde{R}^{1,s_1}_{241}\right)$, and the third $\left(\tilde{R}^{1,s_1}_{3}, \tilde{R}^{1,s_1}_{4}, ..., \tilde{R}^{1,s_1}_{242}\right)$ and so forth. These sequences are run for all stocks and all days in the study period.

On the topic of target selection, I follow Takeuchi & Lee (2013) and Fischer & Krauss (2018) to define a binary classification problem where the response variable (Y^s_{t+1}) for each date t and stock s can be divided into two classes, namely *class 0* and *class 1*. When defining the classes, I place all one-day returns of all available stocks in the next period in order from lowest to highest. A stock is placed in class 1 if its one-day return is higher than the cross-sectional median return of all stocks in the next period, while it is placed in class 0 if it is below the median.

The different layers of my LSTM network are configured as follows:

- The input layer consists of one feature, namely the standardized one-day return ($\tilde{R}^{1,s}_t$) as discussed above.
- The LSTM layers consist of 25 hidden neurons created through CuDNNLSTM with a dropout value of 0.1.
- The output layer consists of two neurons and a Softmax activation function.

Further, the LSTM network has a categorical cross-entropy loss function and an RMSprop optimizer. The RMSprop helps in that it speeds up the gradient descent process by dampening the large oscillations that may occur.

4.2 The Random Forest

Concerning feature generation for the random forest, I use the simple returns ($R^{m,s}_t = \frac{P^s_t}{P^s_{t-m}} - 1$) of each stock s over m periods. Following Takeuchi & Lee (2013) and Krauss et al. (2017), I consider these m periods: $m \in \{\{1, 2, 3, ..., 20\} \cup \{40, 60, 80, ..., 240\}\}$. Here, I first use the

returns of the initial 20 periods, before changing to a "lower resolution," and use returns from longer and longer periods with 20-period jumps. This gives me 31 features in total, which corresponds to approximately one trading year of 240 days.

The output of the random forest is exactly the same as for the LSTM network described above. The binary response variable Y_{t+1}^s assigns a stock s to class 0 if the one-period return is below the cross-sectional median, and class 1 if it is above.

The random forest is constructed with the following specifications:

- Total number of decision trees in the forest: 1000.
- Max depth of each tree: 20.
- Uses simple returns ($R_t^{m,s}$) as features.
- Random feature selection given by $m = \sqrt{p}$, as described in chapter 2 of this thesis.

4.3 Portfolio Construction and Transaction Costs

For all stocks in both models, I forecast the probability ($\hat{p}_{t+1|t}^s$) that a given stock will outperform the cross-sectional median of all stocks in the next period (*t+1*), only using information available up to time t in order to avoid look-ahead bias.

All stocks are then ranked according to this probability from highest to lowest for each period *t+1*. I then select the top k stocks for a *long portfolio*, and the lowest k stocks for a *short portfolio*, similarly to Krauss et al. (2017). Thus, the total size of the combined long-short portfolio will be $2 \times k$.

On the topic of transaction costs, I follow Avellaneda & Lee (2010), Krauss et al. (2017), Fischer & Krauss (2018), and Clegg & Krauss (2018), and apply a cost of 5 basis points per trade, which is common in the literature. Thus, this level of transaction cost is appropriate for comparing profitability between different studies.

4.4 Portfolio Performance Metrics

To compare the different models and portfolio sizes, I use the following metrics:

The first metrics are relatively self-explanatory, they are the mean daily returns of the long and short portfolios, which combined gives us the total mean daily return of the models.

The standard error is given by the standard deviation of the daily returns, divided by the square root of the number of observations,

$$SE = \frac{\sigma}{\sqrt{n}}.$$

The *t-statistic* tells us whether the mean daily returns are significantly different from zero by subtracting the hypothesized population value (in my case zero) from the sample mean, and dividing by the standard error,

$$t - statistic = \frac{\bar{x} - \mu_0}{SE}.$$

I also use metrics that simply describe the distribution of the returns, ranging from the lowest to the highest observed value, increasing by one quartile.

Share>0 shows the share of the daily returns which exceed 0. This metric is useful for telling us how accurate the trading strategy developed by the models is in reality.

The standard deviation (σ) is among the most fundamental measures of volatility of financial models and is given by

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{s=1}^{n}[x(s) - \bar{x}]^2}.$$

Skewness is a measure of the asymmetry of a distribution and is the ratio of average cubed deviations from the sample average (the third moment), to the cubed standard deviation (Bodie et al., 2018), given as

$$Skewness = Average\left[\frac{(x - \bar{x})^3}{\hat{\sigma}^3}\right].$$

Kurtosis is another measure of asymmetry, which examines the effects and likelihood of extreme values in a distribution, also known as *fat tails*. This is measured as deviations

from the average raised to the fourth power, scaled by the standard deviation raised to the fourth power, minus 3. The reason for subtracting 3 is that the expected ratio for a normal distribution equals 3, so by subtracting 3, we use the normal distribution as a benchmark (the kurtosis of a normal distribution is, in effect, defined as 0)(Bodie et al., 2018),

$$Kurtosis = Average\left[\frac{(x-\bar{x})^4}{\hat{\sigma}^4}\right] - 3.$$

Value at Risk (VaR) is commonly used in finance for describing risk associated with a given investment. It is defined as a very low percentile of the whole return distribution.

I use two levels of Value at Risk, 5% and 1%. 1% VaR implies that 99% of the returns of an investment will be in excess of the VaR, while 1% will be below. Similarly, for the 5% VaR, we expect 95% of the returns to exceed the VaR, while 5% will be worse (Bodie et al., 2018).

Assuming for the sake of example that a return distribution is normally distributed, the VaR would be fully dependent on the standard deviation and the mean of the distribution. Knowing that -2.33 is the first percentile of the standard normal distribution, the 1% VaR can be expressed as

$$VaR(1\%, normal) = Mean - 2.33SD.$$

Max Drawdown is another important metric in finance, which describes the difference between the peak value and the lowest low of the portfolio. Mathematically, expressed as

$$\text{Maximum Drawdown } = \frac{\text{peak equity value } - \text{lowest equity value}}{\text{peak equity value}}.$$

Finally, the last two metrics will be quite important to investors, namely the annualized returns, and the annualized standard deviations. For annualized returns, we can relate it to the total return, $x_f(T)$, over a holding period denoted by T, mathematically given as

$$\text{Annualized Returns} = [1 + x_f(T)]^{\frac{1}{T}} - 1.$$

To get the annualized standard deviation, we multiply the standard deviation by the square root of the number of periods per year (Peterson et al., 2014)

$$\sigma \cdot \sqrt{periods}.$$

5 Empirical Results and Discussion

My empirical results and discussion chapter will consist of five primary parts. In the first part, I will examine the overall accuracy of the LSTM network and discuss its development through the study period. I then compare the different portfolio sizes and methods against each other and the OSEBX using the Sharpe ratio and information ratio in the second part. In the third part, I will focus on the portfolio size with the strongest overall performance and make further analysis of it. In the fourth, I will examine the development of profitability through the years of the study period, and discuss any changes that have occurred. Finally, in the fifth part, I will present the results of a multiple regression which measures the model's exposure to systematic risk.

Krauss et al. (2017) found that there was an inverse relationship between the number of stocks k in a portfolio and the corresponding returns and volatility of that portfolio. When k increased, returns and volatility decreased. To see if this relationship holds in the Norwegian market, I have constructed portfolios of three different sizes, namely $k \in \{5, 10, 13\}$. The reason for not constructing even larger portfolios are that the OSEBX is relatively small compared to the S&P 500 used by Krauss et al. (2017), which could lead to the models selecting sub-optimal stocks, simply because there are no other alternatives, as well as the computational intensity which would be required to complete all necessary calculations for larger portfolios.

5.1 Accuracy

Before diving into the results of the models themselves, I must first assure the reader that these results are not random. To prove this, I take the mean accuracy of the LSTM network over the entire study period, 52.7%, and test it on a binomial distribution. When considering a portfolio with *k=10* stocks, this gives us $2978 \times 10 \times 2$ which is the total number of trading days, multiplied by the portfolio size multiplied by the types of portfolio (long and short). This gives us a total of 59560 stocks classified, with 52.7% of those classifications being correct. Using a binomial distribution, I can examine the probability of the model making 52.7% accurate classifications when the true accuracy is 50%. This distribution is given as

$X \sim B(n = 59560, p = 0.5, q = 0.5)$ where X is the number of successes (correctly predicted stocks). With this formula, I calculate the probability of predicting with a 52.7% accuracy, if the true accuracy of the network was actually 50%. Not surprisingly, the probability of this is extremely low, at 1.62266e-40.

When examining the accuracy of the LSTM network over all epochs in chronological order, we can observe that it has been relatively stable over the different periods. There is however a slight negative slope, which I find by running a simple linear regression, illustrated by the golden line in figure 5.1. The result of the linear model is an estimated slope of -5.020e-05, with a standard error of 3.751e-06, t-value of -13.38, and a p-value of <2e-16, making it significant on all levels. This result may be interpreted as the LSTM network losing 0.00502% of its accuracy per epoch, which is concerning in terms of future applications of LSTM networks in financial markets. It should however be noted that there is a noticeable decrease in accuracy for the latest epochs which were in 2020, a year of financial market anomalies and disruptive market moves due to COVID-19 (Banquero et al., 2020).

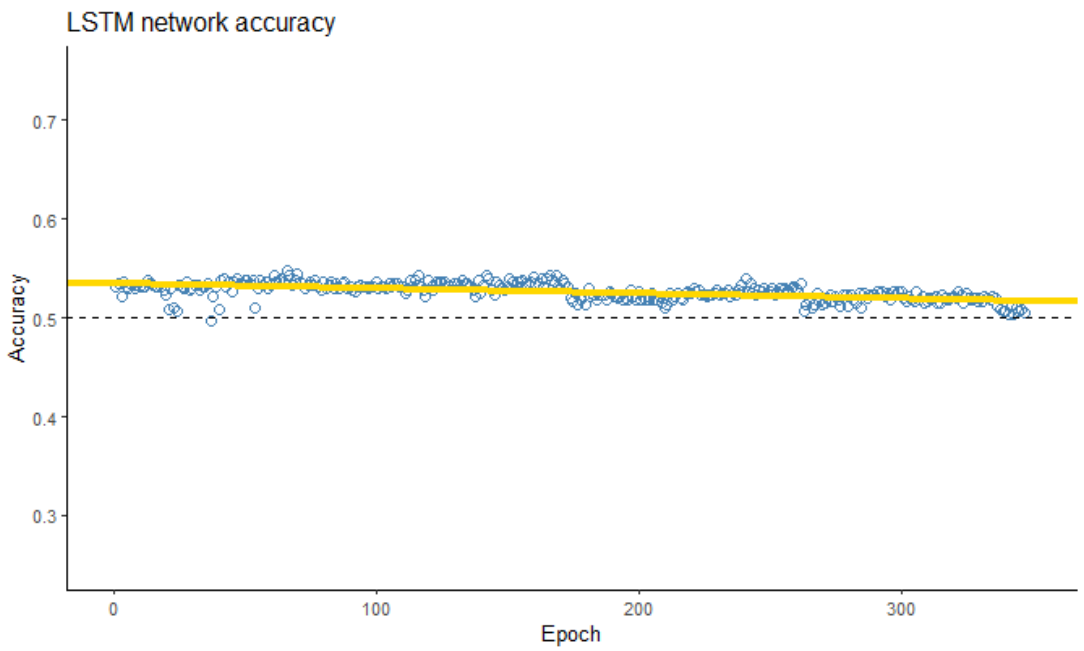


Figure 5.1: Scatter plot of LSTM network accuracy over all epochs

5.2 Analyzing Returns

To get an overview of the risk-return profile of all combinations of portfolio sizes and machine learning methods, I construct a scatter plot with the annualized returns on the Y-axis and annualized standard deviations (σ) on the X-axis, as seen in figure 5.2.

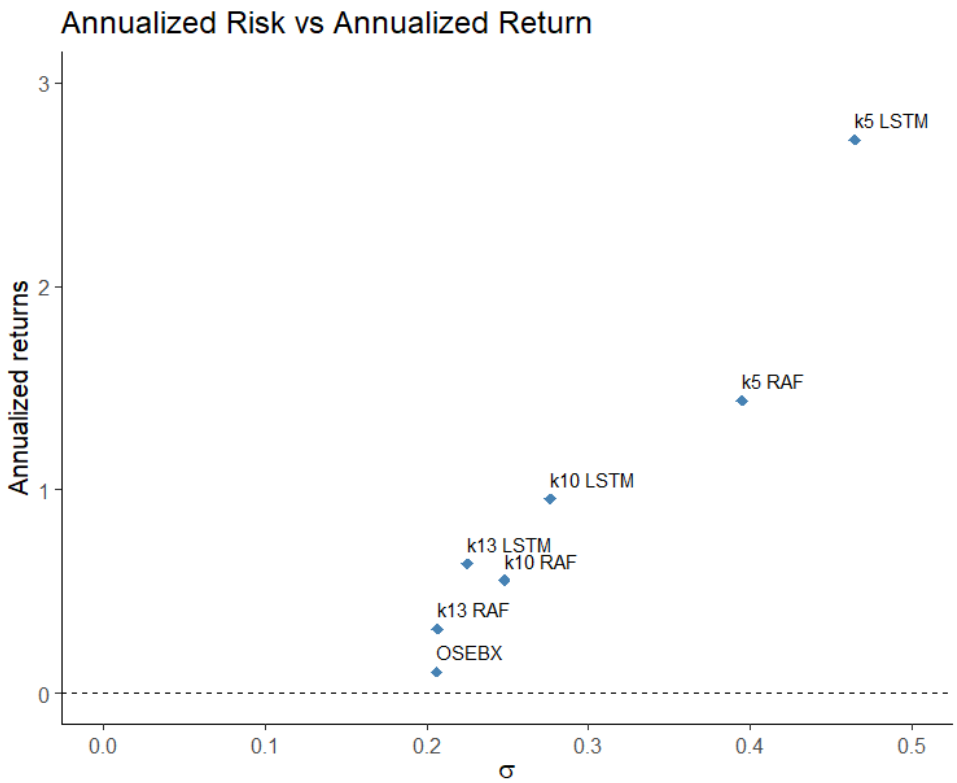


Figure 5.2: Plot of annualized risk vs annualized return after transaction costs

We can immediately see that all models and portfolio sizes are performing well compared to the OSEBX index, with the closest being the random forest model with a portfolio size of 13. It achieves a higher return than the OSEBX while simultaneously having a roughly equivalent risk level. Further, we see that the k 10 to 13 portfolios are relatively close to each other, with the k=5 portfolios being outliers in terms of returns and standard deviations. These types of scatterplots are good for getting a quick and intuitive overview of all portfolio sizes and models, but to get a better impression of which of them a potential investor would prefer, we look to the Sharpe ratio.

Table 5.1: Annualized Sharpe ratios before and after transaction costs

Type	Before transaction costs	After transaction costs
k5 LSTM	10.78966	5.68854
k5 RAF	7.452586	3.512445
k10 LSTM	7.824566	3.30623
k10 RAF	6.109503	2.103419
k13 LSTM	7.320929	2.673188
k13 RAF	5.461627	1.390978
OSEBX	-	0.3968936

Note:
The Norwegian 10-year government bond yield is used when calculating the Sharpe ratio.

In table 5.1, we see the Sharpe ratios for all portfolio sizes and models, as well as the OSEBX for reference. In this table, it is much clearer which combination a potential investor would prefer in terms of risk-adjusted returns, namely the *k=5* portfolio selected by the LSTM network. It dominates the other options both before and after transaction costs have been applied, despite having a relatively high level of risk. Similar to what was observed in the S&P 500 by Krauss et al. (2017), there appears to be a negative relationship between a portfolio's size, and its returns and volatility.

However, several investors deem the Sharpe ratio to be outdated by modern standards, even William Sharpe himself has stated that there are better alternatives out there today. More specifically, Sharpe has recommended the Information ratio (Peterson et al., 2014), which we can see in table 5.2 below.

In table 5.2, we observe the same ranking of portfolios and models which we saw with the older Sharpe ratio, with the *k=5* portfolio size and LSTM network combination reigning supreme before and after transaction costs, with the *k=5* random forest following closely behind.

From these two tables, it appears that the *k=5* portfolio delivers the strongest performance, regardless of method. This increase in performance relative to smaller portfolio sizes is in line with the findings of Krauss et al. (2017), although the smallest portfolio size in their

Table 5.2: Annualized Information ratios before and after transaction costs

Type	Before transaction costs	After transaction costs
k5 LSTM	10.02127	5.204799
k5 RAF	6.674106	3.044502
k10 LSTM	6.159733	2.462599
k10 RAF	4.577624	1.404401
k13 LSTM	5.276082	1.751073
k13 RAF	3.663598	0.7192564
OSEBX	-	-

Note:
As we use the OSEBX index as the benchmark returns, the OSEBX information ratio values return as NaN

paper was a *k=10* portfolio, and the largest being a *k=200* portfolio.

Based on the results from the Sharpe ratio and Information ratio, it is safe to assume that the *k=5* portfolio will yield the strongest risk-adjusted performance, thus, it will be the focus of the remainder of this chapter.

5.3 Examining the Portolio

I begin this section by presenting a comprehensive table of different metrics for the *k=5* portfolio prior to transaction costs, followed by another table with data for after all transaction costs have been applied. Both tables also include the OSEBX for reference. For the particularly curious reader, I have included the same tables for all portfolio sizes and transaction costs in appendix B of this thesis.

5.3.1 Before Transaction Costs

In table 5.3, we can see the mean daily returns for both the long and short portfolios, and it is clear that the long portfolio contributes most to the overall return of both models, but the short portfolio still provides a significant contribution. The sum of the mean daily returns of the long and short portfolio gives us the total mean daily return of the given model, which for the LSTM network is ~0.765%, while the random forest gives ~0.584%. The following standard errors and t-statistics prove that the mean daily returns are significantly different

Table 5.3: Performance Data k5 portfolio before transaction costs

Metric	LSTM (k=5)	RAF (k=5)	OSEBX
Mean return (long)	0.005246502	0.004121689	NA
Mean return (short)	0.002406854	0.00172759	NA
Mean return	0.007653356	0.005849279	0.000479708
Standard error	0.0005363608	0.0004559347	0.000237775
t-statistic	14.269	12.829	2.0175
Lowest	-0.169176700	-0.153261008	-0.0917538117
Quartile 1	-0.007039248	-0.006757942	-0.0055810075
Median	0.006418151	0.003650574	0.0008308918
Quartile 3	0.020686981	0.015533585	0.0070181210
Highest	0.269636856	0.202042632	0.0665158453
Share>0	0.6353257	0.600403	0.5337793
Std.Dev	0.02926978	0.02488084	0.01300175
Skewness	0.2972403	0.7358901	-0.4455084
Kurtosis	6.148643	6.431124	4.215446
5% VaR	-0.03512848	-0.02817399	-0.02029984
1% VaR	-0.06944494	-0.05204822	-0.04026436
Max Drawdown	0.6212994	0.72287	0.3498688
Annualized returns	5.142531	3.028412	0.1046073
Annualized Std.Dev	0.4646433	0.3949711	0.2063964

from 0 for both models.

Next, I examine data from the distributions of the returns, starting with the lowest observed return in a single day. These two values are quite similar for both the LSTM network and random forest, with -16.91% and -15.32% respectively. For reference, the same value for the OSEBX is -9.17%. The first quartile is again quite similar for both, but from the median up to the highest value, the LSTM network seems to pull ahead of the random forest, with the highest return achieved in one day being 26.96% and 20.20% respectively.

The LSTM network also dominates in terms of accuracy, in that it has a higher share of returns larger than 0, but has a higher mean daily standard deviation than the random forest. Both models possess positive skewness, as well as similar kurtosis values. The positive skew will be appreciated by investors, although the random forest is more positively skewed.

For risk measures, the random forest dominates the LSTM network for both 5% and 1%

value at risk, but the random forest has a higher maximum drawdown at 72.2%, compared to the LSTM network which has 62.1%.

Finally, for annualized returns and standard deviations, I observe the same trend we have seen for the other metrics, the LSTM network possesses a higher annualized return of 5.14, but in turn, also has a higher annualized standard deviation.

5.3.2 After Transaction Costs

Now I will make the same analysis for the data after transaction costs have been applied. Examining the effects of transaction costs is important in that it makes the thesis more than strictly theoretical, but lets us evaluate whether these methods of portfolio construction would actually be feasible in a real-world application.

Table 5.4: Performance Data k5 portfolio after transaction costs

Metric	LSTM (k=5)	RAF (k=5)	OSEBX
Mean return (long)	0.004246502	0.003121689	NA
Mean return (short)	0.001406854	0.0007275898	NA
Mean return	0.005653356	0.003849279	0.000479708
Standard error	0.0005363608	0.0004559347	0.000237775
t-statistic	10.54	8.4426	2.0175
Lowest	-0.171176700	-0.155261008	-0.0917538117
Quartile 1	-0.009039248	-0.008757942	-0.0055810075
Median	0.004418151	0.001650574	0.0008308918
Quartile 3	0.018686981	0.013533585	0.0070181210
Highest	0.267636856	0.200042632	0.0665158453
Share>0	0.5950302	0.5443251	0.5337793
Std.Dev	0.02926978	0.02488084	0.01300175
Skewness	0.2972403	0.7358901	-0.4455084
Kurtosis	6.148643	6.431124	4.215446
5% VaR	-0.03712848	-0.03017399	-0.02029984
1% VaR	-0.07144494	-0.05404822	-0.04026436
Max Drawdown	0.738335	0.8463215	0.3498688
Annualized returns	2.721568	1.438787	0.1046073
Annualized Std.Dev	0.4646433	0.3949711	0.2063964

In the post-transaction cost table, we see that the returns of both models have been affected, especially the short portfolio of the random forest, which now only yields 0.07% on average

per day. Despite this, the total mean daily returns are both significantly different from 0 as seen from the relatively high t-statistics. Naturally, the addition of transaction costs will further increase the lowest observed value for both models which now are -17.11% for the LSTM network, and -15.52% for the random forest. These high and consistent returns after transaction costs are contrary to the efficient market hypothesis, as it states that it should not be possible to earn consistent excess returns in a systematic way by e.g. model based trading (Hsu et al., 2016), (Fama, 1970).

Other than that, we see much of the same trends which we observed in the pre-transaction cost table, with the LSTM network delivering high returns but with high levels of volatility in terms of standard deviation, while the random forest provides lower returns but also lower volatility.

Finally, the max drawdowns for the LSTM network and random forest end up at -73.82% and -84.63% respectively, with a net annualized return of 2.722 and 1.439. As we can see, both machine learning methods are still profitable even after transaction costs have been applied, although the LSTM network still has the advantage in terms of returns.

5.4 Profitability in Different Periods

In this subsection, I will break down the results of the $k=5$ portfolio into two main equally long periods, to get a better grasp of how the effectiveness of each method has developed over the study period. I examine the mean daily returns per year in the study period for both the LSTM network and the random forest, as well as the cumulative profits on 1 NOK average investment per day. Finally, I look to the share of stocks with a return greater than 0 per year.

Period 1 goes from 1.1.2009 to 31.12.2014, while period 2 starts 1.1.2015 and ends 26.11.2020, giving me 6 years in each period.

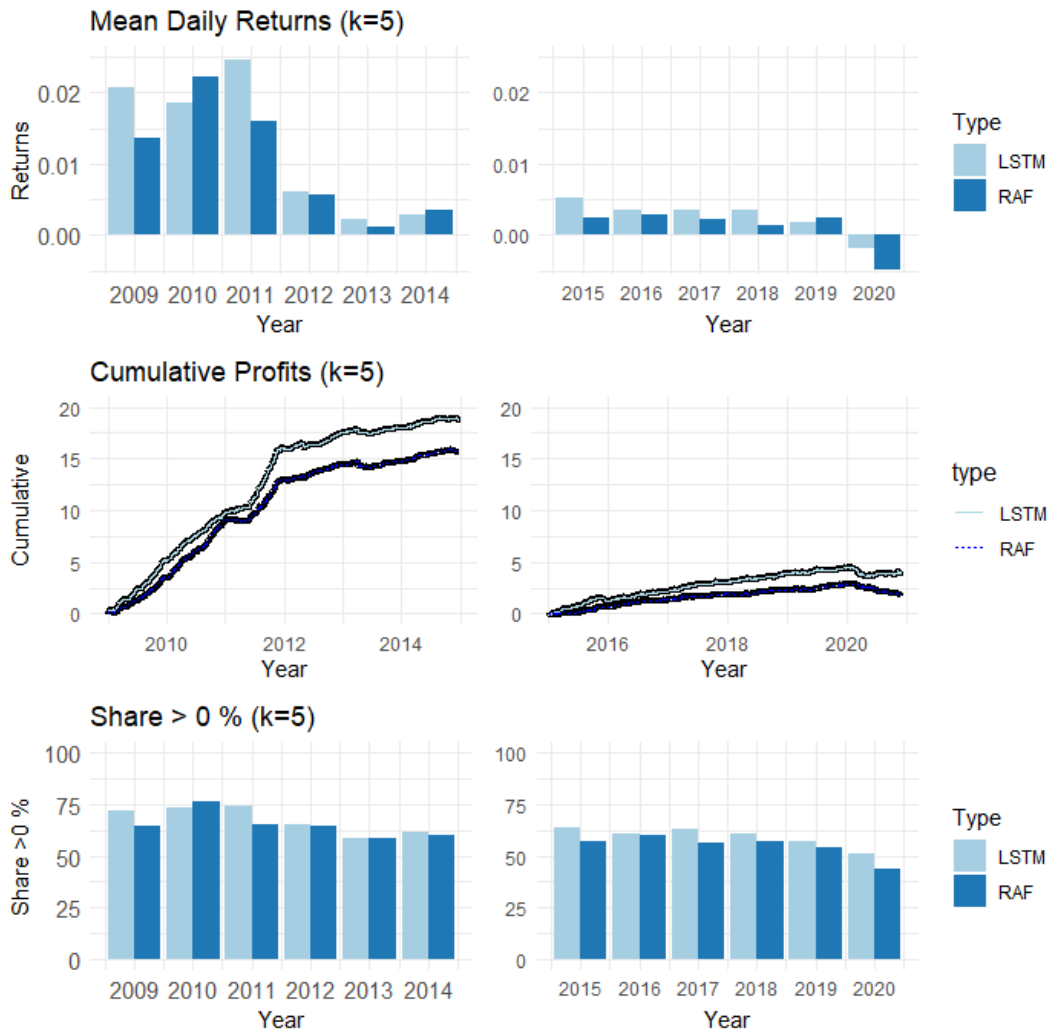


Figure 5.3: Two-period results of the k5 portfolio before transaction costs

Looking at figure 5.3, we see that period 1 displays a very strong performance from both the LSTM network and the random forest, especially in 2009, 2010, and 2011. This is evident by the mean daily return bar chart and the steep slope of the cumulative profits plot, where in the latter, the LSTM network achieves a value of over 19 NOK in late 2014. These returns seem to diminish in 2012, 2013, and 2014, as well as the share of daily returns larger than 0, which also see a decline in this period.

Period 2 sees a continuation of this phenomenon, with a steady waning of daily returns, cumulative profits, and positive returns. Still, the only year which saw negative mean daily

returns was 2020, not entirely surprising considering the large effect the COVID-19 outbreak had on financial markets.

There is a clear negative trend in the daily returns data, as can be seen in figure 5.4.

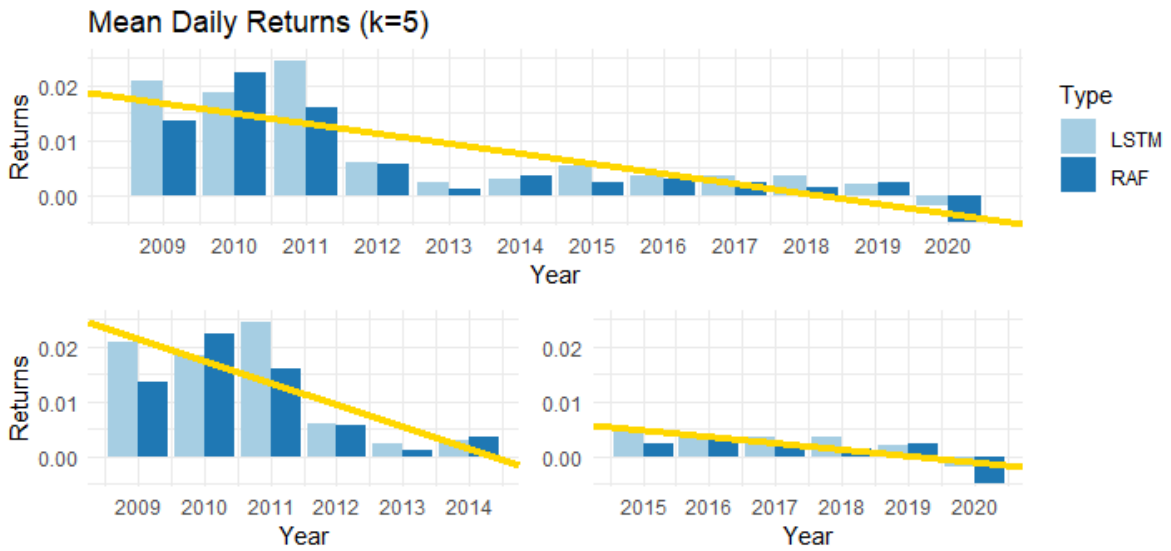


Figure 5.4: Linear regression results of the k5 portfolio before transaction costs

Running a simple linear regression on the mean daily returns data for the different periods gives a more precise impression of how the returns of the two models have developed through the years. The results of this regression analysis can be seen in table 5.5.

Table 5.5: Regression coefficients of daily returns

Period	Slope coefficient	Std. Error	t value	p value
1.1.2009-26.11.2020	-0.0018326***	0.0002842	-6.448	1.74e-06
1.1.2009-31.12.2014	-0.0040161**	0.0008883	-4.521	0.00111
1.1.2015-26.11.2020	-0.0011267**	0.0003182	-3.540	0.00535

Note:
Significance codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Overall, for the whole period, the regression line tells us that the mean daily returns have decreased by a factor of 0.00183 per year since the beginning of the study period. The time

period with the largest decrease is period 1, with a yearly reduction of 0.4% from 1.1.2009 to 31.12.2014. This negative trend has however slowed in recent years, as evident from the 1.1.2015 to 26.11.2020-slope, which is estimated to be -0.00113, or -0.113%. This can also be seen in the plots of the *k=10* and *k=13* portfolios in appendix A.

This decline in profitability is consistent with the existing literature in the field. Fischer & Krauss (2018), Krauss et al. (2017), Bogomolov (2013), Rad et al. (2016), Green et al. (2017), and Clegg & Krauss (2018) all observe a profitability decline in the U.S. market in the period 2009-2011, which continues to the end of their respective study periods.

A notable difference in this profitability decline for the Norwegian market and U.S. market is the time at which it occurs. In this thesis, I observe a decline starting in 2012, up to three years later than in the U.S. market. Bogomolov (2013) also finds that the Australian market does not experience the decline which is observed in the U.S. markets in the period 2009-2011, implying that less mature and less liquid markets experience this decline in a later stage than their U.S counterparts. This is consistent with the findings of Hsu et al. (2016), who discovered a negative relationship between market maturity and returns from machine learning methods.

5.5 Exposure to Systematic Risk

Now, I will present a regression model which examines the *k=5* portfolio's exposure to systematic risk, measured by the factors presented in chapter 2 of this thesis.

The first factors are from the Fama & French (1992) 3-factor model, which consists of a market factor, the SMB-factor which measures the exposure to small minus big stocks, and the HML-factor, which measures exposure to high minus low book-to-market stocks.

Further, I also include the momentum factor developed by Carhart (1997) and the liquidity factor by Pástor & Stambaugh (2003).

Running a Breusch-Pagan test for heteroscedasticity on the fitted model, I get a test value of 1.5127 with 5 degrees of freedom, which equates to a p-value of 0.9116. Clearly, we cannot reject the null hypothesis that the model is homoscedastic with constant variance. This

Table 5.6: Systematic risk regression k=5 portfolios after transaction costs

Factor	LSTM (k=5)	RAF (k=5)
Intercept	0.0056282***	0.0038234***
Market	0.0569468	0.0695965 .
SMB	0.0696615	0.0544602
HML	-0.0090459	-0.0149058
LIQ	-0.0066746	-0.0114442
MOM	-0.0345979	-0.0399754
R-squared	0.00168	0.002641
adj. R-squared	~ 0	0.0009634
Num. Obs	2978	2978

Note:
Significance codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

is positive for the systematic risk model, as the standard errors, confidence intervals, and hypothesis tests associated with the linear model rely upon the assumption of homoscedasticity (James et al., 2013).

Starting with the result of the LSTM network seen in table 5.6, it seems very little of the return of 0.0056 is explained by the five factors applied here, as seen from the R^2 and adjusted R^2 which are both very low.

I observe the same tendencies with the random forest model, although the market factor is proven to be significant on the 10% significance level. It also has a higher R^2 and adjusted R^2 than the LSTM network. Overall, the alphas of both models (measured by the intercept) are both large and highly significant.

What I find in terms of exposure to systematic risk in my model is consistent with the outcomes of similar models in other markets. Fischer & Krauss (2018) found that their LSTM network deployed in the U.S. market had no significant exposure to the market factor or the SMB factor, and exhibited low values for R^2 and adjusted R^2. They also found that the random forest had significant exposure to the market-factor, as well as a higher R^2 and adjusted R^2.

6 Conclusion

In this thesis, I have applied machine learning methods in order to predict stock returns in the Norwegian stock market for the period January 2009 to November 2020. In this task, I have deployed LSTM networks and random forests combined with different portfolio sizes, consisting of long and short portfolios.

The main contribution is a thorough examination of the current state of machine learning methods in finance, in terms of their accuracy, profitability, and exposure to common sources of systematic risk.

The LSTM network exhibits a relatively stable level of accuracy across all epochs, with a mean of 52.7%. When running a linear regression on the accuracy data, I find a slight but significant negative slope of 0.00502%, implying that the overall accuracy of the network is in decline. This could be explained by the inclusion of the year 2020 in the data, which has been the subject of several market anomalies and disruptions due to the COVID-19 pandemic.

I also examine several combinations of portfolio sizes and machine learning methods and explore the profitability of portfolios constructed by these methods versus the OSEBX index. The OSEBX is dominated by all combinations of portfolio size and method, where the *k=5* portfolios stand out in terms of performance relative to the other portfolio sizes. The best overall performance was from the LSTM network when constructing long-short portfolios with 5 stocks each, achieving an annualized Sharpe ratio and Information ratio of 5.68854 and 5.204799 respectively after transaction costs. Second in contention was the random forest *k=5* portfolio with an annualized Sharpe ratio and Information ratio of 3.512445 and 3.044502 respectively.

In further examination of the *k=5* portfolios, I find that the LSTM network outperforms the random forest in terms of returns both before and after transaction costs, but at a higher level of volatility. More specifically, the LSTM network achieved an annualized return and standard deviation of 2.722 and 0.465 respectively after transaction costs, while the random forest achieved an annualized return of 1.438 and a standard deviation of 0.395 after transaction costs.

Although these results are strong and make a good case for the application of LSTM networks and random forests in portfolio construction, it must be noted that the profitability of both models has declined over time. I observe a period of strong returns in the earliest periods, from 2009 to 2012, which subsequently drops off from 2012 to 2014. This decline has continued up to 2020 but at a much lower rate. This is consistent with the literature from other countries and markets, e.g. Fischer & Krauss (2018) and Krauss et al. (2017), who find the profitability of applying machine learning methods in financial markets to be declining, especially in recent years. They ascribe this phenomenon to the fact that machine learning methods are becoming widely available and relatively inexpensive, which leads to potential profits being arbitraged away. This appears to be the case in the Norwegian market as well, although the decline in profits has happened later in Norway, likely due to the fact that the Norwegian capital market is less mature and less liquid than its U.S. counterparts.

I also run a multiple regression to examine the exposure of the $k=5$ portfolio to common sources of systematic risk, namely the Market, SMB, HML, Liquidity, and Momentum-factors. I find that neither of the models has much exposure to these factors, except the random forest, which has some exposure to the Market factor. For further research, I would suggest testing the exposure of LSTM networks and other machine learning methods to other factors, such as a reversal factor, or the VIX index, as seen in Fischer & Krauss (2018), as unfortunately, these factors were not available at the time of writing. Further, I would also suggest examining in even greater detail what factors are causing the international decline in profits from machine learning methods.

In summary, I have found strong evidence that it is feasible to reliably predict returns in the Norwegian stock market using machine learning methods, which was the problem statement of this thesis. Both the LSTM network and random forest are able to construct portfolios of different sizes which are profitable before and after transaction costs. However, the profitability of these methods is in decline in both the Norwegian and U.S. markets, making the future of applying machine learning in financial portfolio construction uncertain, and undoubtedly the subject of many more future studies.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., ... Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems*. https://www.tensorflow.org/

Avellaneda, M., & Lee, J.-H. (2010). Statistical arbitrage in the US equities market. *Quantitative Finance*, *10*(7), 761–782.

Baek, Y., & Kim, H. Y. (2018). ModAugNet: A new forecasting framework for stock market index value with an overfitting prevention LSTM module and a prediction LSTM module. *Expert Systems with Applications*, *113*, 457–480. https://doi.org/10.1016/j.eswa.2018.07.019

Banquero, J. A., Gyramati, A., Laurent, M.-P., Silva, P. J., & Wegner, T. (2020). Applying machine learning in capital markets: Pricing, valuation adjustments, and market risk. In *McKinsey*. https://www.mckinsey.com/business-functions/risk/our-insights/applying-machine-learning-in-capital-markets-pricing-valuation-adjustments-and-market-risk#

Bodie, Z., Kane, A., & Marcus, A. J. (2018). *Investments*. The McGraw-Hill/Irwin.

Bogomolov, T. (2013). Pairs trading based on statistical variability of the spread process. *Quantitative Finance*, *13*(9), 1411–1430.

Brownlee, J. (2021). *How to Choose an Activation Function for Deep Learning*. https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/

Carhart, M. M. (1997). *On Persistence in Mutual Fund Performance: Vol. LII* (No. 1). https://doi.org/10.1111/j.1540-6261.1997.tb03808.x

Chalvatzis, C., & Hristu-Varsakelis, D. (2020). *High-performance stock index trading via neural networks and trees* (Vol. 96, p. 106567). Elsevier Ltd. https://doi.org/10.1016/j.asoc.2020.106567

Chollet, F., & others. (2015). *Keras*. https://keras.io.

Clegg, M., & Krauss, C. (2018). Pairs trading with partial cointegration. *Quantitative Finance*, *18*(1), 121–138.

Dsouza, J. (2020). *What is a GPU and do you need one in Deep Learning?* https://towardsdatascience.com/what-is-a-gpu-and-do-you-need-one-in-deep-learning-718b9597aa0d

Fama, E. F. (1970). Efficient Capital Markets: A Review of Theory and Empirical Work. *The Journal of Finance*, *25*(2), 383. https://doi.org/10.2307/2325486

Fama, E. F., & French, K. R. (1992). The Cross-Section of Expected Stock Returns. *The Journal of Finance*, *47*(2), 427. https://doi.org/10.2307/2329112

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, *270*(2), 654–669. https://doi.org/10.1016/j.ejor.2017.11.054

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv Preprint arXiv:1308.0850*.

Green, J., Hand, J. R., & Zhang, X. F. (2017). The characteristics that provide independent information about average US monthly stock returns. *The Review of Financial Studies*, *30*(12), 4389–4436.

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., R'ıo, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Hsu, M.-W., Lessmann, S., Sung, M.-C., Ma, T., & Johnson, J. E. (2016). Bridging the divide in financial market forecasting: Machine learners vs. Financial economists. *Expert Systems with Applications*, *61*, 215–234.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical*

learning (Vol. 112). Springer.

Jegadeesh, N., & Titman, S. (1993). Returns to Buying Winners and Selling Losers: Implications for Stock Market Efficiency. *The Journal of Finance*, *48*(1), 65. https://doi.org/10.2307/2328882

Krauss, C., Do, X. A., & Huck, N. (2017). Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. *European Journal of Operational Research*, *259*(2), 689–702. https://doi.org/10.1016/j.ejor.2016.10.031

Lintner, J. (1965). Security prices, risk, and maximal gains from diversification. *The Journal of Finance*, *20*(4), 587–615.

McKinney, Wes. (2010). Data Structures for Statistical Computing in Python. In Stéfan van der Walt & Jarrod Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (pp. 56–61). https://doi.org/ 10.25080/Majora-92bf1922-00a

Mina, J., & Xiao, J. Y. (2001). *Return to RiskMetrics: The Evolution of a Standard*. www.riskmetrics.com

Mossin, J. (1966). Equilibrium in a capital asset market. *Econometrica: Journal of the Econometric Society*, 768–783.

Olah, C. (2015). *Understanding LSTM Networks – colah's blog*. http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Oslo Børs. (2021). *OSEBX - characteristics*. https://live.euronext.com/nb/product/indices/NO0007035327-XOSL/market-information

Pástor, L., & Stambaugh, R. F. (2003). Liquidity risk and expected stock returns. *Journal of Political Economy*, *111*(3), 642–685. https://doi.org/10.1086/374184

Peterson, B. G., Carl, P., Boudt, K., Bennett, R., Ulrich, J., & Zivot, E. (2014). PerformanceAnalytics: Econometric tools for performance and risk analysis. *R Package Version*, *1*(3).

Pinto, J. E., Henry, E., Robinson, T. R., & Stowe, J. D. (2015). *Equity asset valuation*. John

Wiley & Sons.

R Core Team. (2017). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. https://www.R-project.org/

Rad, H., Low, R. K. Y., & Faff, R. (2016). The profitability of pairs trading strategies: Distance, cointegration and copula methods. *Quantitative Finance*, *16*(10), 1541–1558.

Sak, H., Senior, A., & Beaufays, F. (2014). Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv Preprint arXiv:1402.1128*.

Sharma, S. (2017). Epoch vs batch size vs iterations. In *Towards Data Science*. https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9

Sharpe, W. F. (1964). Capital asset prices: A theory of market equilibrium under conditions of risk. *The Journal of Finance*, *19*(3), 425–442.

Starmer, J. (2020, October 19). *Neural Networks Pt. 2: Backpropagation Main Ideas*. https://www.youtube.com/watch?v=IN2XmBhILt4&ab_channel=StatQuestwithJoshStarmer

Takeuchi, L., & Lee, Y.-Y. A. (2013). Applying deep learning to enhance momentum trading strategies in stocks. In *Technical report*. Stanford University.

TITLON Team. (2020). *TITLON – financial data for norwegian academic institutions*. https://uit.no/forskning/forskningsgrupper/sub?sub_id=417205&p_document_id=352767

Van Rossum, G., & Drake, F. L. (2009). *Python 3 reference manual*. CreateSpace.

Zeileis, A., Grothendieck, G., Ryan, J. A., Andrews, F., & Zeileis, M. A. (2014). Package 'zoo.' *R Package Version*, 1–7.

Zhou, F., Zhang, Q., Sornette, D., & Jiang, L. (2019). Cascading logistic regression onto gradient boosted decision trees for forecasting and trading stock indices. *Applied Soft Computing Journal*, *84*, 105747. https://doi.org/10.1016/j.asoc.2019.105747

Appendix

Appendix A - Two-period results for the *k=13* and *k=10* portfolios

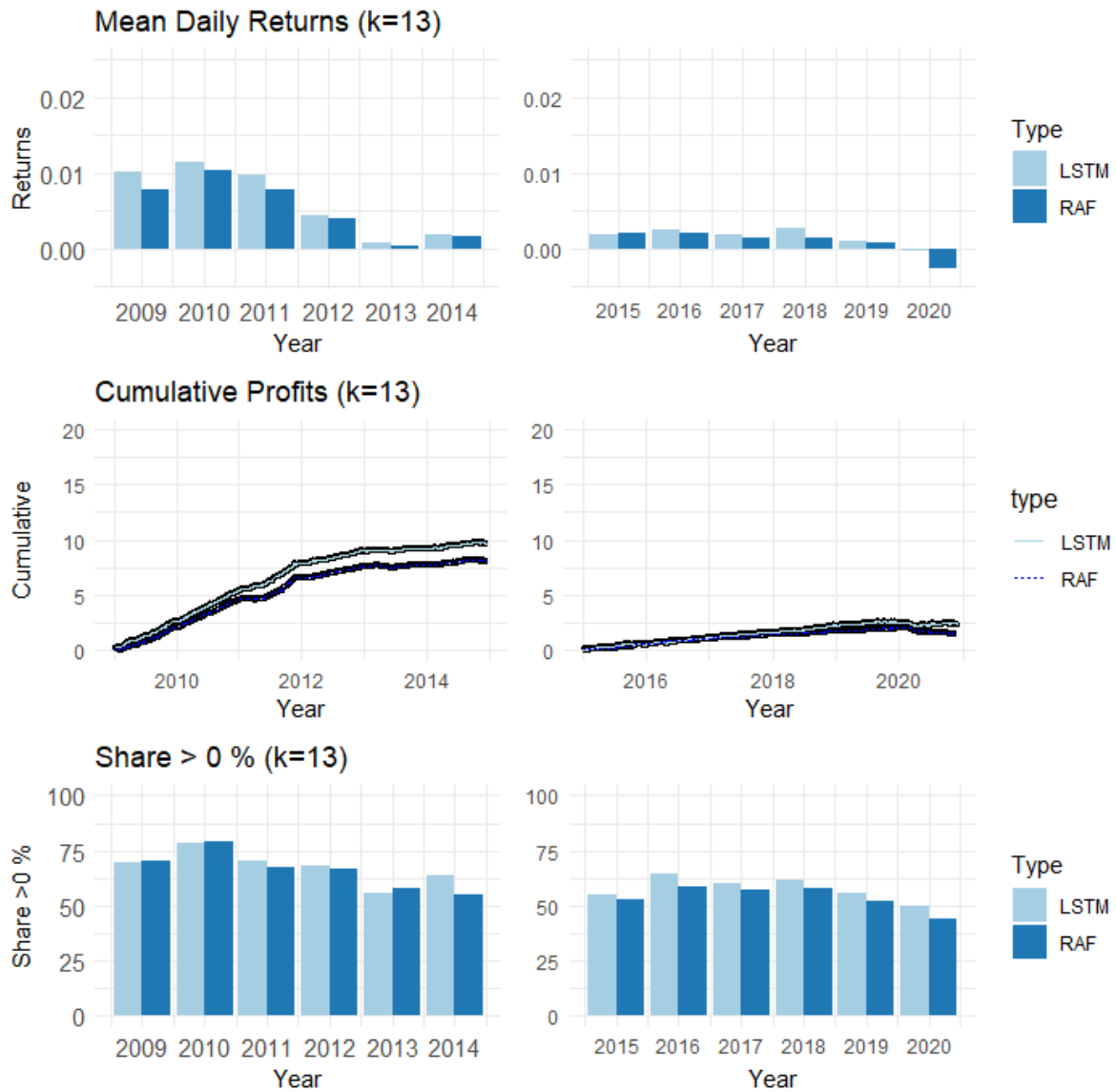


Figure 6.1: Results of the k=13 portfolio before transaction costs

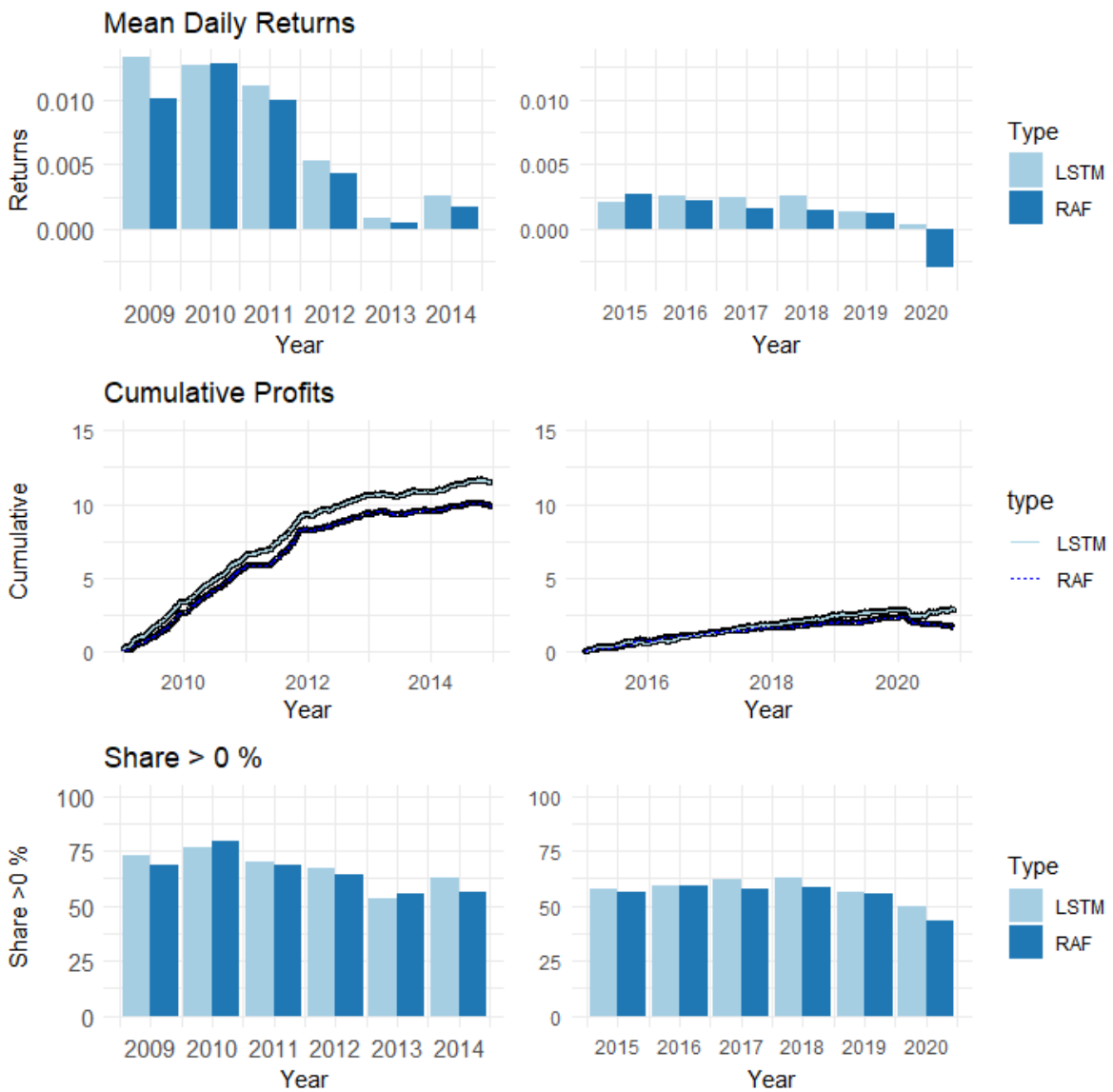


Figure 6.2: Results of the k=10 portfolio before transaction costs

Appendix B - Performance data for the *k=13* and *k=10* portfolios

Table 6.1: Performance Data of the k=10 portfolio before transaction costs

Metric	LSTM	RAF	OSEBX
Mean return (long)	0.003440063	0.002749807	NA
Mean return (short)	0.001375279	0.001124549	NA
Mean return	0.004815342	0.003874356	0.000479708
Standard error	0.0003190041	0.0002864753	0.000237775
t-statistic	15.095	13.524	2.0175
Lowest	-0.141366891	-0.079382848	-0.0917538117
Quartile 1	-0.004279755	-0.005012839	-0.0055810075
Median	0.004287981	0.002960903	0.0008308918
Quartile 3	0.013738886	0.011490819	0.0070181210
Highest	0.137304089	0.155637939	0.0665158453
Share>0	0.626595	0.6040967	0.5337793
Std.Dev	0.01740839	0.01563326	0.01300175
Skewness	-0.1984294	0.6373222	-0.4455084
Kurtosis	5.735306	6.688711	4.215446
5% VaR	-0.02102271	-0.01755973	-0.02029984
1% VaR	-0.04051133	-0.03623617	-0.04026436
Max Drawdown	0.4167586	0.5495453	0.3498688
Annualized returns	2.230411	1.570427	0.1046073
Annualized Std.Dev	0.2763496	0.2481703	0.2063964

Table 6.2: Performance Data of the k=10 portfolio after transaction costs

Metric	LSTM (k=10)	RAF (k=10)	OSEBX
Mean return (long)	0.002440063	0.001749807	NA
Mean return (short)	0.0003752788	0.0001245487	NA
Mean return	0.002815342	0.001874356	0.000479708
Standard error	0.0003190041	0.0002864753	0.000237775
t-statistic	8.8254	6.5428	2.0175
Lowest	-0.143366891	-0.0813828477	-0.0917538117
Quartile 1	-0.006279755	-0.0070128386	-0.0055810075
Median	0.002287981	0.0009609031	0.0008308918
Quartile 3	0.011738886	0.0094908191	0.0070181210
Highest	0.135304089	0.1536379395	0.0665158453
Share>0	0.5698455	0.5339154	0.5337793
Std.Dev	0.01740839	0.01563326	0.01300175
Skewness	-0.1984294	0.6373222	-0.4455084
Kurtosis	5.735306	6.688711	4.215446
5% VaR	-0.02302271	-0.01955973	-0.02029984
1% VaR	-0.04251133	-0.03823617	-0.04026436
Max Drawdown	0.5946016	0.8689869	0.3498688
Annualized returns	0.9549662	0.5548754	0.1046073
Annualized Std.Dev	0.2763496	0.2481703	0.2063964

Table 6.3: Performance Data of the k=13 portfolio before transaction costs

Metric	LSTM (k=13)	RAF (k=13)	OSEBX
Mean return (long)	0.00296656	0.002387043	NA
Mean return (short)	0.001089335	0.0007859702	NA
Mean return	0.004055895	0.003173014	0.000479708
Standard error	0.0002597521	0.0002387086	0.000237775
t-statistic	15.614	13.292	2.0175
Lowest	-0.106764161	-0.062361963	-0.0917538117
Quartile 1	-0.003714226	-0.004021759	-0.0055810075
Median	0.003488089	0.002515325	0.0008308918
Quartile 3	0.011390083	0.009589073	0.0070181210
Highest	0.109048108	0.124368393	0.0665158453
Share>0	0.628274	0.5993956	0.5337793
Std.Dev	0.01417494	0.01302658	0.01300175
Skewness	-0.05049895	0.5621375	-0.4455084
Kurtosis	5.388355	5.536108	4.215446
5% VaR	-0.01698351	-0.01562269	-0.02029984
1% VaR	-0.03142138	-0.03219018	-0.04026436
Max Drawdown	0.3379652	0.4951723	0.3498688
Annualized returns	1.704401	1.175336	0.1046073
Annualized Std.Dev	0.2250203	0.2067906	0.2063964

Table 6.4: Performance Data of the k=13 portfolio after transaction costs

Metric	LSTM (k=13)	RAF (k=13)	OSEBX
Mean return (long)	0.00196656	0.001387043	NA
Mean return (short)	0.0000893	-0.0002140298	NA
Mean return	0.002055895	0.001173014	0.000479708
Standard error	0.0002597521	0.0002387086	0.000237775
t-statistic	7.9148	4.914	2.0175
Lowest	-0.108764161	-0.0643619633	-0.0917538117
Quartile 1	-0.005714226	-0.0060217590	-0.0055810075
Median	0.001488089	0.0005153251	0.0008308918
Quartile 3	0.009390083	0.0075890732	0.0070181210
Highest	0.107048108	0.1223683931	0.0665158453
Share>0	0.560779	0.5184688	0.5337793
Std.Dev	0.01417494	0.01302658	0.01300175
Skewness	-0.05049895	0.5621375	-0.4455084
Kurtosis	5.388355	5.536108	4.215446
5% VaR	-0.01898351	-0.01762269	-0.02029984
1% VaR	-0.03342138	-0.03419018	-0.04026436
Max Drawdown	0.6685311	0.8928981	0.3498688
Annualized returns	0.6361	0.3154665	0.1046073
Annualized Std.Dev	0.2250203	0.2067906	0.2063964