UNIVERSITY OF TURKU

ABSTRACT

	Bachelor's thesis
x	Master's thesis
	Licentiate's thesis
	Doctoral dissertation

Subject	Accounting and Finance	Date	17.11.2020
Author	Jarkko Määttä	Student number	511052
		Number of pages	77 + appendices
Title	Stock market prediction with long short-term memory neural networks: Empirical study on Finnish stock market 1999–2020		
Supervisor	Prof. Mika Vaihekoski		

Abstract

In recent years, advanced machine learning techniques have outperformed previous benchmarks in multiple disciplines, and these methods have also been increasingly applied to stock market prediction tasks. This research aims to fill the research gap in advanced machine learning applications on Finnish stock market by applying long short-term memory (LSTM) neural networks on a stock return movement prediction task for the period between 1999–2020.

The performance of the LSTM network is benchmarked against a conventional recurrent neural network and a logistic regression classifier. Using two alternative sets of input features, the models are trained to produce weekly out-of-sample predictions on stock return movements between 2006 and 2020. Furthermore, these predictions are utilized to derive prediction-based investment portfolios. The best-performing multivariate LSTM model yields an annual return of 12.7% and delivers a Sharpe ratio of 0.459 before transaction costs, while a simple buy-and-hold portfolio achieved an annual return of 8.6% and a Sharpe ratio of 0.338 during the same period. The relative edge of the LSTM-based portfolios holds after transaction costs are considered, but a subperiod analysis reveals that the outperformance is not that eminent during the latter half of the sample.

By unveiling some common characteristics among the stocks selected for trading, the LSTMs are found to independently extract similar patterns to well-known capital market anomalies of short-term mean reversion and momentum. However, the high-level performance of LSTM models cannot be comprehensively explained by these abovementioned effects. The results indicate that the stock returns are partially driven by long-term signals, and that the LSTMs can independently extract this type of subtle information from noisy stock market data.

Despite being relatively complex and having high computational costs, LSTM networks are shown to be suitable methods for stock return movement prediction tasks. Even though the theoretical performance might not fully materialize if the trading strategy is implemented in practice, LSTMs certainly have predictive properties that make them useful tools and complements for different investment purposes.

Key words	Machine learning, forecasting, stock market
Further information	

TURUN YLIOPISTO

TIIVISTELMÄ

	Kandidaatintutkielma
x	Pro gradu -tutkielma
	Lisensiaatintutkielma
	Väitöskirja

Oppiaine	Laskentatoimi ja rahoitus	Päivämäärä	17.11.2020
Tekijä	Jarkko Määttä	Matrikkelinumero	511052
		Sivumäärä	77 + liitteet
Otsikko	Osakemarkkinoiden ennustaminen LSTM-neuroverkon avulla: Empiirinen tutkimus Suomen osakemarkkinoilla 1999–2020		
Ohjaaja	Prof. Mika Vaihekoski		

Tiivistelmä

Viime vuosina koneoppimisen sovellutukset ovat osoittautuneet tehokkaiksi menetelmiksi useilla eri aloilla, ja näitä kehittyneitä menetelmiä on sovellettu yhä enemmän myös osakemarkkinoiden ennustamiseen. Tässä tutkimuksessa sovelletaan LSTM-neuroverkkoja osaketuottojen liikkeiden ennustamiseen Suomen osakemarkkinoilla vuosien 1999–2020 aikana.

LSTM-neuroverkon suorituskykyä verrataan tavanomaiseen takaisinkytkeytyvään neuroverkkoon sekä logistiseen regressiomalliin. Kahdenlaisten eri syötteiden avulla mallit koulutetaan ennustamaan osaketuottojen liikkeitä vuosina 2006–2020, ja näiden ennusteiden pohjalta rakennetaan yksinkertaistettu kaupankäyntistrategia. Parhaiten suoriutuva, useaa syötemuuttujaa hyödyntävä LSTM-neuroverkko yltää 12,7%: n vuosittaiseen tuottoon ja saavuttaa Sharpe-suhdeluvun 0,459 ennen transaktiokustannuksia, kun taas yksinkertainen osta ja pidä -salkku saavuttaa vuotuisen tuoton 8,6% ja Sharpe-suhdeluvun 0,338 samalla tarkastelujaksolla. LSTM-pohjaisten salkkujen suhteellinen etu säilyy transaktiokustannusten huomioon ottamisen jälkeenkin, mutta osittaisperiodikohtainen analyysi paljastaa, että suorituskyky ei ole merkittävästi parempi enää tarkastelujakson loppupuoliskolla.

Tutkimalla LSTM-neuroverkon kaupankäyntiin poimimien osakkeiden yhteisiä piirteitä, havaitaan mallin hyödyntävän samanlaisia kaavoja kuin tunnettuihin markkina-anomalioihin perustuvat strategiat. LSTM-mallien korkean tason suorituskykyä ei kuitenkaan voida selittää kattavasti ainoastaan momentum-teorian tai keskiarvoon palautumisen avulla. Tulokset osoittavat, että osaketuotot ovat osittain pitkäkestoisten signaalien ohjaamia, ja että LSTM-neuroverkot kykenevät itsenäisesti poimimaan tämänkaltaisia signaaleja paljon kohinaa sisältävästä markkinadatasta.

Huolimatta LSTM-neuroverkkojen monimutkaisuudesta ja niiden laskennallisista kustannuksista, tulokset osittavat niiden olevan hyödyllisiä menetelmiä osaketuottojen liikkeiden ennustamiseen. Vaikka teoreettinen suorituskyky ei täysimääräisesti olisi siirrettävissä käytäntöön, LSTM-neuroverkot tarjoavat kuitenkin suotavia ominaisuuksia, jotka tekevät niistä hyödyllisiä työkaluja käytettäväksi erilaisissa sijoitustarkoituksissa.

Avainsanat	Koneoppiminen, ennustaminen, osakemarkkinat
Muita tietoja	

UNIVERSITY OF TURKU
Turku School of Economics

STOCK MARKET PREDICTION WITH LONG SHORT-TERM MEMORY NEURAL NETWORKS

Empirical study on Finnish stock market 1999–2020

Master's Thesis
in Accounting and Finance

Author:
Jarkko Määttä

Supervisor:
Prof. Mika Vaihekoski

17.11.2020
Turku

CONTENTS

FIGURES

TABLES

1 INTRODUCTION

1.1 Background and motivation

The stock market refers to the collection of markets and exchanges where investors and traders buy and sell stocks of companies on a public exchange. Stock market is closely linked with the world of economics and the price of an individual stock is a result of multiple factors such as the financial situation of a company, general economic conditions, political events and investors' sentiment. The ultimate goal of trading in the stock market is to make money by buying securities that are expected to rise in value. Developing efficient market trading strategies and making accurate predictions of stock price movements may therefore yield significant profits for investors. However, stock price modelling and prediction is regarded as one of the most challenging issues among time series analysis since the stock market is essentially a complex, dynamic and somewhat chaotic environment.

True random walk is a stochastic process and does not carry any predictable patterns, and thus any attempts to model it would be pointless. However, there are vast amounts of evidence on different markets that stock returns are not a pure random walk process, and that financial time series may incorporate some sort of hidden patterns. Occasional and somewhat persistent pricing irregularities might occur, e.g., due to irrational investor behavior during high market turmoil (Malkiel 2003). In addition, stock market predictability is argued to be time-inconsistent and to differ considerably between different markets. Many of the methods introduced to exploit these sources of profitability rely on some well-known capital market anomalies, such as mean-reversion and momentum.

Financial time series analysis provides a statistical framework for assessing the behavior of time series, such as asset prices, and the key feature that distinguishes financial time series analysis from other time series analysis is the element of uncertainty. Time series forecasting, in turn, can be generalized as a process that extracts useful information from previously observed values to determine future values. (Tsay 2005.) Traditional statistical methods such as linear regression and autoregressive integrated moving average (ARIMA) models are easy to interpret but they usually require strict assumptions regarding the distributions and stationarity of time series. As the financial market data is known to have nonstationary and noisy characteristics, the forecasting performance of these traditional models is likely to be ineffective unless the exact properties of the time series are

known. Also, the inability of linear models to generalize the key features of the observed data if nonlinear dependencies exist between the variables is a problem since the real world is not linear. (Bao et al. 2017.) These shortcomings have increased interest in more advanced time series analyzing methods such as various applications of machine learning.

For decades, machine learning techniques have been researched and applied in various disciplines, but with the rapid increase in computing power, their potential has increased considerably. In the past two decades advanced machine learning methods have outperformed previous benchmarks, e.g., in natural language processing, rational drug design, image classification and fraud detection. Much of the media attention has been focused on the achievements of deep learning, a subset of machine learning based on artificial neural networks, that provides a group of units for designing network structures according to specific tasks. Artificial neural networks are often considered as having black-box properties and therefore lacking interpretability. (Long et al. 2019.)

Recurrent neural networks (RNN) are particularly useful for time series modelling because they include feedback loops, which enables them to retain information from previous time steps and further utilize this information to predict future targets. However, conventional RNNs have their own limitations, as models' inherent ability to retain data diminishes as the time increases, referred to as the vanishing gradient problem. As a solution to this problem, Hochreiter and Schmidhuber (1997) proposed the long short-term memory (LSTM) architecture which is specifically designed to learn and remember long-term dependencies. Today, LSTM recurrent neural networks and their modifications are some of the leading techniques for sequence learning tasks (Fischer & Krauss 2018).

One of the earliest studies that showed promising results on machine learning applications on stock market prediction tasks, was conducted by White (1988), who uses a simple neural network to detect nonlinear regularities in the daily return of the IBM stock. He fails to find evidence against the efficient market hypothesis as the proposed model is very prone to overfitting, but he shows that even simple neural networks are capable of rich dynamic behavior. In recent years, a growing number of studies have found supporting evidence that machine learning techniques are capable of identifying trading signals in stock market data (see, e.g., Atsalakis & Valavanis 2009, Bahrammirzaee 2010, Fischer & Krauss 2018, Krauss et al. 2017, Nelson et al. 2017).

Krauss et al. (2017) show that a combination of deep neural networks, gradient-boosted trees and random forests outperformed all the individual models, from which the deep neural networks, in fact, achieves the lowest performance. Their ensemble method

-based trading strategy generates a Sharpe ratio of 1.81, five times higher than that of the general market. Fischer and Krauss (2018) expand the recent work of Krauss et al. (2017) and apply a long short-term memory network for stock prediction task, which further improves the performance of the best model suggested by Krauss et al. (2017). Unlike the deep learning model by Krauss et al. (2017), their LSTM network clearly outperforms traditional machine learning and statistical models, also having generally lower exposure to common sources of systematic risks compared to the benchmark models. Both Krauss et al. (2017), as well as Fischer and Krauss (2018), find similar patterns of diminishing returns during the recent decades, but even though the advanced machine learning -based strategies might not show profitability during the recent years, they certainly have predictive properties that make them useful tools and complements for different trading strategies and tasks.

Despite the rapid development in the field of machine learning, academic work on financial time series prediction is fairly limited compared to many other disciplines. This is not particularly surprising given the complexity of financial markets. Varian (2014) also argues that overall, theoretical advances in the field have struggled to incorporate into practice. Financial predictive models, and the investment strategies derived from them, might fit well on historical data but are likely to fail the future. Lack of model interpretability, as well as incentives to maintain profitable methods as secrets, might also explain the lack of publications on this topic, but the promising results achieved in previous literature highlights the need for more extensive research. Also, most of the previous empirical work has been conducted on the U.S. and other foreign markets, and therefore this research aims to fill the research gap in advanced machine learning applications on Finnish stock market.

1.2 Objectives and structure

The main objective of this research is to examine the feasibility and performance of long short-term memory neural networks on the problem of stock market return movement prediction task. In addition, the empirical results will be analyzed with an objective to gain better understanding about the dynamics of the black-box machine learning methods and their implications. Moreover, this research examines how the predictive performance translates into real world performance when a simple trading strategy based on model

predictions is constructed and evaluated with and without transaction costs. Therefore, to some extent this research will contribute to the theory on market efficiency in Finland.

For the empirical analysis, fifteen stocks with adequate liquidity and data availability from the Finnish stock market are selected, and the necessary data for each stock and for all the predictive variables are obtained from the Refinitiv Eikon Datastream. The data sample reaches from the 4th of January 1999 to the 8h of June 2020, a time span that provides a full data availability and a sufficiently large sample size. In this research, the predictions, portfolio construction and empirical analysis are conducted with a weekly observation frequency in order to save computational burden, avoid redundant noise and to capture longer-term dependencies in the data.

The research questions are investigated through an empirical analysis as follows. First, a single predictive LSTM neural network model is constructed, and in order to validate the complexity of the proposed model, a simple recurrent neural network model (RNN) and a standard logistic regression classifier (LOG) are built to use as benchmarks. The predictive models in this research are also compared by using two different sets of input patterns in order to analyze the importance of additional features to the predictive performance. In the first approach, called the univariate approach, only the historical return patterns are used as inputs for the models. In the second one, called the multivariate approach, the past return pattern is complemented with a set of predictive features that are chosen based on previous research on the predictability of stock market returns. The prediction phase is approached as a binary classification problem, as the models are first trained with the input data to generate a conditional probability that a stock will outperform the cross-sectional median return on the following week. The generated outperforming probabilities are further used to construct a simple prediction-based long trading portfolio and to adjust its weight accordingly each week.

The different models are compared in terms of accuracy, profitability and stability to assess if the proposed LSTM network presents any improvements when compared to the benchmark models. Predictive accuracy is evaluated by using different measurements such as the classification accuracy and binary cross-entropy loss function. In addition, a statistical test proposed by Diebold and Mariano (1995) is implemented to evaluate if the model accuracies are statistically different from each other. The models are also validated regarding their financial performance by building a simple prediction-based long trading strategy and further comparing it with a simple buy-and-hold portfolio. The performance metrics used in the comparison of financial performance include, e.g., annualized mean

return, standard deviation, Sharpe ratio and maximum drawdown before transaction costs. Due to the implementational issues regarding liquidity, lending availability and the transaction costs they produce, short trading strategies are not considered in this research. The amount of direct transaction costs is assumed to be five basis points as suggested by Krauss et al. (2017) and Fischer and Krauss (2018). To gain better understanding about the dynamics of the black-box methods, the data is visualized and thoroughly analyzed in order to capture the quintessence of the patterns the LSTM acts upon for selecting the stocks for trading.

The remainder of this paper is organized as follows. Section 2 focuses on the theoretical background of this research and introduces the basic concepts regarding the stock market predictability and financial time series analysis. Next, Section 3 covers artificial neural networks and introduces the basic structures and typical properties of different network architectures, from feedforward and recurrent networks to long short-term memory cells. In addition, an introduction to the network training process is provided, and main findings on previous literature are discussed. Section 4 presents the research methods and provides details about the data preprocessing, feature and target generation, model construction, network training, performance evaluation and portfolio construction. The experimental results are introduced in Section 5, with the main emphasis on the interpretation and practical implementations of the predictions. Finally, Section 6 summarizes the main findings of this research and discusses possible topics left for future research.

First parts of the data preparation and handling are conducted with Excel.[1] Further data handling and examination is executed in Python 3.6.8.[2] The deep learning RNN and LSTM networks are developed with Keras (Chollet 2015) on top of TensorFlow library (Abadi et al. 2015). Moreover, the logistic regression model is built using the Scikit-learn package (Pedregosa et al. 2011). Data visualization is implemented using RStudio.[3]

---

1 Microsoft Corporation (2020) Microsoft Excel. <https://office.microsoft.com/excel>

2 Python Software Foundation (2020) <https://docs.python.org/3.6/>

3 RStudio Team (2020) <http://www.rstudio.com/>

2 THEORETICAL BACKGROUND

2.1 Predictability of stock market returns

2.1.1 Stock market efficiency

Stock market is a set of exchanges where stocks of publicly listed companies are issued, bought and sold. The main objective of trading in the stock market is to generate profits by buying securities that are expected to rise in value. The field of stock market prediction focuses on developing well-defined trading strategies and prediction approaches in order to achieve high returns on investments. However, stock market is a complex and dynamic environment that is affected by numerous interacting political, social, environmental and economic events. The underlying relationships between these factors are not well known, and moreover, these relationships are dynamic and tend to change over time. Therefore, stock price modelling and prediction is considered a challenging issue and there is a large body of conflicting findings in the literature about whether some predictability exists in the stock market.

Many stock market theories and principles are based on the assumption of rational investor behavior. One of the most quoted theories that aims to explain the investment decision-making and stock price behavior is the Efficient Market Hypothesis presented by Fama (1970). According to the theory, security prices at any point in time fully reflect all available information, and therefore consistently beating the market, i.e. generating risk-adjusted excess returns, is impossible. The theory suggests that it is pointless to utilize fundamental or technical analysis to predict market trends or search for undervalued stocks, as stocks always trade at their fair value on the market and the potential for surpluses is exploited as soon as they occur. Since new information occurs unpredictably and the market participants make use of all this information optimally, stock price variations are random in nature. Also, different interpretations of new information by investors do not make the market inefficient unless someone is able to systematically exceed market returns with their investments. In an efficient market, the market prices of securities can significantly deviate from their real values, but the changes are unpredictable, i.e., follow a random walk (Malkiel 2003).

In his paper, Fama (1970) categorize tests of efficiency into weak form, semi-strong form, and strong form, in order to determine the level of information at which the hypothesis is no longer valid. Weak form efficiency claims that securities prices reflect all

historical information, and there is no potential for outsized risk-adjusted returns through technical analysis. In a market that meets the semi-strong conditions, it is also not possible to obtain excess returns by utilizing fundamental analysis, as the prices of securities also include all publicly available information about the company. In a market operating under strong conditions, the prices of securities reflect not only historical and all publicly available information, but also all unpublished information about the company, i.e. insider information. There is considerable amount of evidence in the literature advocating that the weak form and semi-strong form of the theory holds in the stock market. For example, Malkiel (2003) argues that the generally poor performance of active mutual funds is an indicator of at least semi-strong market efficiency.

The efficient market hypothesis has also received a lot of criticism from both investors and researchers, e.g. for its underlying assumptions, as the hypothesis does not consider trading costs, taxes, and the resulting inefficiencies. In addition, all market participants should interpret new information in the same way, which in practice, never corresponds to reality. Grossmann and Stiglitz (1980) argue that securities prices cannot effectively reflect all available information because the acquisition and production of information incurs costs. If the information were disclosed to all market participants according to the efficient market hypothesis, data collection would be worthless, and this would lead to a lack of incentives to trade. As a result, there is a persistent information imbalance in the market, with individuals acquiring information being compensated for their efforts. In addition to the compensation of information acquisition, Malkiel (2003) argues that irrational behavior of investors might result to occasional and somewhat persistent pricing irregularities, and thus predictable patterns in stock returns. This fights against the idea of rational investor behavior, and at the same time advocates the existence of various market anomalies.

Stock market information is gradually becoming more easily available at an even lower cost, and intensified competition for customers between financial service providers has also reduced transaction costs considerably in the last decade. Due to wide and easy access to information, as well as large number of actors, stock market is often considered to be more efficient than many other markets. However, a market that completely meets the information efficiency requirements is still unlikely to exist. For example, several studies have shown that there is significant time variation in predictability, and that stock market in smaller countries tend to be less efficient compared to bigger markets such as in the U.S. (see, e.g., Farmer et al. 2019, Kara et al. 2011, Nelson et al. 2017).

2.1.2 Variables affecting stock returns

In their comprehensive study, Welch and Goyal (2008) review the performance of predictive variables that have been previously suggested by earlier academic research to be prominent predictors of the equity premium. They perform a systematic examination on both in-sample and out-of- sample performance of different statistical methods using various predictors. The authors argue that the predictability of stock returns found in the previous literature is time-inconsistent and does not hold when new data is introduced. Most of the previously suggested variables with predictive properties failed to deliver consistent out-of-sample results, despite the possibly promising in-sample abilities. This supports the argument that a naive forecast based on the historical mean is as good as any other method for the prediction of the equity premium. Welch and Goyal (2008) also point out that even if a proposed model would be powerful in a research perspective, i.e., has good both in-sample and out-of-sample performance, great confidence on the model is required in order to use it for investing purposes in the long-term.

Neely et al. (2014) present the opposite results to Welch and Goyal (2008) by reporting statistically and economically significant in-sample and out-of-sample predictability in the U.S. equity risk premium. They argue that the predictive power of technical indicators is at least as good as that of macroeconomic variables, both of which detect different information over the business cycle. More accurately, macroeconomic variables better capture the typical surges in the equity risk premium, whereas technical indicators better pick up the typical drops. They also show that combining technical indicators and macroeconomic variables improves both in-sample and out-of-sample performance of the equity risk premium predictions. The most prominent macroeconomic variables in the research included, e.g., dividend yield, book-to-market ratio and the Treasury bill rate. In addition, volume-based indicators were among the most influential technical indicators. The results of Neely et al. (2014) are in line with Blume et al. (1994), who show that volume provides important information that is not yet captured in the market price and is thus a useful tool when modelling the process of security returns.

Several different macroeconomic factors have been suggested in the previous literature to be prominent features explaining stock returns. Some of the most examined factors include long- and short-term interest rates. Qi and Maddala (1999) forecast S&P 500 index returns and use the one-month Treasury bill rate as a predictive feature in their model. They show that interest rates have significant negative correlation with stock market

returns during 1954 –1992. In addition to the effects of macroeconomic factors on stock returns, they show that dividend yield has a positive relationship with excess returns.

Brennan et al. (1998) study factors affecting the monthly returns of common stock companies in the Nasdaq index between 1966 and 1996. They argue that past returns, firm size and book-to-market ratio have strong relations on excess stock returns. In addition, similar to Neely et al. (2014) and Blume et al. (1994) they find the trading volume to have significant negative impact on the average returns.

Er and Vuran (2013) implement a dynamic panel-data analysis to explain the main factors affecting stock returns of several manufacturing firms listed in the Istanbul Stock Exchange between 2003 and 2007. Their findings show that past returns, profitability ratios, firm size and financial activity all have significant effect on stock returns. In addition, macroeconomic factors such as economic growth, exchange rates and oil prices can also be used to explain stock returns. Similar to the findings of Qi and Maddala (1999), they also show that interest rates have significant negative impact on stock returns. Moreover, the exchange rate, firm size and profitability ratios have positive correlation with stock returns. The positive impact of exchange rates on stock returns is also shown by Niaki and Hoseinzade (2013) who utilize design of experiments to determine which factors among 27 financial and economic variables have significant influence on the daily direction of the S&P 500 index. They show that the exchange rates between the U.S. dollar and three main currencies to be among the most influential features. In contrast to Neely et al. (2014) and Blume et al. (1994), they argue that relative change in the trading volume does not have significant impact on the daily forecasts on the S&P 500 index.

Gu et al. (2018) conduct a comparative analysis of different statistical learning methods on the task of stock return prediction. Their predictive dataset consists of over 900 variables, including 94 stock-specific characteristics. They discover the subtle relationships between the features and expected returns by examining each variable at a time and further controlling other variables at their means. Gu et al. (2018) show that variations in momentum, liquidity and volatility provide the most dominant predictive signals among all considered features. The most powerful predictors are also associated with recent price trends and exhibit characteristics of short-term reversal and momentum.

In addition to the vast amount of predictive variables presented in the previous literature, several studies have shown that return predictability tends to shift across time and across different markets, with different factors having effect at different times (see, e.g., Farmer et al. 2019, Fischer & Krauss 2018, Krauss et al. 2017).

2.1.3 Predictability of return movement

Instead of estimating the level of stock or index returns, another part of research focuses on the predictability of return movement. Given a sequence of movements of a stock over time, the goal is to predict whether returns outperform a certain threshold, e.g., zero or the cross-sectional mean of returns in the future. The empirical results of Leung et al. (2000) favor the use of classification models over level estimation models. They argue that classification models provide better results compared to level estimation models in terms of prediction accuracy, as well as in terms of trading profitability. In addition, trading strategies based on classification models more often outperform simple buy-and-hold strategies in terms of risk-adjusted profits.

Mean reversion is a financial theory that suggests that asset prices and returns tend to revert to their long-term average level over time. The theory focuses on reversion from abnormal deviations from the long-term mean, as growth and small fluctuations are typical features for asset price development. Poterba and Summers (1988) investigated the mean-reverting behavior in stock prices and found that transitory price components explain significant amount of stock return variations. Their results show that stock returns show positive serial correlation over short periods and negative correlation over longer periods. This supports the existence of pricing irregularities and stock market predictability, as investing to recently underperforming stocks would yield higher returns compared to recently overperforming stocks.

The momentum effect is a market anomaly that refers to the tendency of asset prices to follow their recent trend. In other words, stocks with strong performance in the past will continue to perform well in the future. Momentum investing is a strategy exploiting the momentum effect by buying stocks with high recent returns, known as winners, and taking a short position on stocks with poor recent returns, known as losers. Different momentum strategies can use variety of methods to identify these winners and losers. (Daniel & Moskowitz 2016.) If the markets are truly efficient as the Efficient Market Hypothesis suggests, the momentum-based strategies should not help in pursuit of returns higher than the risk-adjusted market return. However, previous literature shows support to the profitability of momentum investing.

In one of the first studies on the momentum effect, Jegadeesh and Titman (1993) find evidence of U.S. stock prices exhibiting momentum during the time period of 1965–1989. They show that a momentum-based strategy of buying stocks with high returns in the

previous 3–12 months, and short-selling stocks with low returns during that period yields substantial abnormal returns over the study period. Jegadeesh and Titman (1993) argue that the profitability of momentum strategies does not derive from their exposure to systematic risk factors, but rather is a result of cognitive biases and irrational investor behavior, at least to some extent. Hong et al. (2000) show that momentum investing returns are highest for small stocks with lower analyst coverage. The authors also note that due to short-sales constraints and manager incentives, stocks with low analyst coverage tend to react more quickly to good news than to bad news. According to Daniel and Moskowitz (2016), traditional momentum strategies are vulnerable to occasional, but persistent periods of negative returns that often occur during high market turmoil. Therefore, the average returns and Sharpe ratios of momentum strategies suffer, as it can take years to recover from these infrequent crashes.

2.2 Financial time series analysis

Financial econometrics can be defined as the application of statistical techniques to quantitative problems arising from finance. These econometric techniques can be useful in, e.g. determining asset prices or returns, studying relationships between features, and forecasting future values of financial variables. (Brooks 2014.) Financial time series analysis is a part of financial econometrics, that revolves around the theory and practice of asset valuation over time, and it aims to extract meaningful information from the data in order to understand the dynamic dependence of financial phenomena. The key feature that distinguishes financial time series analysis from traditional time series analysis is the element of uncertainty. Time series approach to forecasting, in turn, includes collecting and analyzing historical observations in order to determine future values by developing a model which aims to capture the underlying data-generating process. (Tsay 2005.)

Time series analysis can help to understand the past, and if time series observations are assumed to contain information about the future development of a variable, some function $f(\cdot)$ of the past observations can be found to make predictions one or multiple time steps ahead. Let y_t denote the value of a variable at time t. A univariate, single time step $t+1$ ahead prediction made at the end of a period t can therefore be expressed as

$$\hat{y}_{t+1} = f(y_t, y_{t-1}, \dots, y_{t-h}),$$

where h denotes the distance to the earliest observation included in the prediction. In many applications, especially in financial time series analysis, the values of a variable are not only dependent on its past values, but also on the past values of several other variables. This additional information can be utilized to produce a multivariate forecast $\hat{y}_{1,t+1}$ as the function of a multiple time series expressed as

$$\hat{y}_{1,t+1} = f_1(y_{1,t}, y_{2,t}, \dots, y_{k,t}, y_{1,t-1}, y_{2,t-1}, \dots, y_{k,t-h}),$$

where $y_1, y_2, \dots y_k$ are the related variables. In contrast to a univariate time series data that contains only a single time-dependent variable, a multivariate time series data consists of multiple time-dependent variables. Multivariate time series analysis is typically used to model and explain interdependencies and co-movements among different variables, and to obtain useful predictions about their future development. (Lütkepohl 2005.)

Traditional linear time series applications, such as linear regression or autoregressive moving average (ARMA) models, have dominated the field for decades mostly due to their simple structure and good interpretability. The vast majority of traditional financial analysis and prediction models rely on strict underlying assumptions, such as normality and stationarity of time series, which often do not apply in reality. Some methods require variables to be normally distributed in order to make accurate conclusions, but the violation of this assumption does not usually lead to significant inefficiency of the models. However, in most cases the data are required to be stationary, i.e. the statistical properties of the time series, such as mean and variance, should not change over time. Stationary processes are easier to analyze and predict, as the past statistical properties will remain the same in the future. A nonstationary time series is therefore often rendered through transformations to become approximately stationary. (Tsay 2005, Tsay & Chen 2019.)

Traditional linear models are often preferred due to their simplicity, but they are limited in many respects. For example, conventional financial models are not able to account for nonlinearity that is common for real-world phenomena, such as in questions related to the stock market. Clements et al. (2004) argue that many financial time series indeed exhibit nonlinear behavior. They note that different types of nonlinearity exist, as some variables only have occasional predictive power, whereas others might show self-exciting or catastrophic behavior. As the financial market data is known to have nonstationary and noisy characteristics, the forecasting performance of traditional linear models is likely to

be ineffective unless the exact properties of the time series are known. (Rozenberg et al. 2012.) In addition, issues such as collinearity and decreasing degrees of freedom prevent these traditional models from employing large input patterns for predictive tasks. According to Varian (2014), conventional linear econometric tools are often enough to make statistical inference in practice as they provide good approximations, but some phenomena are unique to large datasets and alternative, more flexible tools are required. Tsay and Chen (2019) argue that many times nonlinear models are able to make significantly better contributions, particularly if large amount of data is available. Given the uncertain nature of stock market and the limitations of traditional econometric tools, there is room for more advanced models that better recover the underlying data-generating process and capture the nonlinear relationships in the data without prior knowledge of the statistical properties or nonlinearities in the input data. (Rozenberg et al. 2012.)

Nonlinear methods of time series analysis take into account at least some degree of nonlinearity in the data, and thus allow modelling more complex phenomena where the dynamic dependencies between variables are not known beforehand. Despite its long history in the literature, the performance of nonlinear forecasting models was not that advantageous yet in the early 2000s (see, e.g., Clements et al. 2004, De Goojier & Kumar 1992). According to Clements et al. (2004), the relatively poor performance of nonlinear models was mostly due to the inadequacy of the models, as well as linear models being reasonable approximators with less complexity. However, the sharp increase in computational power, along with the overall development in the field has expanded the usability of nonlinear models significantly.

One of the most popular nonlinear methods are neural networks, that can be classified as semi-parametric statistical procedures, i.e. they can have both parametric and nonparametric components (Tsay 2005). Unlike most traditional statistical methods, neural networks do not require many restrictive assumptions on the underlying data-generating mechanism. For example, the stationarity of time series does not have an impact on the predictive power of neural networks. This feature makes neural networks less prone to model misspecification. Usually the underlying process is unknown, and therefore the ability of neural networks to learn from data is a valuable feature. Neural networks are also relatively robust to noise in the input data. (Brownlee 2018.) In contrast to traditional linear models that can by definition represent solely linear functions, one of the most powerful characteristics of neural networks is their ability to represent any function, known as the universal function approximation property. The theorem states that a

feedforward network with at least a single hidden layer containing a finite number of neurons can approximate any continuous function to an arbitrary degree of accuracy. This feature makes neural networks general and flexible tools especially for prediction tasks. (Goodfellow et al. 2016.)

Despite all the favorable characteristics, neural networks have also received some criticism. Brooks (2014), among others, points out that the lack of diagnostics or specification tests is one of the major disadvantages of neural networks. The parameter values and coefficients in neural networks do not have meaningful theoretical interpretations and therefore, there is no clear manner to evaluate the adequacy of the model. In other words, since there is no clear link between the model weights and the function f being approximated, the approximation given by the network does not provide any insights to the form of that function. This is why neural networks are often described as black boxes. In addition, there is no theoretical basis to determine the optimal network architectures or hyperparameters, which might easily lead to overfitting or to otherwise non-optimal performance. Neural networks are also data-driven models and their success is heavily dependent on the patterns represented by the input variables, and also the analysis of which characteristics in the input data are irrelevant also remains an open issue (Rozenberg et al. 2012).

3 ARTIFICIAL NEURAL NETWORKS

3.1 Feedforward networks

The concept of artificial neural networks has drawn inspiration from simplified models of the brain and natural neural networks. However, the complex working principles of the brain differ significantly from the operations of computers, so detailed modeling is challenging. According to Goodfellow et al. (2016), brain function is simply not yet well enough known to be used as a comprehensive guide for the machine learning algorithms. They point out that today, the research and development of neural networks differs considerably from neurobiology and computational neuroscience. The structure of artificial neural networks can still be described in very same terminology as natural neural networks.

A neural network is an interconnected set of computational units called neurons or nodes, that uses a specific training or learning approach to identify the fundamental functional relationships or patterns in a set of data. Warren McCulloch and Walter Pitts presented a simple definition of an artificial neuron back in 1943, and it has been used as the basis for later neural network structures. The following introduction to model of a neuron follows the description of Haykin (2009). A nonlinear model of a neuron can be defined as an information-processing unit that produces a linear mapping of the input data and further defines the output through a nonlinear activation function. This type of neuron can be expressed by the following equation

$$\boldsymbol{y} = \varphi\left(\boldsymbol{W}_{xy}\boldsymbol{x} + \boldsymbol{b}_y\right),$$

where $\boldsymbol{y}$ is the output vector, $\varphi(\cdot)$ is an elementwise nonlinear activation function, $\boldsymbol{W}_{xy}$ is a weight matrix between the input and output layers, $\boldsymbol{x}$ is the input vector, and $\boldsymbol{b}_y$ is an output bias vector. When the sum of the neuron inputs equals to zero, the bias vector allows the neuron output signal to deviate from zero.

The activation function $\varphi(\cdot)$ limits the neuron output signal to the desired value range depending on the function used, and further transfers the weighted activation from one layer to another. The choice of an appropriate activation function depends on the nature of the problem to be solved and the need to describe the nonlinearity of the underlying phenomenon. The function can be, for example, a logistic sigmoid activation function

$$sigmoid(x) = \frac{1}{1+e^{-x}}, \quad (1)$$

that squashes the values to the open interval $(0, 1)$, drawing close to zero when x becomes very negative, and close to one when x becomes very positive. Another commonly used activation function is a hyperbolic tangent activation function

$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}, \quad (2)$$

that is bound to the range $(-1, 1)$. However, the activation function must be of a nonlinear type for the model to be able to approximate any continuous function with arbitrary accuracy, i.e., have the universal approximation capability. (Goodfellow et al. 2016.)

One of the quintessential examples of neural network structures is the feedforward network or multilayer perceptron (MLP), which consists of one or more hidden layers in addition to the input and output layers. It can be defined as a mathematical function that is formed by composing multiple simpler functions to map some set of inputs to outputs. In a feedforward neural network, the neurons are arranged in layers and the connections extend only from the previous layer to the next, so that the neurons in the same layer are never interconnected. As a result, the network structure is static, i.e., it has no so-called memory and its output depends only on the values of the inputs. Traditional feedforward neural networks are particularly suitable to problems where mainly the relationship between input data and targets is examined, but they cannot make use of sequential information very well.

The structure of a two-layer MLP, i.e., a feedforward network with one hidden layer and a linear output layer, is illustrated in Figure 1, where x_i are the input features, h_n are the nodes in the hidden layer and y is the network output. The arrows signify the direction of information flow and represent the weighted connections between the layers. Here, the information goes through the inputs to hidden nodes and further to the output, so that the information flow is one-directional and there is no feedback included in the model. (Goodfellow et al. 2016.)

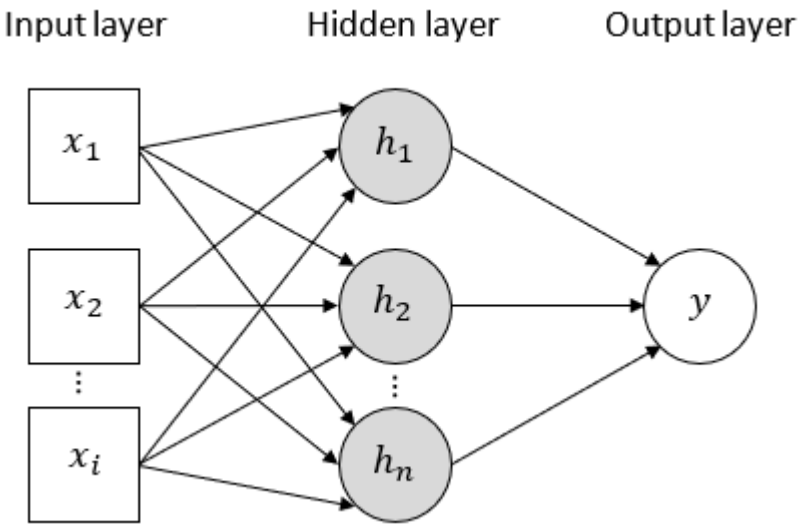


Figure 1. Two-layer multilayer perceptron (MLP)

A neural network with no hidden layers and only a single output layer using a logistic sigmoid activation function, is essentially the same as a logistic regression model. However, the addition of a single layer of hidden units converts the model into a two-layer MLP. (Goodfellow et al. 2016.) The two-layer MLP that is presented in Figure 1, can be written by the following equations

$$\begin{aligned} \boldsymbol{h} &= \varphi(\boldsymbol{W}_{xh}\boldsymbol{x} + \boldsymbol{b}_h) \\ \boldsymbol{y} &= \boldsymbol{W}_{hy}\boldsymbol{h} + \boldsymbol{b}_y, \end{aligned} \tag{3}$$

where $\boldsymbol{h}$ is a vector representing the activations of the hidden layer, $\varphi(\cdot)$ is an elementwise nonlinear activation function, $\boldsymbol{W}$ terms denote the weight matrices, $\boldsymbol{x}$ is the input vector, $\boldsymbol{b}$ terms are bias vectors and $\boldsymbol{y}$ is the output vector. In this example the output vector is a linear combination of the weighted hidden activations and the output bias.

To some extent, MLPs can be used for sequence prediction problems, but they suffer some key limitations that highlight the need for more adequate models for time series analysis tasks. They require, e.g., that the temporal dependencies between variables are specified beforehand in the design of the model, and that the sequences are set to be of fixed length. (Brownlee 2018).

3.2 Recurrent networks

Unlike the feedforward networks, recurrent neural networks or RNNs contain at least one feedback loop, whereby each node transmits its output back to all other nodes. These feedback loops allow the models to store data and process long sequences of input values, and thus make them more useful tools for time series analysis. One of the fundamental features of RNNs is parameter sharing, which enables them to process data with various lengths and generalize across them. The structure of a recurrent network makes it possible to use the same transition function with same parameters at every time step. Since the same weights and parameters are shared for multiple time steps, RNNs can learn a single model that can also be applied to sequence lengths that were not used during the training process. (Goodfellow et al. 2016.)

The following description of the structure of a standard recurrent network is based on the work of Graves et al. (2013). Using an input vector sequence $\boldsymbol{x} = (x_1, x_2, \dots, x_T)$ and iterating through time from $t = 1$ to T, the model computes the vector sequence for the hidden state $\boldsymbol{h} = (h_1, h_2, \dots, h_T)$ and vector sequence for the output $\boldsymbol{y} = (y_1, y_2, \dots, y_T)$ by the following equations

$$\begin{aligned} \boldsymbol{h}_t &= \varphi(\boldsymbol{W}_{xh}\boldsymbol{x}_t + \boldsymbol{W}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_h) \\ y_t &= \boldsymbol{W}_{hy}\boldsymbol{h}_t + \boldsymbol{b}_y, \end{aligned} \tag{4}$$

where $\boldsymbol{h}_t$ is the hidden vector at time t, $\varphi(\cdot)$ is an elementwise nonlinear activation function, $\boldsymbol{W}$ terms denote the weight matrices, $\boldsymbol{x}_t$ is the input vector at time t, $\boldsymbol{b}$ terms are bias vectors and $\boldsymbol{y}$ is the output vector. The structure of a standard recurrent network is also presented in Figure 2, where the arrows represents the weight matrices and biases between layers. On the left side of Figure 2, the network is presented as a circuit diagram where the black square indicates a delay of a single time step. The same network is also presented as an unfolded computational graph on the right side of Figure 2, where each node of the network is associated with an individual time step. The information is processed from the input $\boldsymbol{x}$ by incorporating it into the hidden state $\boldsymbol{h}$, from where it is passed forward through time and given as an output $\boldsymbol{y}$ at each time step t.

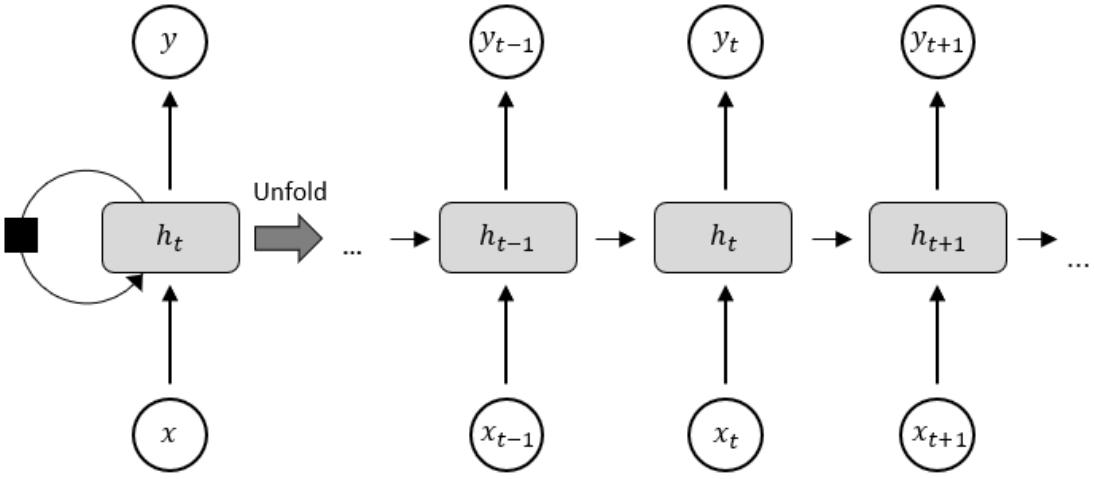


Figure 2. Recurrent neural network

Standard recurrent neural network is essentially a set of recurrently connected computational units that have some sort of simple structure, like a single sigmoid or tanh activation function. The structure of RNN is therefore very similar to MLP, as can be observed by comparing Equation 3 and Equation 4, the recurrent connections making the essential distinction. The recurrent layer repeatedly applies the same operation at each time step, and the layer output is copied and returned to the network as input. Therefore, in the hidden state equation of RNN, in addition to the current input $\boldsymbol{W}_{xh}\boldsymbol{x}$, also the previous output $\boldsymbol{W}_{hh}\boldsymbol{h}_{t-1}$ is analyzed in each round of learning. The repeated application of same function with the same parameters leads to certain difficulties, such as the problem of vanishing and exploding gradients (Goodfellow et al. 2016). Although standard RNNs can handle time series better than feedforward networks, due to the vanishing or exploding gradients they fail to capture dependencies spanning more than ten discrete time steps (Gers et al. 2000).

The problem of vanishing and exploding gradients refers to a shortcoming in the training of neural networks, where the backpropagated error signal, i.e., the influence of a given input, either vanishes exponentially or blows up to infinity while it cycles through the recurrent connections. The nature of this problem depends on the choice of the activation function. Vanishing gradient problem might arise with activation functions that squash the input space into a small output range, while exploding gradients might occur when the derivatives can take on larger values. Vanishing gradients make the learning process more difficult, since adjusting parameters to the correct direction becomes increasingly challenging and eventually the weight updates are so small as to have no effect.

In the worst case, the vanishing gradients might completely block the model from further training. This problem becomes worse as the number of hidden layers and number of time steps in the model increases. Exploding gradients, on the other hand, make the learning process more unstable as the weight updates become larger, and this can make the model effectively unable to learn from the training data. (Brownlee 2018, Goodfellow et al. 2016.)

3.3 Long short-term memory cells

Long short-term memory is regarded as one of the most successful recurrent neural network architectures. It was first introduced by Hochreiter and Schmidhuber (1997) as a solution to the problem of vanishing and exploding gradients. The computational unit of the LSTM is commonly referred to as a memory cell, and it is specifically designed to learn and remember long-term dependencies. Instead of a single layer in the computational unit, LSTMs are typically composed of three multiplicative layers or gates that regulate the information flow through the memory cell: an input gate, a forget gate and an output gate. These gates are weighted functions that have ability to add or remove information from the memory cell state.

The following introduction to LSTM networks follows the descriptions of Graves et al. (2013) and Goodfellow et al. (2016). Instead of units that compute only a nonlinear transformation of inputs and recurrent units, each LSTM cell includes an internal recurrence in addition to the step-to-step recurrence of RNN. To apply LSTM cells into a recurrent network, the activation function $\varphi(\cdot)$ in Equation 4 is now replaced by the following composite function

$$\begin{aligned}
\boldsymbol{f}_t &= sigmoid(\boldsymbol{W}_{xf}\boldsymbol{x}_t + \boldsymbol{W}_{hf}\boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \\
\boldsymbol{i}_t &= sigmoid(\boldsymbol{W}_{xi}\boldsymbol{x}_t + \boldsymbol{W}_{hi}\boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \\
\tilde{\boldsymbol{c}}_t &= \tanh\left(\boldsymbol{W}_{x\tilde{c}}\boldsymbol{x}_t + \boldsymbol{W}_{h\tilde{c}}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{\tilde{c}}\right) \\
\boldsymbol{o}_t &= sigmoid(\boldsymbol{W}_{xo}\boldsymbol{x}_t + \boldsymbol{W}_{ho}\boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \\
\boldsymbol{c}_t &= \boldsymbol{f}_t\boldsymbol{c}_{t-1} + \boldsymbol{i}_t\tilde{\boldsymbol{c}}_t \\
\boldsymbol{h}_t &= \boldsymbol{o}_t\tanh(\mathbf{c}_\mathrm{t}),
\end{aligned}$$

where $\boldsymbol{f}$, $\boldsymbol{i}$, $\boldsymbol{o}$, $\boldsymbol{c}$ and $\boldsymbol{h}$ are respectively the forget gate, input gate, output gate, cell state and hidden state vectors, and $\tilde{\boldsymbol{c}}$ denotes the candidate values to be potentially added to the cell states. All the weight matrices $\boldsymbol{W}$ from the cell to gate are diagonal. The structure of

the LSTM cell in a recurrent network is also illustrated in Figure 3, where ⊗ and ⊕ denote pointwise operations. All three gating units use the nonlinear sigmoid activation function, but the candidate unit can have any nonlinear squashing activation function, e.g., a hyperbolic tangent activation function.

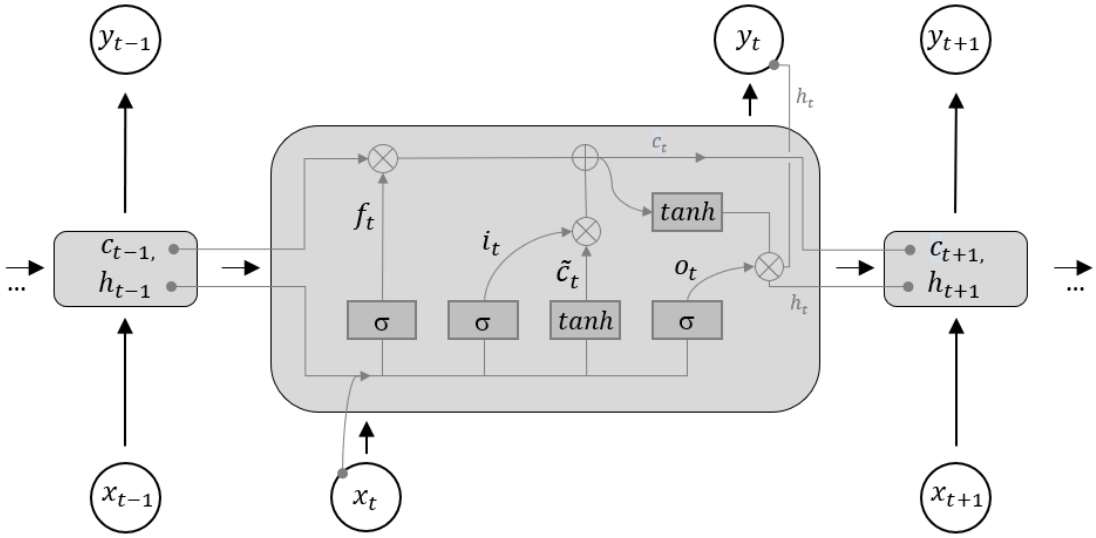


Figure 3. Long short-term memory cell in a recurrent network

The gates in the LSTM cell control the information flow through the network. The input gate defines which information from the input vector and previous output is important and is let through the model by adding it to the memory cell state. Gers et al. (2000) presented the forget gate that learns to reset the LSTM cell state and discard information that seems irrelevant for determining the nature of inputs. While the input gate and the forget gate are used to update the internal state, the output gate is the final limiter for the memory cell state. It defines the relevant information from the memory cell that is further used as output and pushed to the next time step calculations (Brownlee 2018). The purpose of the cell state vector is to store the internal values of the network to be carried for the subsequent time steps.

The input features are presented timestep by timestep to the LSTM network, and the final output will be returned after the last element of the input sequence has been processed. The modifications in the structure of LSTMs compared to standard RNNs enables them to overcome the problem of vanishing and exploding gradients, and to predict time series when time steps with arbitrary size are included. (Fischer & Krauss 2018.)

3.4 Neural network training

Deep learning algorithms can be roughly divided into three major paradigms, that are supervised learning, unsupervised learning and reinforcement learning, of which the latter two are out of the scope of this research. Supervised learning refers to a task of learning a function that maps a set of inputs to outputs using paired input-output examples. The model is provided with a desired output in order to adjust model parameters according to the difference between the target and the actual response given by the model, i.e., according to the error signal. The term supervised learning refers to the presence of an external instructor guiding the learning process. (Goodfellow et al. 2016, Haykin 2009.)

Typically, the performance of neural network models during the training process is measured by some sort of error or fitness function, such as MSE, SSE or MAPE. The choice of the error or fitness function has significant impact on the results, and good results with one performance measure might not repeat with some other measure. (Rozenberg et al. 2012.) In the case of a binary classification problem, the final layer of the network should end with a sigmoid activation function in Equation 1, and thus give a probability value between 0 and 1 as an output. Binary classification problem with a sigmoid output should be combined with a binary cross-entropy loss function, that can be defined for a single target-output pair by the following equation

$$BCE\left(y_t, \hat{P}_t\right) = -\left(y_t \times \log\left(\hat{P}_t\right) + (1 - y_t) \times \log\left(1 - \hat{P}_t\right)\right), \tag{5}$$

where y_t denotes the target label, and $\hat{P}_t$ the conditional probability given as an output by the model at time t. The binary cross-entropy loss function computes the cross-entropy loss between true labels and predicted labels, and thus not only measures correct predictions, but also penalizes for the difference between the predicted probability and the true label. (Chollet 2018.)

The goal of the training process in supervised learning problem is to approximate a true data-generation function by learning the error-minimizing parameter values of the input-output mapping. In order to optimize the parameter values, the common choice is to apply backpropagation through time (BPTT) algorithm, which is a gradient descent procedure. The idea behind the BPTT is to allow the information from the selected loss function to flow backwards through the network in order to calculate the parameter-specific gradients. The computed gradients are further used to update each parameter with

gradient descent, and the operation is repeated until satisfactory level of error is achieved, or until the parameters are converged. The BPTT applies several multiplications and is therefore sensitive to the activation values in the network. In some cases, this might lead to the abovementioned problems of vanishing and exploding gradients. (Goodfellow et al. 2016, Tsay & Chen 2019.)

The training process of neural networks involves training epochs, where the weights and biases are driven iteratively towards the lowest cost in the validation set. After the whole training sample has been fed through the training loop, one epoch of training has been performed. The number of epochs is therefore the number of times that the full training dataset is fed into the network. (Bengio 2012, Goodfellow et al. 2016.) The parameter optimization process is illustrated in Figure 4, where the model performance is visualized by comparing the training loss and validation loss curves. The loss curves tend to start from high values but start decreasing after each epoch, as the model begins to enable a better fit by learning the meaningful relationships in the data. Given enough modelling capacity, the training error is reducing steadily when the number of epochs increase, but the error in the validation set starts rising. In other words, eventually the model tends to start memorizing the training data when the number of epochs increases, which leads to increasing validation error, and thus decreased prediction accuracy. The training process can be haltered after predetermined number of epochs if the validation error continues to rise. The model parameters that minimize the validation error can later be retrieved from the optimum point in the training process.

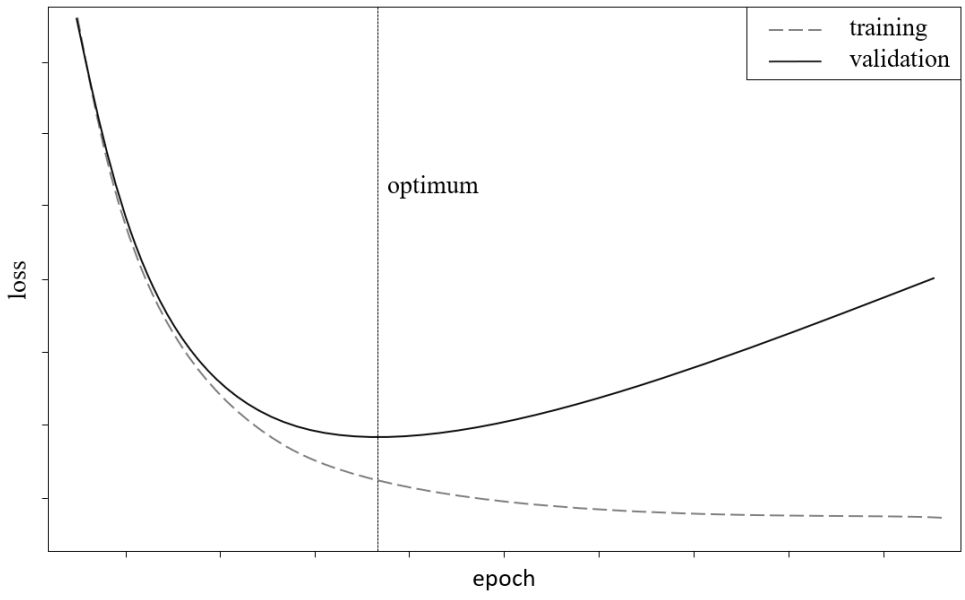


Figure 4. Training and validation loss curves during neural network training

Bias is an error of an estimator that occurs from the expected deviation between the predictions and true values. Variance, in turn, measures the error generated from the deviation between expected predictions and real predictions. Variance can be interpreted as the sensitivity of a model to small changes in the input data. The bias-variance tradeoff is a fundamental concept in machine learning and refers to the conflict in minimizing these sources of error simultaneously. In general, a simple model with low modelling capacity bear to have high bias and low variance, which often results to underfitting. Underfitting refers to a situation where the model is not able to sufficiently learn the relationships between dependent and independent variables within the training dataset. A complex model with high modelling capacity easily starts to overfitting the data, and thus will contain a small bias and high variance. Overfitting refers to a situation when the model begins memorizing, i.e., modelling the random noise in the training data and thus starts to lose its out-of-sample generalization ability. Underfitting and overfitting are one of the biggest challenges faced in practical applications of machine learning, and more particularly in the training process of neural networks. There are no universal rules on how to set up the configuration of a neural network, and thus these issues occur regularly. In order to avoid these issues, for example different regularization methods have been introduced in the literature. (Goodfellow et al. 2016, James et al. 2013, Tsay & Chen 2019.)

3.5 Neural networks in financial market prediction

In the late 1980s the attitude towards neural networks and their potential for stock market prediction was still very pessimistic. The level of technology was still far from present and the computational complexity of neural network models limited their potential. Neural networks had already been successfully applied in other disciplines, but White (1988) was among the first to give an indication of their potential also in the stock market. In his research, he shoots for evidence against the efficient market hypothesis by using a simple neural network to detect nonlinear regularities in the daily return of the IBM stock. He argues that the results based on training data were overly optimistic as even simple neural networks were really prone to overfitting.

Atsalakis and Valavanis (2009) review more than one hundred articles that focused on the utilization of neural networks in financial market prediction tasks. Most of the surveyed papers use reasonably simple input data, such as the opening and closing prices of stocks. In their study, Atsalakis and Valavanis (2009) conclude that neural networks

are suitable methods for stock market prediction tasks as they are often able to capture complex and nonlinear relationships in stock market data. The neural network models in the reviewed studies mostly outperformed the benchmark methods in both predictive and financial performance. They also note that one of the most powerful characteristics of neural networks is that they do not require prior knowledge about the underlying data-generating mechanism. Bahrammirzaee (2010) comes to a similar conclusion in his study by reviewing previous scientific publications on the application of neural networks in the financial sector. He finds that neural networks often yield much more accurate results than traditional statistical models, especially when the research problems are nonlinear in nature. However, Bahrammirzaee (2010) also notes that some studies present contrasting results, where neural networks are outperformed by traditional statistical methods and thus the superiority of neural networks in forecasting tasks cannot be concluded.

Chen et al. (2003) apply a probabilistic neural network to predict the daily directions of index returns of the Taiwan Stock Exchange. Their results show that neural networks are useful tools for predicting directional movements of index returns, as the network predictions are more accurate compared to pure random walk prediction models. In addition, their neural network -based investment strategy achieves higher returns than a simple buy-and-hold strategy. However, the sharp downturn in the Taiwanese financial sector during the testing period partly explains the poor performance of the simple buy-and-hold strategy.

Another study on a smaller market was conducted by Kara et al. (2011), who examine the predictability of daily stock price index movement in the Istanbul Stock Exchange and compare the performance between artificial neural networks and support vector machines. Using a set of ten technical indicators, between 1997 and 2007 their ANN model achieves an average classification accuracy of 75.74%, compared to the 71.52% of the support vector machine. The lowest predictive accuracy is achieved in 2001, when Turkey was suffering from a severe economic crisis and the Istanbul Stock Exchange eventually lost 30% of its value.

Krauss et al. (2017) compare deep neural networks, random forests, gradient-boosted trees, and different configurations of these models on stock movement prediction task of the S&P 500 constituents from 1992 to 2015. In contrast to the studies of Chen et al. (2003) and Kara et al. (2011), models are trained to predict the stocks that outperform the general market, rather than predict pure directional movements of each stock. Furthermore, each of the models are trained with the past three years of lagged returns of all S&P

500 constituents. They find that deep neural networks do not individually outperform traditional methods as they reach daily returns of 0.33 percent prior to transaction cost, compared to the 0.37 percent by gradient-boosted trees and 0.43 percent by random forests. However, their best performing model is an ensemble method that combines all the three individual techniques and receives daily returns of 0.45 percent prior to transaction costs. A long-short portfolio constructed based on the ensemble method predictions generates a Sharpe ratio of 1.81 after transaction costs, compared to 0.35 of the S&P 500 during the sample period. When examining the sources of profitability, Krauss et al. (2017) find that the model primarily utilizes the returns from the past 12-month and one-week periods when selecting stocks for trading. They note that the returns start to decline soon after 2001 and argue that this might be due to the rapid increase in computing power and extensive application of machine learning in the stock market. In addition, the performance of all models starts to cripple even more when entering to the 2010s.

The work of Krauss et al. (2017) is expanded by Fischer and Krauss (2018), who deploy a long short-term memory network to predict daily out-of-sample directional movements for the constituents of the S&P 500 index from 1992 to 2015. Similar to Krauss et al. (2017) their models are trained to predict which stocks outperform the cross-sectional median on the following day. They use the past 240 one-day returns of each stock as input features. The benchmark methods are random forest, standard deep neural net and a simple logistic regression classifier. They find that the LSTM network reaches statistically and economically significant daily returns of 0.46 percent, and a Sharpe ratio of 5.8 prior to transaction cost, outperforming all memory-free benchmark methods. However, their results show similar patterns as the ones of Krauss et al. (2017), since after 2010 the profitability of the LSTM strategy fluctuates around zero after transaction costs, so the high-level performance seems to be arbitraged away. More interestingly, they find that stocks selected for trading by the model share similar characteristics; below-mean momentum, high volatility prior to trading and a tendency for mean-reversion. By performing regression analyses on systematic risk factors, they also find that their LSTM portfolio has generally lower exposure to well known risk factors than the baseline methods, also similar to the results of Krauss et al. (2017).

Long et al (2019) introduce a multi-filters neural network that utilizes convolutional and recurrent neurons and apply it on a daily stock price movement prediction task on the Chinese stock market index CSI 300. They find that advanced deep learning models clearly outperform statistical models, but also more traditional machine learning models,

such as single-structure RNNs or LSTMs in terms of accuracy and profitability. Furthermore, they find that deep learning methods are well capable of identifying trading signals from historical market data. However, the proposed multi-filters network does not provide greatest stability among all models, as the proposed single-structure RNN achieved a significant Sharpe ratio of 6.42.

Some of the academic papers on the topic focus on frequencies of less than one trading day. Nelson et al. (2017) use a LSTM classification model to predict short-term trends of stock prices for the IBovespa index constituents from the BM&F Bovespa stock exchange, with a data span from 2008 until 2015. They utilize stock price candlesticks and a wide set of technical indicators as model inputs and achieve promising results with their LSTM network. Their model outperforms the benchmark methods by having an average of 55.9% directional accuracy. However, they conclude that even though the prediction accuracy was satisfactory, the model variance could be lower, and an even more stable model could thus be constructed. In addition to the predictive accuracy, they compare different methods in terms of their financial performance, e.g., by computing the maximum drawdown of a portfolio, and conclude that the LSTM-based trading strategy involves less risk compared to the baseline methods. The lower risk exposure of their LSTM portfolio is in line with the results of Krauss et al. (2017) and Fischer and Krauss (2018).

Makridakis et al. (2018) evaluate the performance of different machine learning models and statistical methods on univariate time series prediction tasks. Their extensive study includes over a thousand monthly economic and financial time series, and the predictions are made with multiple horizons. They show that that traditional statistical models outperform the machine learning models on all accuracy measures and examined forecasting horizons. They argue that the additional complexity and required computational power of the machine learning models is not convenient when univariate time series are modelled. In addition, they show that LSTM network is not able to outperform a simple MLP structure in terms of forecasting accuracy, even though LSTMs are practically designed for sequence prediction tasks. The results of Makridakis et al. (2018) indicate that univariate time series do not include any complex patterns that only more sophisticated methods could be able to capture. However, the results are inconsistent with the ones of Fischer and Krauss (2018), who show that LSTM network outperforms all benchmark methods in a univariate time series prediction task.

There is no clear consensus on the literature as to whether machine learning models are better than traditional models in predicting stock market developments. The black-

box properties and lack of universal optimization rules might partly explain the conflicting findings, since simpler models might provide greater confidence, at least if the predictions are used for investment purposes. Another possible explanation is that stock market information that has been collected from multiple sources and has been processed by some complex methods, may be comparable to insider information to some extent. The information is valuable to its holder as long as it stays private or is known only by a very small group of people. When the information leaks or becomes public, its usability in the pursuit of excess returns begin to vanish as it slowly starts to reflect in stock prices. Therefore, that might partly explain why the advancements on the field might not fully materialize in the literature.

Different data preprocessing methods, model architectures, model hyperparameters, parameter initialization and the stochasticity of the training process all have a great influence on the performance of different models and therefore make clear interpretation of the results difficult. Furthermore, the results even on the same dataset might significantly differ between studies using similar models. However, advanced machine learning models have proved to have several desirable attributes and thus highlight the need to fill the research gap in machine learning applications on stock market prediction tasks on different markets.

4 RESEARCH METHODS

4.1 Data

Fifteen stocks from the Finnish stock market are selected for the research: Nokia, Sampo A, Fortum, UPM-Kymmene, TietoEVRY, Huhtamäki, Kemira, Kesko B, Neles, Wärtsilä, Outokumpu A, Atria A, Bittium, Finnair, and Uponor. All the selected stocks provide adequate liquidity and full data availability for the selected time period, also covering a wide range of industries. The selected stock universe is not survivorship bias free, since the selected stocks are also current ones, and companies that went bankrupt or were delisted during the research window are ignored. Providing a survivorship bias free stock universe would unnecessarily increase the complexity of the research, without brining considerable additional value considering the objectives of the research.

The data for the empirical analysis are gathered from the Refinitiv Eikon Datastream. The full research sample covers the period from the 11th of January 1999 to the 8th of June 2020, as the selected data span provides full data availability and sufficiently large sample size. In this research, the predictions, portfolio construction and empirical analysis are conducted with a weekly observation frequency in order to avoid excessive noise and to capture longer-term dependencies in the data. The decision to use weekly observations instead of daily observations also significantly reduces computational burden. All of the obtained time series contain a total of 1118 weekly observations, where the day of the week of each observation is Monday.

For the historical return pattern construction, weekly total return indices are used since they are the most adequate metric for return computations, taking dividends and all relevant corporate actions into account (Fischer & Krauss 2018). Total return index shows a theoretical growth in value of a stock holding over a period, assuming that dividends are re-invested. The price of an individual stock s at time t is acquired from the corresponding total return index and is represented as P_t^s. A simple return R for a stock over m periods can thus be expressed by the following equation

$$R_t^s[m] = \frac{P_t^s - P_{t-m}^s}{P_{t-m}^s}.$$

The summary statistics of the stock returns for the full sample period are presented in Table 1. All the stock returns are leptokurtic and on average have high excess kurtosis, which is a typical characteristic in the stock market since the returns tend to center toward the mean and have fat-tailed distributions.

Table 1. Summary statistics of the stock returns 1999–2020

The table reports weekly mean, standard deviation, skewness and excess kurtosis of the returns for selected stocks from the 11th of January 1999 to the 8th of June 2020. Weekly mean returns and standard deviations are denoted in percent.

Stock	RIC	Mean	St. dev.	Skewness	Kurtosis
Nokia	NOKIA	0.150	6.406	0.096	3.371
Sampo A	SAMPO	0.347	3.959	-0.285	4.808
Fortum	FORTUM	0.310	3.679	-0.378	3.213
UPM-Kymmene	UPM	0.281	4.715	0.090	1.692
TietoEVRY	TIETO	0.198	5.737	0.769	8.145
Huhtamäki	HUH1V	0.283	4.202	0.248	4.271
Kemira	KEMIRA	0.301	4.549	0.118	7.849
Kesko B	KESKOB	0.332	3.835	-0.048	3.640
Neles	NELES	0.357	5.545	-0.095	2.612
Wärtsilä	WRT1V	0.387	4.988	0.157	3.838
Outokumpu A	OUT1V	0.138	7.022	0.618	3.585
Atria A	ATRAV	0.220	4.015	0.475	6.493
Bittium	BITTI	0.285	7.294	0.773	6.614
Finnair	FIA1S	0.139	4.585	0.477	4.078
Uponor	UPONOR	0.268	4.947	-0.177	2.718

In addition to the historical returns, additional predictive variables, hereinafter referred as features, are considered in order to analyze the importance of supplemental variables to the predictive performance. The selection of the additional features is based on the theory of stock return predictability, as well as on previous studies on the topic. Data availability and computational complexity were also taken into account in the feature selection process. To evaluate the incremental value added by the additional features, two sets of input patterns are used with each proposed model. In the univariate approach, hereinafter denoted as *(R)*, only the historical return patterns of each individual stock are used as inputs. In the multivariate approach, hereinafter denoted as *(All)*, the stock return patterns are complemented with a set of four stock-specific, and four common features. Aggregate variables, such as moving averages or momentum factors, are not considered since LSTM networks should be able to pick up these patterns from the data (Fischer & Krauss 2018).

The first stock-specific feature is turnover by volume (*VO*), that displays the number of shares traded for a stock over a particular week. The volume figures are expressed in thousands, adjusted for capital events and the default volumes are taken from the primary exchange of a country. Volume data is frequently employed together with past prices in order to identify market trends and is commonly used as a predictive feature (see, e.g., Blume et al. 1994, Neely et al. 2014). Second stock-specific feature is market value (*MV*), that is the stock price multiplied by the number of ordinary shares in issue. The market value figures are also adjusted for capital events and the amount in issue is updated whenever new tranches of stock are issued. However, for companies with multiple classes of equity capital, the market value is expressed only according to the individual issue. Price-to-book value (*PB*) is the third stock-specific feature in the multivariate approach and is calculated as the share price divided by the book value per share. The final stock-specific feature is the dividend yield (*DY*), that expresses the dividend per share as a percentage of the share price. The calculation of the underlying dividend is based on an anticipated annual dividend and excludes special or once-off dividends. Dividend yield is calculated on gross dividends, also including tax credits. According to Er and Vuran (2013), market value, dividend yield and price-to-book ratio have been found to be variables significantly affecting the stock returns. Brennan et al. (1998) also show that stock returns are strongly related to volume, firm size and book-to-market ratios.

The set of four common features include the VSTOXX index (*V2TX*), three-month Euribor rate (*EB3M*), the euro versus U.S. dollar reference exchange rate (*EUSD*), and the weekly total return of the OMX Helsinki stock index (*OMXH*). The VSTOXX index is an EU volatility index designed to reflect market expectations of near- to long-term volatility. It is based on EURO STOXX 50 real time options prices and measures the square root of the implied variance across all options at certain point in time. The Euribor interest rates are the most important European interbank interest rates, and the three-month Euribor is for loans that have a maturity of three months. The euro versus U.S. dollar reference exchange rate used in this research is published by the European Central Bank. Although Kara et al. (2011) only use technical indicators as inputs in their predictive models, they suggest considering exchange rates and interest rates as features. Er and Vuran (2013), as well as Zhong and Enke (2017), show that exchange and interest rates are among the most significant variables affecting stock returns.

4.2 Training, validation and prediction sets

The raw data is split into study periods, composed of training samples of 390 weeks for in-sample training, and prediction samples of 52 weeks for out-of-sample predictions. For the neural network models, each training sample will be further divided into training and validation sets in order to provide an unbiased evaluation of the model fit while tuning model parameters and hyperparameters. The prediction set is used to evaluate the performance of the models with unseen data, and thus it yields an estimate of their generalization capacity. Typically, the training-validation ratio is recommended to be approximately 80-20 when training neural networks (see, e.g., Brownlee 2018, Fischer & Krauss 2018, Murphy 2012). Therefore, six years (312 weeks) of data are used as a training set, whereas the preceding one and a half years (78 weeks) of data are used as a validation set. The last fourteen years (728 weeks) of data are used for out-of-sample predictions with fourteen non-overlapping, 52 weeks long prediction periods reaching from the 3rd of July 2006 until the 8th of June 2020. In other words, a rolling estimation window is used, and the whole system will always be rolled forward by the length of a prediction set. The model parameters, e.g., the connection weights, are estimated using the training and validation sets, and next, the models predict weekly out-of-sample directional movements for the selected stocks in the prediction period. The networks are regenerated on a yearly basis, and each year the training sample only includes the previous 390 weeks. Rolling estimation window excludes data that is far in the past and thus allows the networks to be trained according to changing conditions. Re-training the model in an annual frequency also reduces computational costs. The lookback period, i.e., the length of a single feature set, is set to be 60 weeks in order to capture dependencies almost up to 14 months. In other words, the models will utilize information from the preceding 60 weeks to produce an out-of-sample prediction for the following week.

4.3 Feature and target generation

The feature and the target generation processes follow the approach in Fischer and Krauss (2018). For all the proposed models, simple one-week ($m = 1$) returns $R_t^{1,s}$ for each week t and each stock s are calculated and stored to feature matrix $\boldsymbol{r}$ of dimension $n_s \times T$, where n_s denotes the total number of stocks and T is the total number of weeks in the dataset. Next, these simple returns are standardized by subtracting the training set mean

return μ_{train}^{m}, and further dividing the result by the standard deviation σ_{train}^{m} obtained from the same training set. The mean and standard deviation is calculated separately for each training set in rolling fashion and are further used to standardize the returns in the corresponding validation and prediction sets in order to avoid look-ahead bias. Therefore, a single feature is the standardized one-week return $\tilde{R}$ expressed as

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

The input features need to be presented as sequences in the training of the neural network models (Goodfellow et al. 2016). Therefore, the single standardized one-week returns are stacked into overlapping input sequences with a length corresponding to the selected lookback period of 60 weeks. As a result, each study period includes a total of $(252 + 78 + 52) = 382$ consecutive sequences of these standardized one-week returns in the form of $\{\tilde{R}_{t-59}^{1,s}, \tilde{R}_{t-58}^{1,s}, \dots, \tilde{R}_t^{1,s}\}$ for each stock s and each $t \geq 60$ within the study period. These return sequences are used as the sole input in the univariate approach, and as a part of the input set in the multivariate approach. The same sequence construction procedure is also later applied to all the additional features.

Neural networks do not require stationary data, but in order to provide a more stable learning process, detrending can still be applied to some time-series containing a strong trend. The detrending can be carried out in e.g. by taking first difference of the series. (Rozenberg et al. 2012.) In this research, first differencing is applied to all additional features used in the multivariate approach. Since neural networks are able to deal with nonstationary data, stationarity tests are not implemented in this study. None of the time series include missing values and thus interpolation is not required.

Data for sequence prediction and classification problems is typically standardized in the way that was presented with the simple returns, or alternately scaled within the range of 0 and 1 (Bengio 2012). Rozenberg et al. (2012) note that there is some controversy about the necessity and impact of data normalization, as networks should be able to capture all the underlying patterns without any data preprocessing. However, data scaling is typically conducted in order to prevent large inputs from slowing down the training process of a deep learning network. A popular method for normalization is to use min-max scaling, where the data is scaled using the smallest and highest values in the dataset. This method is vulnerable to outliers as the future data might cross the scaling boundaries if

the scaling parameters are not adjusted. (Brownlee 2018.) In order to speed up the learning process, in this research all the additional features all scaled within the range of 0 and 1 by using the MinMaxScaler function from the Scikit-learn package (Pedregosa et al. 2011). In order to avoid look-ahead bias, the scaler is fit within each training set, and the minimum value x_{tmin} and maximum value x_{tmax} of the training sets are further applied to scale the values x in the corresponding validation and prediction sets. Therefore, the common min-max scaling equation can be expressed as

$$x_{scaled} = \frac{x - x_{tmin}}{x_{tmax} - x_{tmin}}.$$

The response or target labels are binary variables displaying whether an individual stock outperforms the cross-sectional median return at t. If the one-week return $R_t^{1,s}$ of a stock s is less than the cross-sectional median return in the same week, the stock is classified as underperforming and denoted by class 0. Furthermore, if the one-week return of a stock is greater or equal to the cross-sectional median, the stock is classified as outperforming and denoted by class 1. The target labels can thus be written as

$$y_t^s = \begin{cases} 1, & if R_t^{1,s} \geq m_t \\ 0, & if R_t^{1,s} < m_t \end{cases},$$

where m_t is the cross-sectional median of the returns at time t. The prediction phase is approached as a binary classification problem where the proposed models first predict the conditional probability $\hat{P}_{t+1|t}^S$ for a stock s to outperform the median return at $t+1$ by utilizing information only until t. The conditional probabilities are further converted to binary class outputs $\hat{y}_t^s$ determined by the equation

$$\hat{y}_t^s = \begin{cases} 1, & if\ \hat{P}_t^s > 0.5 \\ 0, & if\ \hat{P}_t^s \leq 0.5 \end{cases}.$$

4.4 LSTM networks

In order to capture stock-specific patterns in the data, all models are trained separately for each stock. This also lowers the number of input features per model and thus reduces the computational burden. Since there are three model classes and two sets of input patterns, the number of individual models investigated is $3 \times 2 \times 15 = 90$. However, the model configurations are set equal within model classes utilizing same input data.

Hyperparameters are configurations that cannot be estimated from the data and are thus manually specified by the user (Brownlee 2018). In this research, the model architectures and hyperparameters of the LSTM networks are selected by conducting multiple trial and error -based experiments. More specifically, these experiments are implemented on the first validation set by examining the predictive performance with several different model configurations while controlling a single hyperparameter at a time. Since the first validation set is not used as part of the out-of-sample prediction periods, all predictions are made on unseen data and there is no look-ahead bias. Summary of the selected hyperparameters is presented in Table 2. The process of tuning hyperparameters is time-consuming due to the number of tunable hyperparameters and the computational resources required. In addition, the results of the experimental process might be misleading and lead to uncertainty about the true generalization capacity, as the selected configuration is presumably non-optimal (Bengio 2012).

Table 2. Hyperparameters of LSTM networks

This table reports the selected configurations of the proposed LSTM networks. The univariate LSTM network utilizing only the past return pattern is denoted with *(R)*, whereas the multivariate network utilizing the full input feature set is denoted with *(All)*.

	LSTM (R)	LSTM (All)
Number of layers	1	2
Hidden units (layer 1)	3	9
Hidden units (layer 2)	-	9
Dropout ratio	0.30	0.30
Batch size	52	52
Epochs	1000	1000
Activation function	sigmoid	sigmoid
Optimizer	RMSprop	RMSprop

According to Rozenberg et al. (2012), the selection of the number of hidden layers is important, but several studies on the subject show that the performance of neural networks is not very sensitive to this number. Therefore, in most cases using a single hidden layer should provide adequate results with less computational capacity required. In contrast, in their large survey about stock market forecasting techniques, Atsalakis and Valavanis (2009) found out that in studies that utilize neural networks, on average two hidden layers are used in the models. The authors note that additional hidden layers would increase the modelling capacity but also the computational complexity, while one hidden layer might not uncover all relevant patterns in the dataset. According to Goodfellow et al. (2016), a more complex problem demands a more complex model, and chaining multiple hidden layers together can have a significant influence on the results. Several studies also support the use of larger architectures with proper regularization (Bengio 2012, Graves et al. 2013, Goodfellow et al. 2016). Based on these findings and the experiments on the first validation set, a single hidden layer is deployed with the univariate model, while two hidden layers are deployed with the multivariate model.

The selection of the number of hidden units in the hidden layers follow the same idea, where shallow architectures are considered less prone to overfitting and more stable in terms of predictive performance. The experiments show that in the univariate approach, shallow architectures perform better compared to more complex structures. Overall, three hidden units in the single hidden layer provides the most stable performance, and thus it is applied with the univariate model. Due to the additional complexity in the multivariate approach, also the initial number of hidden units is set to be higher compared to the univariate approach. As suggested by Bengio (2012), the number of hidden units is set to be equal in both hidden layers. The most stable performance was achieved with nine units within both hidden layers, and thus those are applied with the multivariate approach. Number of weights and bias terms to be trained in a single LSTM layer is calculated as $4(h(i+1)+h^2)$, where h denotes the number of hidden units in the hidden layer and i is the total number of input features fed into the network. Therefore, the number of trainable parameters in *LSTM (R)* network is $4(3(1+1)+3^2) = 60$, and $2 \times 4(9(9+1)+9^2) = 1368$ in the multivariate approach. The size and type of the output layer in the network is determined depending on the nature of the problem. In contrast to the study of Fischer and Krauss (2018), who use a softmax activation function with two output nodes, in this research one output node with a sigmoid activation function in Equation 1 is deployed to generate the conditional outperforming probabilities for each stock.

Dropout is a computationally cheap regularization method for reducing overfitting, first introduced by Srivastava et al. (2014), and as suggested by Goodfellow et al. (2016), it is used within the recurrent layers of LSTM networks. The objective of dropout regularization is to improve the generalization capacity, and thus the out-of-sample performance of the network by dropping out a defined portion of the layer outputs during training process. The best performance in both the univariate and the multivariate approach was achieved with 0.30 dropout ratio. On average, higher or lower dropout ratios resulted to a decline in performance in both approaches.

In addition to dropout regularization, early stopping method is used in order to reduce the risk of overfitting. According to Goodfellow et al. (2016), early stopping is one of the most common forms of regularization and should be adopted almost without exception when training neural networks. The early stopping method allows the model to halt the training process after the validation loss has stopped decreasing from its lowest point during a selected number of epochs. The early stopping threshold in this research is set to abort the training process when there is no improvement in model performance during the last thirty epochs, as this should provide enough buffer in the case that validation error increases only temporarily and soon starts decreasing again. After the training process is terminated, the weights and parameters are retrieved from the point with the lowest validation error using the ModelCheckpoint callback (Chollet 2015).

The batch size is the number of examples fed through the training loop each time before adjusting the weights. A batch size of one means feeding the data through the network one example at a time. According to Bengio (2012) the batch size is mostly used to control the speed of learning, not so much the performance. However, in the hyperparameter experiments, smaller batch sizes resulted in an increasing training loss and bad overall performance. The most stable performance was achieved with a batch size of 52, which is applied for all models.

Similar to Fischer and Krauss (2018), RMSprop optimization algorithm is utilized in the training of the LSTM network. In contrast to a conventional gradient descent that exploits a fixed learning rate, RMSprop utilizes mini-batches and uses a decay rate along with a global learning rate. Due to this type of adaptive learning rate, no other configurations to the learning rate is required (Hinton 2012). RMSprop is considered an adequate optimization algorithm when training deep neural networks, regardless of the problem in question (see, e.g., Chollet 2018, Goodfellow et al. 2016).

4.5 Benchmark models

4.5.1 Conventional RNN

A conventional recurrent neural network is applied to show the relative advantage of the introduction of long short-term memory cells to the network structure. In principle, the LSTM network should be able to capture the long-term dependencies better than the simple RNN network, and therefore the superior performance of the LSTM model would indicate that these types of relationships exist in the data. Instead of the LSTM cells in the hidden layers of the network, hyperbolic tangent activation function in Equation 2 is deployed using the SimpleRNN layer in Keras (Chollet 2015). The *RNN (R)* and *RNN (All)* architectures and hyperparameters are set to be similar with the corresponding LSTM networks. Therefore, any additional optimization to the RNN configurations is not deployed in order to reduce computational costs and to provide a more equal comparison between the models. Number of trainable weight and biases in a single RNN layer is calculated as $h(i+1)+h^2$, making it one fourth of those to the corresponding LSTM networks.

4.5.2 Logistic regression

Logistic regression is a commonly used statistical model for classification problems due to its interpretability and simplicity. It can also be classified as a simple neural network with one output neuron, as in the simplest terms, adding hidden units to a logistic regression model converts it into a multilayer perceptron (Goodfellow et al. 2016). Logistic regression is highly suitable for modeling binary dependent variables, and therefore acts as a good baseline method in this research. However, the performance of this standard classification model is more dependent on the data representation, i.e., the features of the model and thus using the same lookback period as with the LSTM and RNN models is not reasonable.

As suggested by Varian (2014), the lookback period is selected by running a 10-fold cross-validation procedure for both logistic regression models *LOG(R)* and *LOG(All)*, with lag lengths from 1 to 60. Using the last seven or eight observations of each variable as inputs provided the most stable results across all stocks with the univariate models, whereas lag length of eight provided the best results with the multivariate model. Therefore, the selected lookback period of the logistic regression is set to be eight, thus

including observations approximately from the past two months. Unlike in the paper of Niaki and Hoseinzade (2013), who include the validation set to fit the logistic regression model, the validation set is excluded in order to ensure an equal starting point for all models. The default options, L2 regularization and L-BFGS solver, are used and maximum number of iterations is restricted to 1000. The logistic regression model is built using the Scikit-learn package by Pedregosa et al. (2011).

4.6 Performance evaluation

Several different measures are used to compare the predictive performance of the proposed models. These include classification accuracy, precision, recall and F1 score. Classification accuracy measures the percentage of correct classifications out of all classifications. Precision tells what fraction of the positive predictions (class = 1) were true positives, while recall is the fraction of true positives that were correctly classified. The F1 score can be interpreted as the harmonic mean between precision and recall and is calculated as follows

$$F1 = 2 \times \frac{precision \times recall}{precision + recall}.$$

As suggested by Gu et al. (2018) as well as by Fischer and Krauss (2018), the Diebold-Mariano test is deployed in order to evaluate if the out-of-sample performance of the LSTM models are truly different compared to the benchmark models. The null hypothesis in the Diebold-Mariano test assumes that the prediction accuracies of two models are equal, and the test statistic can be expressed as

$$DM_{12} = \frac{\bar{d}}{\sqrt{[\gamma_0 + 2\sum_{k=1}^{h-1}\gamma_k]/T}},$$

where γ_k are the autocovariances and

$$\bar{d}_{12} = \frac{1}{TK}\sum d_{12},$$

where T is the number of time steps and K is the number of stocks, and further

$$d_{12} = |\hat{e}_1| - |\hat{e}_2|,$$

where $\hat{e}_i$ are the residuals of the forecasts. In addition to the Diebold-Mariano test, a statistical estimate is provided to assess whether the models achieved the total accuracies randomly. The number of correct predictions in a pure chance model could be presented with a binomial distribution

$$\boldsymbol{X} \sim \mathrm{Bin}(n = TK, p = 0.5, q = 0.5),$$

where T is the number of time steps and K is the number of stocks. For a such large number of observations, a normal distribution can be used as an approximation to the binomial distribution

$$\boldsymbol{X} \sim \mathrm{N}\big(\mu = np, \sigma = \sqrt{npq}\big).$$

Consequently, the probability of achieving the realized accuracy if the true model accuracy is 50 percent, can be computed.

Unlike most traditional prediction models, neural network algorithms are stochastic and thus make different predictions each time the constructed model is trained on the same dataset (Brownlee 2018). This can be taken into account by running the networks several times and taking the averages of the results. However, in this research the exact model predictions are needed for the portfolio construction phase and thus using the average of results is not possible. Therefore, the model predictions and other statistics for the performance evaluation are taken from the first run of each model after the models have been constructed. Furthermore, in order to get a better overview on the generalization capacity of the neural network models, the robustness is later evaluated by comparing the predictive accuracies across several runs.

4.7 Portfolio construction

The proposed predictive models produce the conditional probabilities for each stock s to outperform the cross-sectional median return at $t + 1$ by utilizing information only until step t. These probabilities are further used as the base for the portfolio construction phase, where a simple prediction-based long trading strategy is implemented, and each model is compared in terms of profitability of the constructed portfolios. Due to the implementational issues regarding liquidity, lending availability and the transaction costs they produce, short trading strategies are not considered in this study.

All stocks are ranked in descending order of these probabilities $\hat{P}_t^s$ for each week, and these cross-sectional rankings are used to update the positions in a long portfolio. The portfolio construction is based on a simple heuristic approach where k number of stocks are bought at a closing price at the end of each week and portfolio is adjusted accordingly. The k number of stocks selected are the ones with the highest predicted outperforming probabilities. The portfolio value V at each week t can be expressed by the following equation

$$V_t = V_{t-1} \times \left[1 + \sum_{k=0}^{K} (w_{t-1,k} \times r_{t,k})\right],$$

where $w_{t-1,k}$ denotes the weight of stock k in the portfolio at the previous week and $r_{t,k}$ is the return of a stock from $t - 1$ to t. Instead of using equal weights within the portfolio, bet sizing is implemented to consider the differences in the certainty of the predictions and thus possibly increasing profitability of the strategies (López de Prado 2018). The bet sizing is implemented as follows. First, the conditional probabilities are standardized separately for each week t and next, the k stocks with the highest probabilities are selected. Finally, the weights of the selected stocks in the portfolio are calculated by re-scaling the standardized probabilities so that the weights sum up to one.

Most academic papers on the topic trade day-to-day or week-to-week on adjusted closing prices and only accounting for the direct transaction costs, usually assumed to be around five basis points (Fischer & Krauss 2018, Krauss et al. 2017). The level of direct transaction costs per trade in this research is thus also set to be five basis points, and no other transaction costs are considered in the calculations. The portfolio value after direct transaction costs can be expressed as

$$V_t = V_{t-1} \times \left[1 + \sum_{k=0}^{K} (w_{t-1,k} \times (r_{t,k} - x \times p_{t.k}))\right],$$

where x denotes the amount of direct cost of a single transaction and $p_{t,k}$ is a variable that indicates the number of transactions applied for stock k at week t. The number of transactions equals two in a situation where a stock is held in the portfolio only for a single week. Cumulative compounded returns of the portfolios are calculated at the end of the trading process, and the profitability of the proposed models are evaluated with and without transaction costs. The portfolio values are indexed to 100 with the 3rd of July 2006 as the base date. Other performance metrics used in the comparison of financial performance include, i.e., annualized mean return, annualized standard deviation, skewness, excess kurtosis, maximum drawdown and annualized Sharpe ratio prior to transaction costs. The prediction-based trading strategies are further compared in terms of their financial performance with a simple buy-and-hold strategy, where all fifteen stocks are bought in the beginning of the trading window and held until the final date.

5 RESULTS

5.1 Predictive performance

Throughout this section, the univariate and multivariate models are separated with notations *(R)* and *(All)*, respectively. The univariate models include only the historical return patterns as features, whereas the multivariate models also include the *VO*, *MV*, *PB*, *DY*, *OMXH*, *V2TX*, *EB3M* and *EUSD* features. Moreover, the out-of-sample predictions cover the period from the 3rd of July 2006 until the 8th of June 2020. A common choice in the literature is to use the average of the three-month Euribor rate during the sample period as the risk-free rate. However, due to current market conditions and monetary policy decisions, the average three-month Euribor rate is negative during the prediction sample and therefore, the risk-free rate in this research is set to zero and no further risk-adjustments are made in the calculations.

The predictive performance of the proposed models during the full prediction sample is provided in Table 3. The total classification accuracy, binary cross-entropy loss (BCE), precision, recall and F1 score are reported separately for each model. The binary cross-entropy loss is computed as in Equation 5, but instead of a single target-output pair, the loss is calculated for the full sample. Classification accuracy is the rate of correct classifications, whereas precision is the rate of correct positive predictions out of all positive predictions. Recall measures the robustness of the model and tells the fraction of correctly classified true positives. Furthermore, F1 score can be interpreted as the harmonic mean between precision and recall. Finally, the bottom row of the table displays the probabilities that the predictive accuracies would have been achieved by random guess.

Table 3. Predictive performance of the proposed models 2006–2020

The table reports the total classification accuracy, binary cross-entropy loss (BCE), precision, recall and F1 score of each model during the full prediction sample from the 3rd of July 2006 until the 8th of June 2020. Bottom row displays the probabilities that models in the column label achieved the accuracy by chance.

	LSTM (R)	RNN (R)	LOG (R)	LSTM (All)	RNN (All)	LOG (All)
Accuracy	0.528	0.517	0.514	0.530	0.521	0.517
BCE	0.694	0.703	0.714	0.700	0.712	0.738
Precision	0.542	0.536	0.536	0.546	0.541	0.540
Recall	0.738	0.691	0.660	0.708	0.666	0.631
F1 score	0.625	0.604	0.591	0.617	0.597	0.582
Prob.	(0.000)	(0.000)	(0.002)	(0.000)	(0.000)	(0.000)

Probability that the realized accuracy is achieved if the true accuracy of the model is 50 percent is practically zero for all models, except for the *LOG (R)* model, where the probability is 0.2 percent. In other words, all the proposed models delivered accuracies that are statistically significantly different from a random guess. The *LSTM (All)* delivers the highest classification accuracy of 53.0%, with the *LSTM (R)* coming in second with a 52.8% accuracy. In fact, the LSTMs outperform their benchmarks in every single performance measure. Each multivariate model outperforms the corresponding univariate model in terms of classification accuracy. In addition, the logistic regression models are outperformed by the RNNs, that in turn, are outperformed by the LSTMs. Therefore, the results imply that the additional complexity of the models also provide additional value in terms of predictive performance. These findings are in line with Fischer and Krauss (2018) and Kara et al. (2011) who also show that additional complexity comes with a higher classification accuracy. In contrast, Krauss et al. (2017) find deep neural nets delivering worse classification accuracy compared to individual benchmark methods or their proposed ensemble model.

Despite having a lower classification accuracy, the *LSTM (R)* outperforms the *LSTM (All)* in terms of binary cross-entropy loss, recall and F1 score. Binary cross-entropy loss is the only measure that should be minimized, as it penalizes for the difference between the predicted probabilities and true outcomes. Thus, the lower BCE indicates that the certainty of the predictions given by the *LSTM (R)* are more on point compared to other methods, including the multivariate LSTM. High recall rate and F1 score also support the notion that the *LSTM (R)* certainly has some desirable properties, as it is able to correctly classify a substantial part of the true positives using only historical returns as inputs.

Statistical test proposed by Diebold and Mariano (1995) evaluates the significance of differences in the out-of-sample predictions among the models, and the p-values of the test are reported in Table 4. The test makes pairwise comparisons of the true model accuracies, and p-values lower than 0.05 mean that the forecasts of a method in the row label are superior to the forecasts of the method in the column label at 95% confidence level. The results from the Diebold-Mariano test are consistent with the findings of Fischer and Krauss (2018) as both LSTMs provide statistically significant improvement over the benchmark methods at least with a 95% confidence level, with the exception that *LSTM (R)* outperforms *RNN (All)* only at a 90% confidence level. In addition, both RNNs provide improvements over the LOG models, but the differences are statistically significant only between the *RNN (All)* and the *LOG (R)* model with a p-value of 0.062.

Table 4. Diebold-Mariano pairwise comparisons of model accuracies

The table reports the p-values of the Diebold-Mariano test with null hypothesis that the predictions of two models are equal. The alternative hypothesis states that the predictions of a method in the row label are superior to the predictions of the method in the column label. The predictions are made on the period from 3rd of July 2006 until 8th of June 2020.

	LSTM (R)	RNN (R)	LOG (R)	LSTM (All)	RNN (All)	LOG (All)
LSTM (R)	-	0.010	0.002	0.688	0.077	0.018
RNN (R)	0.990	-	0.282	0.995	0.788	0.506
LOG (R)	0.998	0.718	-	0.999	0.938	0.757
LSTM (All)	0.312	0.005	0.001	-	0.041	0.008
RNN (All)	0.923	0.212	0.062	0.959	-	0.168
LOG (All)	0.982	0.494	0.243	0.992	0.832	-

The introduction of LSTM cells in the network structure produces greater improvement in performance against the benchmark model classes among the univariate models, than with the models utilizing the full set of input features. In other words, the properties of the LSTM networks are relatively more valuable when only a single input feature is utilized. Furthermore, none of the multivariate models delivers statistically significant improvements over the univariate models within the same model class.

The results from the Diebold-Mariano test indicate that the LSTMs are able to capture complex relationships in the data that cannot be extracted by any of the benchmark methods. Furthermore, the relative importance of the introduced additional features is not that significant, even though they improve the results marginally. However, this does not rule out the relevancy of the additional features, as a different sample size or different model configurations could turn the results in favor of the multivariate models. In addition, even though the improvement over the corresponding univariate models is only marginal, all multivariate neural network models deliver major improvements over both logistic regression models. It is also to be noted that even though the difference between the classification accuracies of univariate and multivariate models is not statistically significant, it might be significant in an economic sense when the predictive performance is translated into a prediction-based investment strategy.

In addition to addressing overall accuracies, stock-specific classification accuracies of the proposed models for the full prediction sample are reported in Table 5. In addition, average accuracies on each stock across all models are presented. The accuracies fluctuate considerably between stocks, and regarding some of the stocks, the accuracies are on a notably similar level between different models.

Table 5. Stock-specific classification accuracies 2006–2020

This table reports the stock-specific classification accuracies of the proposed models during the full prediction sample from the 3rd of July 2006 until the 8th of June 2020. The final row provides the total classification accuracy across all stocks for a model in the column label. The final column reports the average accuracy across all models on a stock in the row label.

Stock	LSTM(R)	RNN(R)	LOG(R)	LSTM(All)	RNN(All)	LOG(All)	Avg.
Nokia	0.507	0.529	0.492	0.507	0.538	0.519	0.515
Sampo A	0.596	0.565	0.567	0.587	0.574	0.555	0.574
Fortum	0.549	0.518	0.523	0.548	0.508	0.527	0.529
UPM-Kymmene	0.532	0.493	0.525	0.545	0.516	0.508	0.520
TietoEVRY	0.525	0.518	0.515	0.525	0.527	0.516	0.521
Huhtamäki	0.504	0.484	0.508	0.523	0.522	0.521	0.510
Kemira	0.541	0.563	0.563	0.559	0.556	0.560	0.557
Kesko B	0.547	0.554	0.541	0.522	0.527	0.503	0.532
Neles	0.536	0.526	0.515	0.540	0.504	0.519	0.523
Wärtsilä	0.560	0.549	0.496	0.552	0.534	0.527	0.537
Outokumpu A	0.512	0.508	0.478	0.508	0.496	0.484	0.498
Atria A	0.493	0.474	0.486	0.492	0.510	0.518	0.495
Bittium	0.510	0.503	0.495	0.514	0.492	0.497	0.502
Finnair	0.486	0.482	0.475	0.482	0.500	0.479	0.484
Uponor	0.525	0.485	0.525	0.554	0.508	0.516	0.519
Total	0.528	0.517	0.514	0.530	0.521	0.517	0.521

The classification performance of all models is relatively poor regarding Outokumpu A, Atria A and Finnair stocks, with the average accuracies falling below the 50 percent mark. More interestingly, regarding Finnair only one model, *RNN (All)*, achieved 50% accuracy, and that just barely. Since all models are trained for each stock separately, these findings indicate that the models had difficulties learning any predictable patterns regarding these abovementioned stocks. Each of these companies operate within different industries, and thus no clear linkage to the weak performance can be drawn in that sense. In contrast, Sampo A and Kemira are stocks with the highest stock-specific accuracies, with average accuracies of 57.4% and 55.7%, respectively. The high-level performance regarding these stocks is constant across all models, including all univariate models, indicating that some powerful predictable patterns exists in the past returns.

Huhtamäki is the only stock where the introduction of additional predictive features notably improved the performance of all model classes. Otherwise the additional features improved the performance in around half of the cases, so no significant patterns can be

observed in that regard. As the results from the Diebold-Mariano test suggest, the additional features do not provide statistically significant improvements on the predictive performance. However, the importance seems to be stock-specific as the additional features have more effect on the predictability of some stocks than on others.

LSTM (R) and *LSTM (All)* models are the two top performing classifiers regarding more than half of the stocks, more specifically regarding Sampo A, Fortum, UPM-Kymmene, Neles, Wärtsilä, Outokumpu A, Bittium and Uponor. In contrast, regarding Nokia, the LSTMs only beat the *LOG (R)* model while being outperformed by the other benchmark methods. Overall, the LSTMs deliver the most stable and promising predictive performance out of all models.

Predictive performance of the proposed models is shown to vary between different stocks, and in order to explore also the possible time-variation, 26-week rolling classification accuracies are presented in Figure 5. Time series of accuracies alone are too volatile to be analyzed and thus rolling accuracies are used. Performance of the univariate models is displayed in one figure, and performance of the multivariate models in another. The sample covers the period from the 1st of January 2007 until the 8th of June 2020.

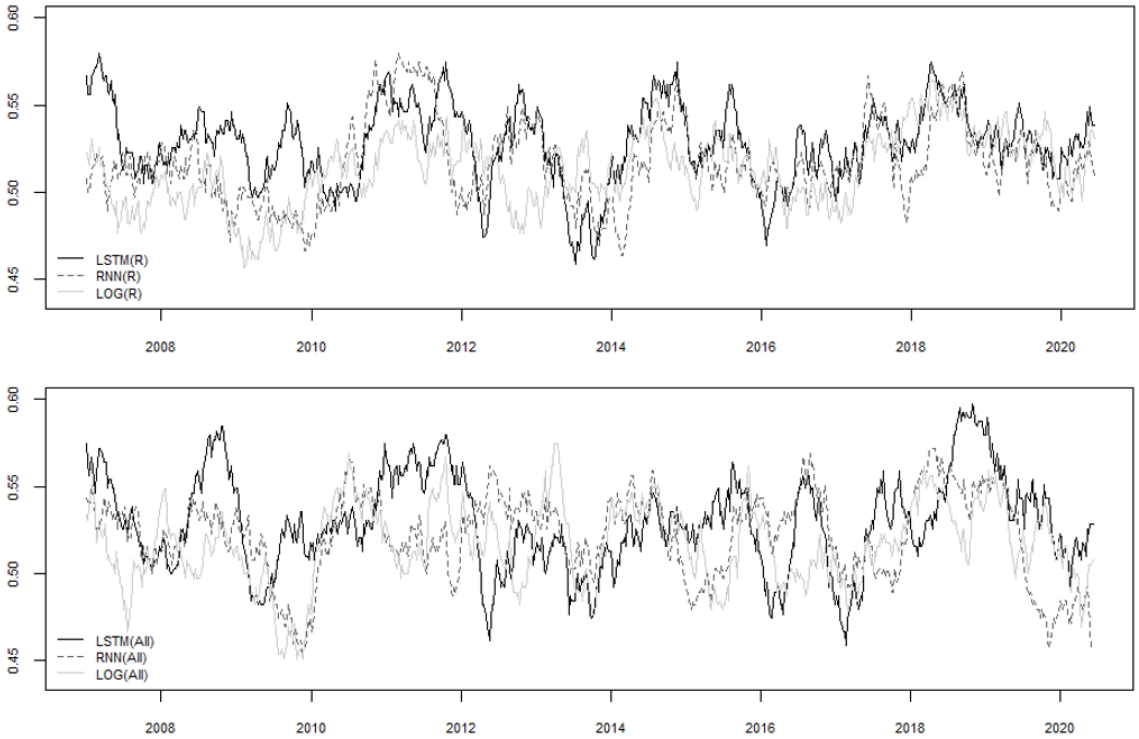


Figure 5. Time series of 26-week rolling classification accuracies

The top (bottom) figure displays the time series of 26-week rolling classification accuracies of the univariate (multivariate) models. The sample of the 26-week rolling accuracies covers the period from the 1st of January 2007 until the 8th of June 2020.

Substantial co-movement can be observed in the rolling accuracies of different models, within both the univariate and the multivariate models. Similarities in the time varying accuracies between models indicate that there is significant time variation in the predictability of stock returns. There is a large body of evidence in the literature suggesting that stock return predictability varies over time and tends to be higher during market turmoil, such as during the global financial crisis or the European debt crisis. In addition, different variables are shown to have varying predictive power over different periods during the business cycle. (see, e.g., Daniel & Moskowitz 2016, Farmer et al. 2019, Fischer & Krauss 2018, Neely et al. 2014.)

All of the proposed models face a steep decline in accuracy somewhere between the end of 2008 and the latter half of 2009. The decline is more long-lasting among the benchmark methods, whereas the performance of both LSTMs starts improving around the early stages of 2009. Furthermore, there are other notable downturns in predictability, e.g., around 2012–2013 and in 2016.

Three periods with moderately higher predictability can be identified. During the period between 2010 and 2012, most of the models regularly achieved an accuracy over 50 percent, and the same occurred around 2014–2015 and 2018–2019. However, there are also periods where some models deliver divergent performance, and periods where notable slumps and surges occur with various delays. Without a few exceptions, the accuracies of all three univariate models exhibit similar movements throughout the sample and the co-movement is more apparent towards the end of 2010s. On the other hand, there are slightly more differences in the accuracies between the multivariate models.

To evaluate the difference between univariate and multivariate models, Figure 6 displays the time series of the difference between the 26-week rolling classification accuracies of *LSTM (All)* and *LSTM (R)* models. The difference is computed by deducting the 26-week rolling accuracy of the *LSTM (R)* from the 26-week rolling accuracy of *LSTM (All)*. The difference fluctuates on both sides during the sample period, the most significant difference coming around 2019 in the favor of *LSTM (All)*. However, there are also several occasions when the univariate LSTM outperforms the multivariate LSTM, most notably during 2012, 2014 and 2017. The *LSTM (All)* seems to outperform the *LSTM (R)* mostly during times when the average prediction accuracies are on a higher level and vice-versa. These findings suggest that during periods with higher predictability the multivariate model benefits from the supplementary information provided by the additional features.

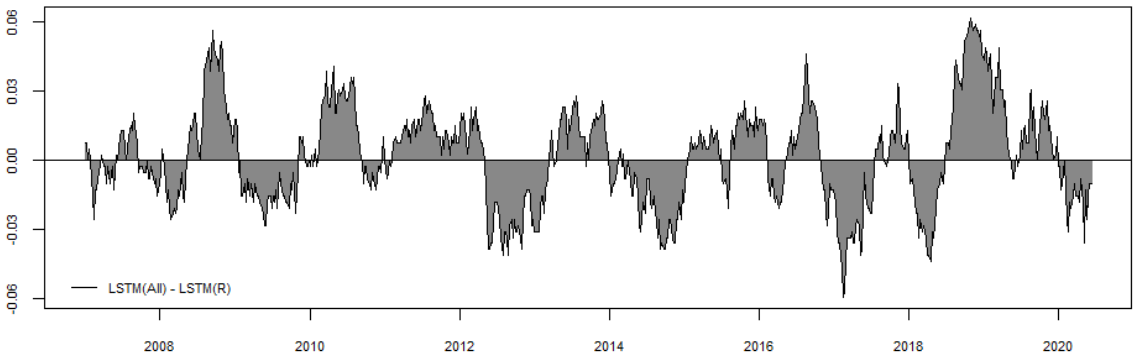


Figure 6. LSTM (All) – LSTM (R) 26-week rolling classification accuracy

The figure displays the difference between the 26-week rolling classification accuracies of the *LSTM (All)* and *LSTM (R)* models. The sample covers the period from the 1st of January 2007 until the 8th of June 2020

According to Bahrammirzaee (2010), one detail that needs to be noted when predicting with neural networks is that some patterns can be present in the prediction set of a network that have not been present in the training set, especially during events such as the global financial crisis. This might lead to decreasing accuracy as the models rely on the learned dependencies and cannot react to new and unique events quickly enough. In other words, even though the network learns the meaningful dependencies in the training set and generalizes well on the validation set, the performance on the prediction set suffers due to constantly changing market conditions. All models in this research are re-trained in an annual frequency, which makes them even more vulnerable to such circumstances.

5.2 Investment strategy performance

The results of Fischer and Krauss (2018) show that reducing the number of stocks held in a portfolio increases the profitability of the prediction-based trading strategies significantly. However, the S&P 500 index includes five hundred stocks as opposed to the fifteen stocks available in this research. Therefore, in order to take advantage of the certainty of different predictions, but also keeping the number of stocks in the portfolio high enough, the *k* number of stocks kept in a portfolio each week is set to be five and no other variations are considered.

The summary statistics for the prediction-based portfolios and the simple buy-and-hold portfolio prior to transaction costs are presented in Table 6. The buy-and-hold portfolio is constructed by acquiring all fifteen stocks in the beginning of the trading window and keeping them in the portfolio until the final date. The full trading window covers the period from the 3rd of July until the 8th of June 2020.

Table 6. Portfolio summary statistics prior to transaction costs 2006–2020

The table reports annualized mean return, standard deviation, and Sharpe ratio of the constructed portfolios with a weekly observation frequency. In addition, the skewness, excess kurtosis, maximum drawdown (MDD) and sample 95% Value-at-Risk of the portfolio returns are reported. The buy-and-hold portfolio includes all the presented fifteen stocks, whereas the prediction-based portfolios include five stocks with the highest outperforming probabilities each week within the prediction sample. The sample covers the period from the 3rd of July 2006 until the 8th of June 2020.

	Return	Std. dev.	Sharpe	Skewness	Kurtosis	MDD	VaR 95%
LSTM (R)	0.112	0.253	0.441	-0.346	4.322	0.624	0.304
RNN (R)	0.092	0.262	0.353	-0.378	4.879	0.592	0.338
LOG (R)	0.037	0.268	0.136	-0.198	4.356	0.690	0.404
LSTM (All)	0.127	0.277	0.459	-0.153	6.912	0.700	0.328
RNN (All)	0.100	0.274	0.366	-0.174	4.769	0.682	0.350
LOG (All)	0.066	0.266	0.246	-0.210	2.927	0.651	0.372
Buy-and-hold	0.082	0.242	0.338	-0.264	3.913	0.642	0.316

The annualized returns of the *LSTM (R)* portfolio prior to transaction costs are 11.2% compared to 9.2% for the *RNN (R)* and 3.7% for the *LOG (R)*. Furthermore, the annualized returns of the multivariate models are 12.7 % for the *LSTM (All)*, 10.0% for the *RNN (All)* and 6.6% for the *LOG (All)*. The simple buy-and-hold strategy achieves an annual return of 8.2% during the sample period from the 3rd of July 2006 until the 8th of June 2020. Logistic regression models generate substantially lower annual returns even compared to the buy-and-hold portfolio, whereas the LSTMs clearly outperform all benchmark portfolios with RNNs falling somewhere between the LSTMs and buy-and-hold portfolio. With respect to standard deviation, the *LSTM (R)* portfolio is less volatile compared to the other prediction-based models with 25.3%, only falling behind to the simple buy-and-hold portfolio that has the lowest annualized standard deviation of 24.2%. On the other hand, the *LSTM (All)* portfolio delivers the highest annualized standard deviation of 27.7%.

All the weekly portfolio returns are negatively skewed with *LSTM (All)* having the least negative skewness of -0.153. Furthermore, all portfolio returns have excess kurtosis with *LSTM (All)* exhibiting significantly higher excess kurtosis compared to any other portfolio. The excess kurtosis of *LSTM (All)* returns is 6.912, compared to 3.913 of the buy-and-hold returns. These findings are similar to findings of Fischer and Krauss (2018) and Krauss et al. (2017) who report significantly higher excess kurtosis for their machine learning -based portfolios compared to those of benchmark portfolios. However, the *LSTM (All)* is the only portfolio that stands out in that regard.

The Sharpe ratio measures the return of a portfolio in relation to its risk, and it can be interpreted as the signal-to-noise ratio. Even though the *LSTM (All)* portfolio bears the highest standard deviation of all portfolios, it also delivers the highest annualized Sharpe ratio of 0.459, thus compensating the higher risk with higher returns. Furthermore, the *LSTM (R)* portfolio also has relatively high Sharpe ratio of 0.441, compared to 0.366 of the *RNN (All)*, 0.353 of the *RNN (R)*, 0.338 of the buy-and-hold portfolio, 0.246 of the *LOG (All)* and 0.136 of the *LOG (R)*. Even the highest Sharpe ratio does not reach the same levels as the best performing portfolios in Fischer and Krauss (2018), Krauss et al (2017) and Long et al. (2019). However, all those studies are conducted with a daily observation frequency and on different markets and thus the results are not directly comparable.

Maximum drawdown measures the highest observed loss from a historical peak to a trough of a portfolio. Value-at-Risk, in turn, quantifies the level of risk exposure in a portfolio by demonstrating the degree and frequency of potential losses. The annual 95% Value-at-Risk measures the maximum percentage loss in a year at a 95% confidence level. Both, the maximum drawdown and the Value-at-Risk measure the downside risk of a portfolio over time. Among the prediction-based portfolios the *LSTM (R)* and *LSTM (All)* deliver the lowest Value-at-Risk with 30.4% and 32.8%, respectively. The former has even less downside risk than the buy-and-hold portfolio that has 31.6% Value-at-Risk. The *LSTM (R)* portfolio shows desirable risk characteristics having the lowest annual Value-at-Risk and also the second lowest annual maximum drawdown of -62.4%. The lower risk exposure of machine learning -based portfolios is also supported by the findings of Fischer and Krauss (2018), Krauss et al. (2017) and Nelson et al. (2017).

The maximum drawdowns of the portfolios range from 59.2% to 70.0%, but the main driver for those values is the global financial crisis and its aftermath around 2009. In order to visualize the portfolio returns over time, the cumulative compounded portfolio returns prior to transaction costs for the full sample period are displayed in Figure 7. From the plotted time series, another steep dive in the portfolio values can be observed during the COVID-19 outbreak in the early 2020, when all portfolios faced a sharp decline in value. The cumulative return figures display how radical the collapse is during the first months of 2020. Looking at the cumulative returns of the full sample in Figure 7, the LSTM portfolios seem to start gaining their edge around 2010. In addition, around 2019 all portfolios face a slump in cumulative returns, except for the *LSTM (All)* portfolio that continues with a strong trend until the early 2020.

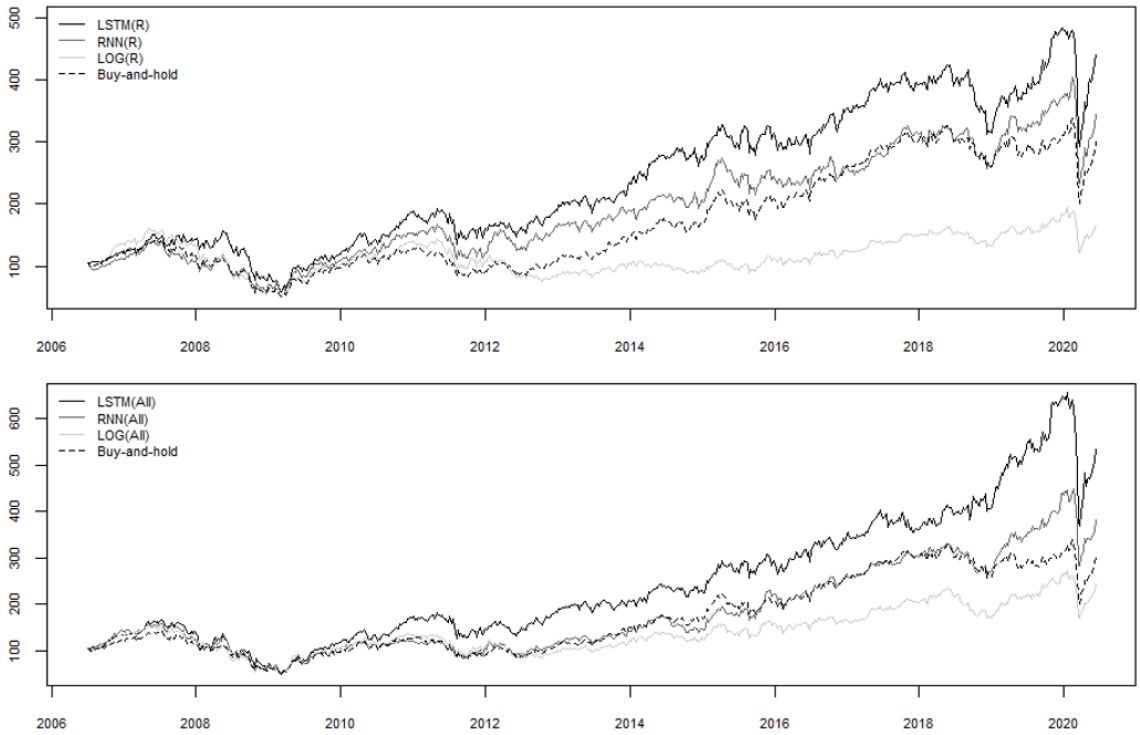


Figure 7. Cumulative compounded returns prior to transaction costs 2006–2020

The top (bottom) figure displays time series of the cumulative compounded portfolio returns of the univariate (multivariate) models and a buy-and-hold method prior to transaction costs. The time series are displayed with a weekly observation frequency. The full sample covers the period from the 3rd of July 2006 until the 8th of June 2020.

The assumed fixed trading costs of five basis points underestimate the total transaction costs and might give an overly positive view on the gained profits. The actual bid-ask spread accounts for a substantial part of the total transaction costs and would most likely reduce the profitability of the trading strategies if implemented in practice. The bid-ask spread also tends to rise during high market turmoil, which would potentially cut the gained profits even more. In addition, the implemented bet sizing produces additional costs, as even though a single stock would not be bought or sold during a particular week, adjusting its weight in the portfolio incurs costs. However, this is not accounted separately in the calculations of the transaction costs but is rather included in the relatively high cost of five basis points per a single buy or sell trade. Evaluating the profitability of an investment strategy after transaction costs is always difficult, as the accounted transaction costs are, at best, only estimates.

Table 7 provides insights to the portfolio performance after transactions are considered. Similar to the summary prior to transaction costs, annualized mean return, standard deviation and Sharpe ratio are presented. In addition, the total number of transactions and the percentage cutback of profits caused by the transaction costs are reported.

Table 7. Portfolio summary statistics after transaction costs 2006–2020

The table reports annualized mean return, standard deviation, and Sharpe ratio of the constructed portfolios with a weekly observation frequency. In addition, the total number of transactions and the percentage cutback caused by the transaction costs are reported. The sample covers the period from the 3rd of July 2006 until the 8th of June 2020.

	Return	Std. dev.	Sharpe ratio	Transactions	Cutback
LSTM (R)	0.094	0.253	0.373	10 620	-0.198
RNN (R)	0.070	0.262	0.268	10 806	-0.249
LOG (R)	0.015	0.268	0.058	10 976	-0.250
LSTM (All)	0.111	0.277	0.402	10 362	-0.181
RNN (All)	0.078	0.274	0.285	11 102	-0.250
LOG(All)	0.042	0.266	0.159	11 279	-0.266

Even after transaction costs are considered, the *LSTM (All)* portfolio delivers annualized mean return of 11.1% and a Sharpe ratio of 0.402. The *LSTM (R)* places second in both measures. More interestingly, both LSTM portfolios have considerably less transactions made during the sample period from the 3rd of July 2006 until the 8th of June 2020. The number of transactions in the *LSTM (All)* portfolio is 10 362, compared to the 10 620 in the *LSTM (R)* portfolio and 10 806 in the *RNN (R)* portfolio. Due to smaller amount of transactions, the transactions also cut less out of the portfolio profits than with the benchmark portfolios. The transaction costs cut out 18.1% of the *LSTM (All)* profits and 19.8% of the *LSTM (R)* profits, whereas on average the benchmark method portfolios suffer a profit loss of approximately 25% after transaction costs are considered. The LSTM portfolios thus provide more stability, possibly due to the predictions being based on longer-term patterns in comparison to the other models.

The weak form efficiency conditions of the Efficient Market Hypothesis claim that securities prices reflect all historical information, and there is no potential for outsized risk-adjusted returns through technical analysis (Fama 1970). The high-level performance of the *LSTM (R)* model suggests that Finnish stock market does not satisfy these conditions, as the model is able to outperform a simple buy-and-hold strategy after transaction costs utilizing only lagged stock returns in the predictions.

The cumulative compounded portfolio returns after transaction costs are displayed in Figure 8 with a weekly observation frequency. Time series of the buy-and-hold portfolio returns is also displayed along with other portfolios in order to assess whether the prediction-based portfolios maintain their edge when transaction costs are considered. The full prediction sample covers the period from the 3rd of July 2006 until the 8th of June 2020.

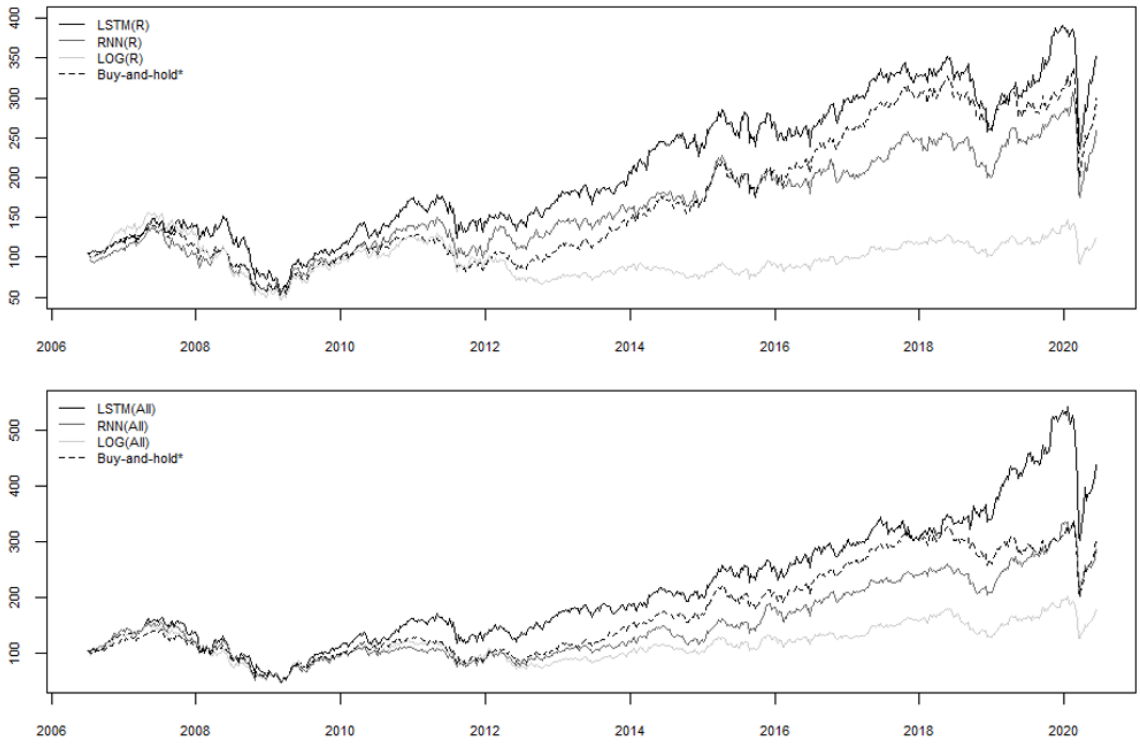


Figure 8. Cumulative compounded returns after transaction costs 2006–2020

The top (bottom) figure displays time series of the cumulative compounded portfolio returns of the univariate (multivariate) models and a buy-and-hold method after transaction costs. The time series are displayed with a weekly observation frequency. The full sample covers the period from the 3rd of July 2006 until the 8th of June 2020.

Only the LSTM portfolios are able to outperform the buy-and-hold portfolio when transaction costs are taken into account, and from those the *LSTM (R)* only barely maintains its edge. Figure 8 demonstrates how most of the excess profits of the *LSTM (All)* portfolio develop after the second quarter of 2018. Therefore, the subperiod from the beginning of 2018 until the first month of 2020 is taken under further review.

5.3 Subperiod analysis

The performance of LSTM portfolios is evaluated within different subperiods in order to provide more detail about the predictive performance and risk-profile of the portfolios over time. Subperiod analysis is performed in two steps. First, the prediction sample is divided into two split-periods, the first ranging from the 3rd of July 2006 until the 31st of December 2012, and the second from the 7th of January 2013 until the 8th of July 2020. Next, the period from the 1st of January 2018 until the 27th of January 2020 is analyzed to gain better understanding about the performance of *LSTM (All)* portfolio during that period. Table 8 reports the summary statistics of portfolios within the two split-periods.

Table 8. Portfolio summary statistics 2006–2012 and 2013–2020

The table reports annualized mean return, standard deviation, and Sharpe ratio of the buy-and-hold (BAH) and both LSTM portfolios with a weekly observation frequency. In addition, the skewness, excess kurtosis, maximum drawdown (MDD) and sample 95% value-at-risk of the portfolio returns are reported. The first split-period ranges from the 3rd of July 2006 until the 31st of December 2012, and the second from the 7th of January 2013 until the 8th of July 2020.

	2006–2012			2013–2020		
	LSTM (All)	LSTM (R)	BAH	LSTM (All)	LSTM (R)	BAH
Return	0.045	0.047	0.006	0.073	0.057	0.067
Std. dev.	0.331	0.296	0.203	0.220	0.208	0.203
Sharpe	0.135	0.159	0.029	0.332	0.273	0.332
Skewness	0.233	0.034	-1.213	-1.246	-1.268	-1.213
Kurtosis	4.742	2.103	7.552	9.627	8.917	7.552
MDD	0.700	0.624	0.642	0.440	0.395	0.405
VaR 95%	0.499	0.440	0.328	0.289	0.285	0.266

During the first split-period between 2006 and 2012, the simple buy-and-hold generates annual returns of only 0.6%, whereas in comparison the *LSTM (All)* and *LSTM (R)* portfolios deliver annualized returns of 4.5% and 4.7%, respectively. Furthermore, the *LSTM (R)* returns are less volatile compared to *LSTM (All)* and have the lowest maximum drawdown among all portfolios. The returns in the buy-and-hold portfolio have high negative skewness, whereas the returns of both LSTM portfolios are positively skewed. The univariate LSTM delivers the best overall performance during the first split-period that also includes the time of the global financial crisis.

During the first split-period the buy-and-hold portfolio delivers poor performance, but during the second split-sample the buy-and-hold portfolio is on level with the best performing *LSTM (All)* portfolio in terms of Sharpe ratio, and comes close on annual returns, while also outperforming the *LSTM (R)* portfolio in both measures. The buy-and-hold portfolio also delivers lower downside risk compared to the multivariate portfolio. Furthermore, the multivariate LSTM performs better than the univariate LSTM during the second split-period, unlike in the first split-period where the situation is the opposite.

Fischer and Krauss (2018), and Krauss et al. (2018) show that the profitability of their trading strategies starts to diminish during the 2010s. For example, Fisher and Krauss (2018) find that during the 2010s, the profitability of their strategy fluctuates around zero after transaction costs are considered. Similar pattern can be observed from the results of the subsample analysis in Table 8, as from 2013 onwards, the buy-and-hold

portfolio is on par with the best performing *LSTM (All)* portfolio prior to transaction costs. Moreover, adding transaction costs to the results would cut out the last bit of edge that the *LSTM (All)* portfolio might have. Despite generating annual returns of 7.3%, the multivariate LSTM portfolio fails to compensate the higher risks with additional returns, when put into comparison with the buy-and-hold portfolio.

Even though the LSTM portfolios deliver higher annual returns and Sharpe ratios during the second split-period than during the first split-period, their performance in relation to the benchmark buy-and-hold portfolio is significantly better during the first split-period. Chen et al. (2003) argue that one of the main reasons that their neural network - based investment strategy manages to outperform a simple buy-and-hold strategy is that the Taiwanese financial sector faced a sharp plunge during the test period. The economic crisis led to poor performance of the buy-and-hold portfolio but did not have that large effect on the neural network portfolio returns. The findings of Chen et al. (2003) are similar to the ones in this subperiod analysis, as the relative performance of the buy-and-hold portfolio is substantially worse during the period including the global financial crisis.

During the full trading window, the *LSTM (All)* portfolio delivers the highest annual mean return and Sharpe ratio out of all constructed portfolios with and without transaction costs. Looking at the cumulative compounded returns in Figure 7 and Figure 8, a substantial part of this outperformance can be attributed to the last two years of the prediction sample. In order to evaluate the sources of this profitability, another, alternate subperiod review is conducted starting from the 1st of January 2018. The cumulative compounded returns in Figure 7 and Figure 8 display how the COVID-19 outbreak, among other possible factors, caused a steep decline in all portfolio values during the first months of 2020. Consequently, this period of declining returns is excluded from this subperiod analysis and the analysis focuses on the period until the 27th of January 2020, right before all the portfolio returns start plummeting.

The second subperiod review is carried out by looking into stock-specific predictive performance and the structure of the *LSTM (All)* portfolio during the period ranging from the 1st of January 2018 until the 27th of January 2020, a period of 109 weeks. For this subsample, the cumulative compounded returns of all portfolios prior to transaction are displayed in Figure 9, where the return time series are displayed with a weekly observation frequency. Time series of the univariate portfolios are plotted separately in one figure and time series of the multivariate portfolios on another. Performance of the buy-and-hold portfolio is plotted on both figures for the ease of comparison.

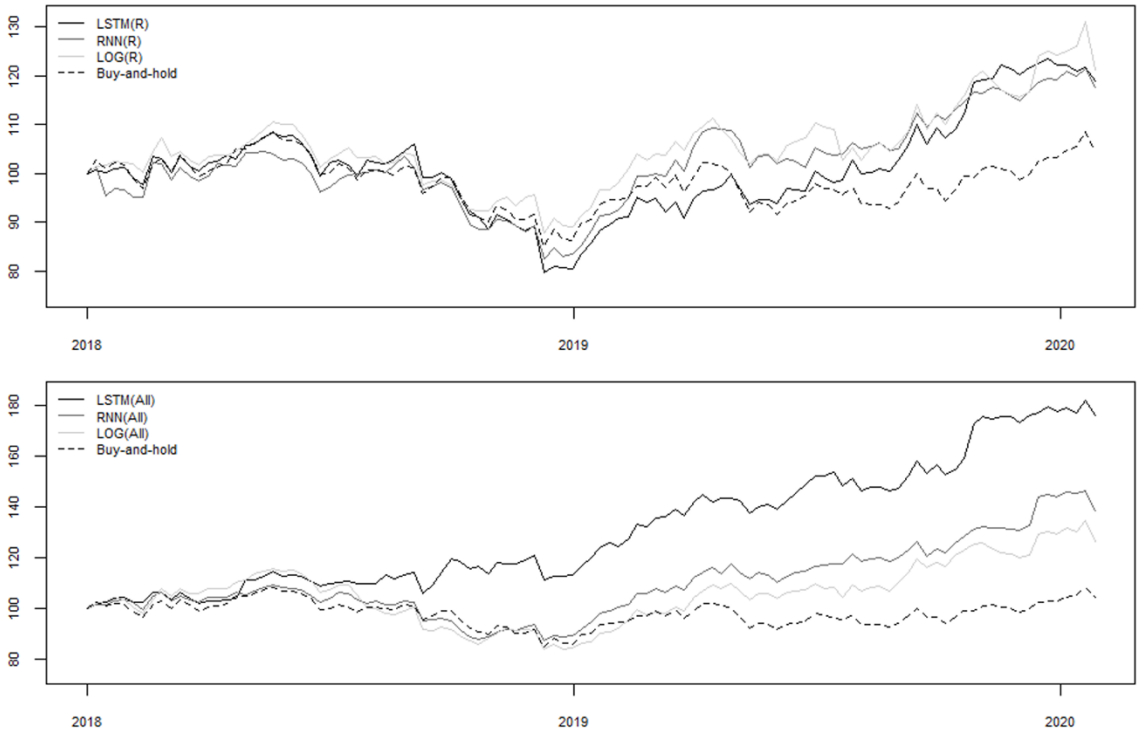


Figure 9. Cumulative compounded returns prior to transaction costs (2018–2020)

The top (bottom) figure displays time series of the cumulative compounded portfolio returns of the univariate (multivariate) models and a buy-and-hold method prior to transaction costs. The time series are displayed with a weekly observation frequency. The subsample covers the period from the 1st of January 2018 until the 27th of January 2020.

Due to the relatively small number of stocks available, all portfolios exhibit substantial co-movement throughout the years. However, in around the latter half of 2019, all portfolios face a notable decline in cumulative returns, except for the *LSTM (All)* portfolio that continues with a strong trend until the early 2020. This finding suggest that the multivariate LSTM model is able to capture some information from the data that other models completely miss out. In order to gain better understanding behind this outperformance, a stock-specific analysis of the *LSTM (All)* portfolio is presented in Table 9.

Total cumulative returns of each stock during the subperiod are presented in the first column of Table 9. During the 109 weeks in the subperiod, Outokumpu A and Finnair lost over half of their value, and just over a third of Wärtsilä's value got depleted. The *LSTM (All)* model correctly classifies these stocks over 50 percent of the time and manages to mostly avoid investing in these plummeting stocks. In fact, Outokumpu A is not selected a single time in the portfolio during these 109 weeks, while Finnair is selected only four times and Wärtsilä just 17 times. Overall, investments on stocks with negative cumulative returns over the subperiod is relatively low, except regarding Uponor, which is selected 50 times and has a total cumulative return of -20.1% during the subperiod.

Table 9. Stock-specific analysis of the LSTM (All) portfolio 2018–2020

The table provides stock-specific analysis of LSTM (All) portfolio during the subperiod from the 1st of January 2018 until the 27th of January 2020, a total of 109 weeks. The first column reports the cumulative compounded return of a stock in the row label. The second column reports stock-specific prediction accuracies of the LSTM (All) model. Third column reports the number of weeks that a stock in the row label is held in the LSTM (All) portfolio during the subperiod. The final column reports, in percentage, the average weekly return gained from investments on the stock in the column label during the subperiod.

Stock	Total return	Prediction accuracy	Weeks in portfolio	Avg. Return
Nokia	-0.021	0.560	5	-0.027
Sampo A	0.017	0.578	58	-0.013
Fortum	0.565	0.596	43	0.096
UPM-	0.182	0.560	58	0.003
TietoEVRY	0.231	0.569	22	0.059
Huhtamäki	0.197	0.468	71	0.203
Kemira	0.283	0.569	46	0.193
Kesko B	0.460	0.633	84	0.498
Neles	0.280	0.450	34	0.253
Wärtsilä	-0.348	0.505	17	-0.017
Outokumpu A	-0.569	0.523	0	0.000
Atria A	-0.118	0.550	6	0.148
Bittium	0.281	0.523	47	0.160
Finnair	-0.549	0.514	4	-0.057
Uponor	-0.201	0.606	50	0.029

The *LSTM (All)* model achieved some impressive prediction accuracies during the subperiod with, e.g., 63.3% for Kesko B, 60.6% for Uponor, 59.6% for Fortum and 57.8% for Sampo A. On the other hand, the model achieves an accuracy lower than 50 percent regarding Huhtamäki and Neles but still manages to achieve average weekly returns of over 0.2 percent with both stocks. Substantial part of the investments is made on stocks with an upward trend during the subperiod. In addition, even though Uponor lost 20.1% of its value during the subperiod, on average the *LSTM (All)* model received positive weekly returns for investments on that stock.

The exceptional performance of the *LSTM (All)* portfolio during the period from the 1st of January 2018 until the 27th of January 2020 is therefore mainly driven by the fact that most of the time it avoids investing in plunging stocks. During the subperiod the model provides high classification accuracies regarding most of the stocks and is thus able to time the investments well, like in the case regarding Uponor.

5.4 Common patterns and sources of profitability

Machine learning models have been proven to be capable of extracting subtle patterns from the data without any prior knowledge about them. Fischer and Krauss (2018) find that stocks selected for trading by their LSTM model shared similar characteristics; below-mean momentum, high volatility prior to trading and a tendency for mean-reversion. Furthermore, Krauss et al. (2017) examine the sources of profitability of their ensemble method -based portfolio and find that the model primarily utilizes the returns from the past 12-month and one-week periods when selecting stocks for trading. In other words, both Fischer and Krauss (2018) and Krauss et al. (2017) show that machine learning models can independently pick up patterns that are typical for some well-known market anomalies, such as momentum and short-term reversal.

Based on the findings of Fischer and Krauss (2018) and Krauss et al. (2018), a similar analysis is conducted in order to examine the patterns that the stocks exhibit during the last year prior trading. The analysis is executed separately for each portfolio as follows. First, for each week and each top-5 stock in the portfolio, the weekly returns from the past 52 weeks prior trading are collected in a vector $\boldsymbol{m}$ of dimension $52 \times 5T$, where T is the number time steps in the trading window. Next, these one-week returns are averaged separately for each week from $t - 52$ until $t - 1$. Finally, these averaged one-week returns are accumulated by indexing the starting level of 0 on the day $t - 52$ in order to generate a time series of the average accumulated returns prior to the selection for trading. These averaged time series are depicted in Figure 10 and are contrasted to the mean return of the buy-and-hold portfolio.

The top-5 stocks that are selected in the portfolio by the *LSTM (R)*, *LSTM (All)* and *RNN (R)* models exhibit declining cumulative returns approximately two months prior trading. Moreover, the cumulative returns eventually fall below the mean level during the last two weeks before being picked up in the portfolio. Fischer and Krauss (2018) find similar patterns as their LSTM network go long on stocks that start losing value at an accelerating pace during the last nine days prior selection. The findings of Fisher and Krauss (2018) are more evident, as during those nine days, the stocks lose around 50 percent of the gains from the past year. In this research a weekly observation frequency is used, and thus the results cannot be directly compared. However, these characteristics are fairly similar, since the nine-day period of declining returns in Fischer and Krauss (2018) corresponds to the two-week period of declining returns in this research.

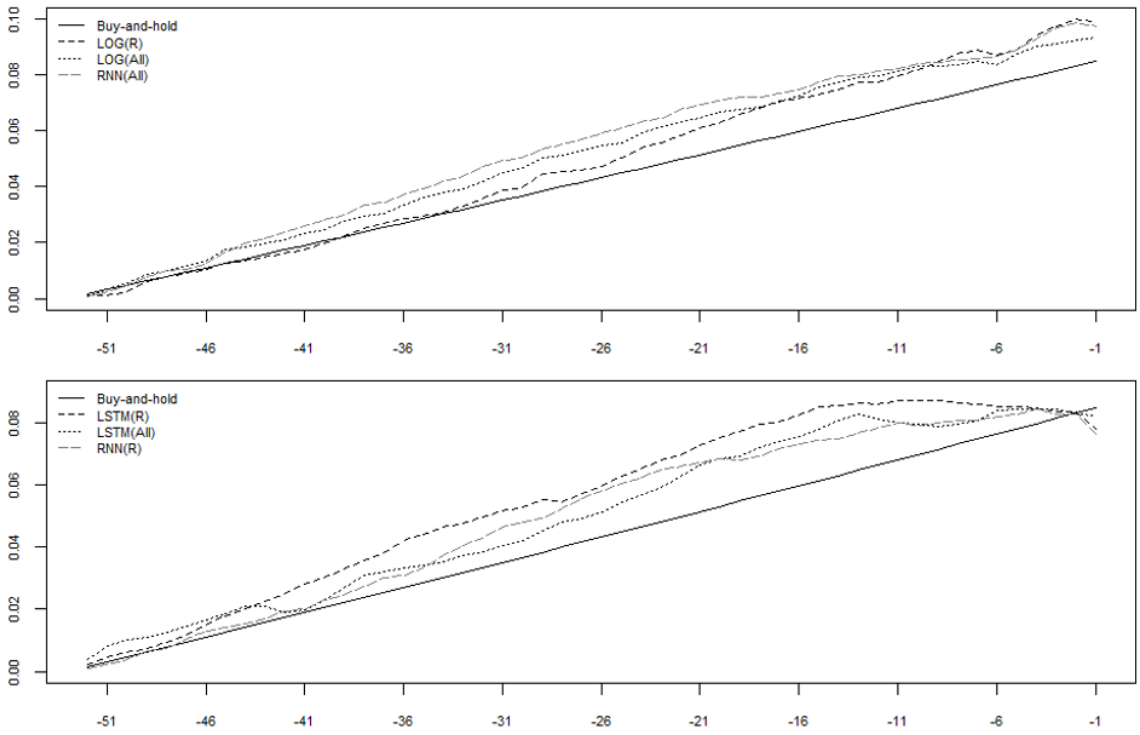


Figure 10. Average accumulated stock returns prior to selection for trading

The figure displays average accumulated stock returns from the past 52 days prior to selection for trading at day t. The x-axis displays the distance to the trading day t, while the y-axis displays the level of cumulative returns. The return indices are formed from a sample period from the 4th of July 2005 until the 8th of June 2020.

It is not a big of a surprise that the memory-free logistic regression models are not able to pick up these types of patterns from the data. However, the fact that the *RNN (All)* failed to capture these patterns is unanticipated in the sense that *RNN (R)* was successful on the same task. One possible explanation for that is that unlike with LSTM networks that converged well during the training process, the *RNN (All)* model was very prone to overfitting.[4] This might be explained by the large amount of input features combined with a complex model architecture. Network hyperparameters in this research are optimized for the LSTM models, and those same hyperparameters are also applied to corresponding RNN models. Therefore, even though the *RNN (All)* show slightly better performance compared to the *RNN (R)* model and it outperforms both logistic regression models in almost every measure, the performance seems to be non-optimal and the model might get distracted by the additional complexity introduced in the multivariate approach.

4 See appendices for the visualization of the LSTMs training process during the final study period

In addition to the accumulated return indices visualized in Figure 10, mean stock returns within various periods prior trading are computed. In Table 10, mean stock returns are reported for the past one-week, 6-month and 12-month periods prior to trading.

Table 10. Mean stock returns within different periods prior to trading

The table reports mean stock returns within different periods depicted in the row label, for the stocks picked by a model in the column label. The predictions and portfolio asset allocations are made at time t. The returns are denoted in percent.

	LSTM (R)	RNN (R)	LOG (R)	LSTM (All)	RNN (All)	LOG (All)
[t–1, t]	-0.106	-0.183	-0.082	0.025	-0.147	0.041
[t–26, t]	0.207	0.156	0.195	0.217	0.188	0.179
[t–52, t]	0.235	0.201	0.207	0.253	0.212	0.204

The top-5 stocks selected for trading exhibit significant negative returns one week prior to the prediction, except for the *LOG (All)* and *LSTM (All)* models. Even though the mean one-week returns are positive for the *LSTM (All)*, they are still significantly lower compared to the mean returns during the past 6-month and 12-month periods. Moreover, the *LSTM (All)* models shows divergent behavior by selecting stocks that have high average returns during the past 12-month period. These findings suggest that the multivariate LSTM model relies more on longer-term patterns in comparison to the other models. Krauss et al. (2017) also find that the past one-week and 12-month returns are the most prominent features in making one-day ahead predictions of stock return movements.

Leveraging these findings, a one-week mean-reversion portfolio (*MR1*), and a 12-month momentum portfolio (*MOM12*) are constructed in order to capture the quintessence of the patterns that the models exhibit when picking up stocks for trading. The *MR1* portfolio is constructed by selecting each week the top-5 stocks with the most negative one-week returns prior to the week of selection. Moreover, the *MOM12* portfolio is constructed by selecting each week the top-5 stocks with the highest average returns from the past 12-month period. The annualized mean return, standard deviation and Sharpe ratio for the *MR1* and *MOM12* portfolios are depicted in Table 11, both with and without transaction costs. The portfolios that account for transaction costs are denoted with (*T*) after the name of the portfolio. Moreover, the total number of transactions and the percentage cutback caused by the transaction costs are reported. The full trading window covers the period from the 3rd of July 2006 until the 8th of June 2020.

Table 11. Mean-reversion and momentum portfolios 2006–2020

The table reports annualized mean return, standard deviation, and Sharpe ratio of the mean-reversion (MR1) and momentum (MOM12) portfolios with a weekly observation frequency. In addition, the total number of transactions and the percentage cutback caused by the transaction costs are reported. Portfolios with transaction costs are also reported and denoted with (T) after the name of the portfolio. The sample covers the period from the 3rd of July 2006 until the 8th of June 2020.

	Return	Std. dev.	Sharpe ratio	Transactions	Cutback
MR1	0.121	0.281	0.432	-	-
MR1 (T)	0.096	0.281	0.342	11 303	-0.376
MOM12	0.085	0.254	0.335	-	-
MOM12 (T)	0.069	0.254	0.273	10 561	-0.228

The *MR1* portfolio delivers an annual return of 12.1%, outperforming all but the *LSTM (All)* portfolio before transaction costs are considered. However, due to the relatively higher standard deviation of 28.1%, the *MR1* portfolio also takes more risk, delivering a Sharpe ratio of 0.432. Therefore, in terms of risk-return profile it falls behind the *LSTM (All)* and the *LSTM (R)*, that generate Sharpe ratios of 0.459 and 0.441, respectively. When transaction costs are taken into account, the total profits in the *MR1 (T)* portfolio drop by 37.6%. In comparison, the transaction costs only cut under 20 percent off from the LSTM portfolios, since both LSTMs performed substantially lower amount of transactions during the trading window. The mean-reversion -based trading strategy barely manages to outperform the buy-and-hold portfolio after transaction costs are considered.

The *MOM12* portfolio does not reach anywhere the same level of performance with the mean-reversion portfolio, as it delivers an annual return of 8.5% prior to transaction costs, and just 6.9% when transaction costs are considered. The percentage cutback caused by the transaction costs is 22.8%, which is less compared to the *MR1 (T)* portfolio but higher compared to both LSTM portfolios. The number of transactions is considerably less than with the *MR1 (T)*, which is natural since the 12-month momentum portfolio relies on longer-term signals compared to the one-week mean-reversion portfolio. However, the number of transactions in the *LSTM (All)* portfolio is even lower (10 362) than in the *MOM12 (T)* portfolio. Both LSTM portfolios maintain their edge in terms of profitability much better compared to all benchmark models, as well as compared to the proposed mean-reversion and momentum portfolios when transactions are considered. This mostly attributes to the substantially lower number of transactions made.

These findings support the existence of pricing irregularities and stock market predictability, as a strategy investing to the underperforming stocks of the past week yields

substantially higher returns compared to a simple buy-and-hold portfolio. On the other hand, the momentum-based investment strategy shows relatively poor performance. According to Daniel and Moskowitz (2016), traditional momentum strategies are vulnerable to occasional, but persistent periods of negative returns that often occur during high market turmoil, which might partially explain the poor performance of the *MOM12* portfolio.

Even though the simplified mean-reversion strategy yields annual returns almost as high as the *LSTM (All)* portfolio, it has less desirable risk-return profile. Both portfolios also utilize different type of predictive signals, with the multivariate LSTM portfolio relying on longer-term predictive signals in the sense that it performs the least transactions among all portfolios and picks up stocks with high average returns during the past 12-month period. Thus, a simplified mean-reversion behavior does not completely explain the reasoning behind the patterns the *LSTM (All)* model utilizes to select stocks for trading. Therefore, some subtle patterns must exist that only the more advanced model is able to extract from the noisy stock market data.

The LSTM networks partly rely on the same patterns with well-known capital market anomalies without any prior knowledge of these patterns. More interestingly, also the *LSTM (R)* and *RNN (R)* networks are capable of identifying some of these patterns using only the past 60-week standardized returns as input features. The capability of LSTM networks to pick up these patterns independently is also shown by Fischer and Krauss (2018).

5.5 Robustness analysis

As stated before, unlike most statistical models, neural networks are stochastic and thus make different predictions each time the constructed model is trained on the same dataset. According to López de Prado (2018), one should be careful when evaluating the true power of machine learning models, as some of them might provide good results just by chance. However, these credibility issues can be tackled by performing different tests on model robustness. A machine learning model is to large extent a summation of the model configuration, training procedure and the quality of the data. Machine learning models should be evaluated on case-by-case basis as each model is different, and usually constructed on that specific task. Reliable evaluation of different models is also difficult in the sense that two completely different models can achieve exact same results.

In order to get a better overview on the generalization capacity of the proposed LSTM networks, their robustness is evaluated by comparing the classification accuracies across several runs. Table 12 provides the results of a robustness check, where each LSTM network was run 20 times and the minimum, maximum and average classification accuracies across those runs were computed.

Table 12. LSTM network classification accuracies across 20 runs

The table reports the minimum, maximum and average classification accuracies of the proposed LSTM networks across 20 runs. The accuracies are denoted in percent.

Accuracy	LSTM (R)	LSTM (All)
Minimum	52.26	52.71
Maximum	53.04	53.36
Average	52.77	53.13

Both LSTMs show good robustness in the sense that the classification accuracies do not drop below 52 percent a single time. In fact, both of the LSTMs constantly deliver higher classification accuracies than any of the benchmark models. The average classification accuracy of the *LSTM (R)* model matches the accuracy obtained in the first run, whereas the average accuracy of *LSTM (All)* model is in fact higher than the 53.0% obtained in the first run. In other words, the performance of the *LSTM (All)* model should, in theory, be even better than the results in this research would indicate. However, Welch and Goyal (2008) point out that even if a predictive model would be powerful in a research perspective, i.e., has good both in-sample and out-of-sample performance, great confidence on the model is required in order to use it for investing purposes in the long-term.

6 CONCLUSIONS

This research focused on investigating whether long short-term memory (LSTM) neural networks are suitable candidates for predicting stock return movements in the Finnish stock market. Two alternative sets of input features were analyzed, and the LSTMs were benchmarked against conventional recurrent networks and logistic regression classifiers. Performance of the proposed models were evaluated by producing weekly out-of-sample predictions for the period between 2006–2020, and by further utilizing these predictions to derive prediction-based investment portfolios. Moreover, the performance of LSTM networks within different subperiods were analyzed.

In terms of classification accuracy, the LSTMs delivered robust performance and clearly outperformed all benchmark methods even after running the networks multiple times. The best performing LSTM model utilized a set of nine input features, including lagged returns, four stock-specific features and four general features. The additional input features were not found to be statistically significant in terms of predictability, but they were shown to provide additional value in an economic sense. The multivariate LSTM model delivered an annual return of 12.7% and a Sharpe ratio of 0.459 before transaction costs, also providing the most desirable risk-return profile among all portfolios. A univariate LSTM model utilizing only lagged returns as inputs delivered an annual return of 11.2% and a Sharpe ratio of 0.441, while a buy-and-hold portfolio yielded an 8.6% annual return and a Sharpe ratio of 0.338 within the same sample period.

Both LSTM models were able to maintain their edge during the full sample even after implementing transaction costs, but a subperiod analysis revealed that the profitability mostly attributes to the first half of the sample. However, the performance of the multivariate LSTM network stood out from other models during the period between 2018–2020, since the model managed to avoid investing in steeply declining stocks.

By unveiling some common characteristics among the stocks selected for trading, the LSTMs were found to independently extract similar patterns to well-known capital market anomalies of short-term mean reversion and momentum. A mean-reversion portfolio constructed based on these findings achieved nearly similar profitability prior to transaction costs but had a less satisfactory risk-return profile. Therefore, the high-level performance of LSTM models could be partially, but not completely explained by short-term reversal effects.

The results show that the proposed LSTMs are able to recognize profitable price patterns from the data, and these patterns cannot be unequivocally attributed to any specific characteristics. This would indicate that the stock returns are partly driven by longer-term signals, and that the proposed LSTM networks can extract this type of subtle information from noisy stock market data. These finding also posit a challenge to the theory of market efficiency in Finland, as there are periods within the sample when the market exhibits a higher level of predictability.

Despite incorporating high computational costs and abundant complexity, LSTM network was shown to be a suitable method for stock return movement prediction task. Even though the theoretical performance might not fully materialize if implemented in practice, LSTMs certainly have predictive properties that make them useful tools and complements for different investment strategies.

In terms of the investment strategies, especially in the current market turmoil it could be beneficial to introduce a probability threshold in order to avoid investing in a declining market. Instead of predicting which stocks outperform the cross-sectional median, one could make predictions about the actual sign of the movements, or alternately use a combination of these methods. Since the field of machine learning and its applications on finance is developing rapidly, more advanced models and their combinations could be considered in future research.

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. – Levenberg, J. – Mané, D. – Monga, R. – Moore, S. – Murray, D. – Olah, C. – Schuster, M. – Shlens, J. – Steiner, B. – Sutskever, I. – Talwar, K. – Tucker, P. – Vanhoucke, V. – Vasudevan, V. – Viégas, F. – Vinyals, O. – Warden, P. – Wattenberg, M. – Wicke, M. – Yu, Y. – Zheng, X. (2015) TensorFlow: Large-scale machine learning on heterogeneous systems. <http://tensorflow.org/>, retrieved 22.1.2020.

Atsalakis, G. S. – Valavanis, K. P. (2009) Surveying Stock Market Forecasting Techniques – Part II: Soft Computing Methods. *Expert Systems with Applications*, Vol. 36 (3), 5932–5941.

Bahrammirzaee, A. (2010) A comparative survey of artificial intelligence applications in finance: Artificial neural networks, expert system and hybrid intelligent systems. *Neural Computing and Applications*, Vol. 19 (8), 1165–1195.

Bao, W. – Yue, J. – Rao, Y. (2017) A Deep Learning Framework for Financial Time Series Using Stacked Autoencoders and Long-Short Term memory. *PLoS ONE*, Vol. 12 (7).

Bengio, Y. (2012) Practical Recommendations for Gradient-Based Training of Deep Architectures. In: Montavon, G. – Orr, G.B. – Müller, K.-R. (Eds.) *Neural Networks: Tricks of the Trade.* 2nd Ed. Springer-Verlag, Heidelberg, Berlin.

Blume, L. – Easley, D. – O'Hara, M. (1994) Market statistics and technical analysis: The role of volume. *The Journal of Finance,* Vol. 49 (1), 153–181.

Brennan, M. J. – Chordia, T. – Subrahmanyam, A. (1998) Alternative Factor Specifications, Security Characteristics, and the Cross-Section of Expected Stock Returns. *Journal of Financial Economics*, Vol. 49 (3), 345–373.

Brooks, C. (2014) *Introductory Econometrics for Finance*. 3rd Ed. Cambridge University Press, Cambridge.

Brownlee, J. (2018) *Long Short-term Memory Networks with Python: Develop Sequence Prediction Models with Deep Learning*. <https://machinelearningmastery.com/lstms-with-python/>, retrieved 3.2.2020.

Chen, A.-S. – Leung, M. – Daouk, H. (2003) Application of neural networks to an emerging financial market: forecasting and trading the Taiwan Stock Index. *Computers & Operations Research*, Vol. 30, 901–923.

Chollet, F. (2015) *Keras*. <https://keras.io/>, retrieved 22.1.2020.

Chollet, F. (2018) *Deep Learning with Python*. Manning, Shelter Island, NY.

Clements, M. P. – Franses, P. H. – Swanson, N. R. (2004) Forecasting Economic and Financial Time-Series with Non-Linear Models. *International Journal of Forecasting*, Vol. 20 (2), 169–183.

Daniel, K. – Moskowitz, T. J. (2016) Momentum Crashes. *Journal of Financial Economics*, Vol. 122 (2), 221–247.

De Gooijer, J. G. – Kumar, K. (1992) Some Recent Developments in Non-Linear Time Series Modelling, Testing, and Forecasting. *International Journal of Forecasting*, Vol. 8 (2), 135–156.

Diebold, F. X. – Mariano, R. S. (1995) Comparing predictive accuracy. *Journal of Business & Economic Statistics*, Vol. 13 (3), 253–263.

Er, Ş. – Vuran, B. (2013) Factors Affecting Stock Returns of Firms Quoted in ISE Market: A Dynamic Panel Data Approach. *International Journal of Business and Social Research*, Vol. 2 (1), 108–121.

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

Farmer, L. – Schmidt, L. – Timmermann, A. (2019) Pockets of predictability. *CEPR Discussion Paper,* No. DP12885.

Fischer, T. – Krauss, C. (2018) Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, Vol. 270 (2), 654–669.

Gers, F. A. – Schmidhuber, J. – Cummins F. (2000) Learning to Forget: Continual Prediction with LSTM. *Neural Computation*, Vol. 12 (10), 2451–2471.

Goodfellow, I. – Bengio, Y. – Courville, A. (2016) *Deep Learning*. MIT Press, <http://www.deeplearningbook.org/>, retrieved 18.1.2020.

Graves, A. – Mohamed, A. – Hinton, G. (2013) Speech recognition with deep recurrent neural networks. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 6645–6649.

Grossman, S. – Stiglitz, J. (1980) On the impossibility of informationally efficient markets. *The American Economic Review*, Vol. 70 (3), 393–408.

Gu, S. – Kelly, B. – Xiu, D. (2018) Empirical Asset Pricing via Machine Learning. *IDEAS Working Paper Series from RePEc*.

Haykin, S. (2009) *Neural Networks and Learning Machines.* 3rd Ed. Pearson education Inc, New Jersey.

Hinton, G. (2012) Neural networks for machine learning: Lecture 6e.

Hochreiter, S. – Schmidhuber, J. (1997) Long Short-Term Memory. *Neural Computation*, Vol. 9 (8), 1735–1780.

Hong, H. – Lim, T. – Stein, J. C. (2000) Bad News Travels Slowly: Size, Analyst Coverage, and the Profitability of Momentum Strategies. *The Journal of Finance*, Vol. 55 (1), 265–295.

James, G. – Witten, D. – Hastie, T. – Tibshirani, R. (2013) *As Introduction to Statistical Learning: with Applications in R*. Springer-Verlag, New York.

Jegadeesh, N. − Titman, S. (1993) Returns to buying winners and selling losers: Implications for stock market efficiency. *The Journal of Finance*, Vol. 48 (1), 65−91.

Kara, Y. – Acar Boyacioglu, M. – Baykan, Ö. K. (2011) Predicting Direction of Stock Price Index Movement Using Artificial Neural Networks and Support Vector Machines: The Sample of the Istanbul Stock Exchange. *Expert Systems with Applications*, Vol. 38 (5), 5311–5319.

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*, Vol. 259 (2), 689–702.

Leung, M. T. – Daouk, H. – Chen, A.-S. (2000) Forecasting Stock Indices: A Comparison of Classification and Level Estimation Models. *International Journal of Forecasting*, Vol. 16 (2), 173–190.

Long, W. – Lu, Z. – Cui, L. (2019) Deep Learning-Based Feature Engineering for Stock Price Movement Prediction. *Knowledge-Based Systems*, Vol. 164, 163–173.

López de Prado, M. M. (2018) *Advances in Financial Machine Learning*. John Wiley & Sons, Hoboken, New Jersey.

Lütkepohl, H. (2005) *New Introduction to Multiple Time Series Analysis*. Springer-Verlag, Heidelberg, Berlin.

Makridakis, S. – Spiliotis, E. – Assimakopoulos, V. – Hernandez, M. – Alejandro, R. (2018) Statistical and Machine Learning Forecasting Methods: Concerns and Ways Forward. *PloS ONE*, Vol. 13 (3).

Malkiel, B. G. (2003) The efficient market hypothesis and its critics. *Journal of Economic Perspectives*, Vol. 17 (1), 59−82.

Murphy, Kevin P. (2012) *Machine Learning: A Probabilistic Perspective*. The MIT Press, Cambridge.

Neely, C. J. – Rapach, D. E. – Tu, J. – Zhou, G. (2014) Forecasting the Equity Risk Premium: The Role of Technical Indicators. *Management Science*, Vol. 60 (7), 1772–1791.

Nelson, D. M. Q. – Pereira, Adriano C. M. – de Oliveira, Renato A. (2017) Stock Market's Price Movement Prediction with LSTM Neural Networks. *2017 International Joint Conference on Neural Networks (IJCNN)*, 1419–1426.

Niaki, S. – Hoseinzade, S. (2013) Forecasting S&P 500 Index Using Artificial Neural Networks and Design of Experiments. *Journal of Industrial Engineering International*, Vol. 9 (1), 1–9.

Pedregosa, F. – Varoquaux, G. – Gramfort, A. – Michel, V. – Thirion, B. – Grisel, O. – Blondel, M. – Prettenhofer, P. – Weiss, R. – Dubourg, V. – Vanderplas, J. – Passos, A. – Cournapeau, D. – Brucher, M. – Perrot, M. – Duchesnay, E. (2011) Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, Vol. 12, 2825–2830.

Poterba, J. – Summers, L. (1988) Mean Reversion in Stock Prices: Evidence and Implications. *Journal of Financial Economics*, Vol. 22 (1), 27–59.

Qi, M. − Maddala, G. S. (1999) Economic factors and the stock market: A new perspective. *Journal of Forecasting*, Vol. 18 (3), 151−166.

Rozenberg, G. – Bäck, T. – Kok, J. N. (2012) *Handbook of Natural Computing*. Springer-Verlag, Heidelberg, Berlin.

Srivastava, N. – Hinton, G. – Krizhevsky, A. – Sutskever, I. – Salakhutdinov, R. (2014) Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, Vol. 15, 1929–1958.

Tsay, R. S. (2005) *Analysis of Financial Time Series.* 2nd Ed. John Wiley & Sons, Hoboken, New Jersey.

Tsay, R. S. – Chen, R. (2019) *Nonlinear Time Series Analysis*. John Wiley & Sons, Hoboken, New Jersey.

Varian, H. R. (2014) Big Data: New Tricks for Econometrics. *The Journal of Economic Perspectives*, Vol. 28 (2), 3–28.

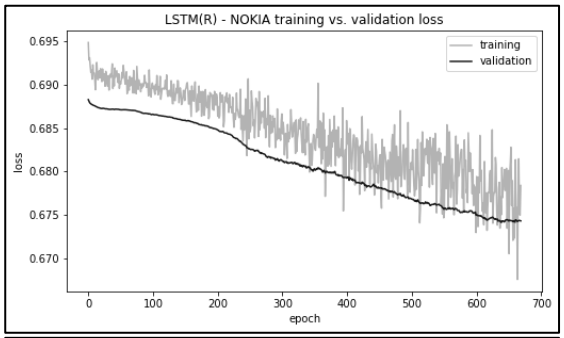
Welch, I. – Goyal, A. (2008) Comprehensive Look at The Empirical Performance of Equity Premium Prediction. *The Review of Financial Studies*, Vol 21. (4), 1455–1508.

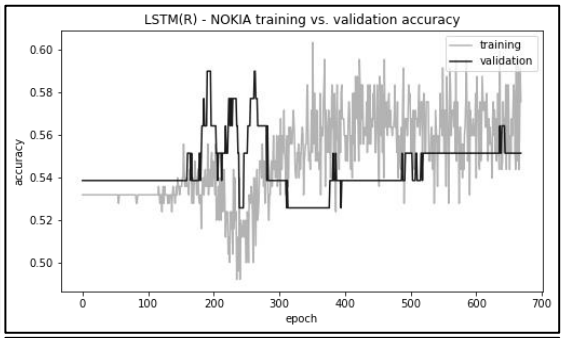
White, H. (1988) Economic prediction using neural networks: the case of IBM daily stock returns. *IEEE International Conference on Neural Networks*, Vol. 2, 451–458.

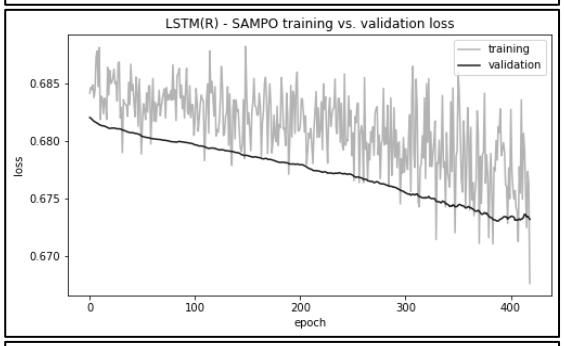
Zhong, X. – Enke, D. (2017) Forecasting Daily Stock Market Return Using Dimensionality Reduction. *Expert Systems with Applications*. Vol. 67 (1), 126–139.

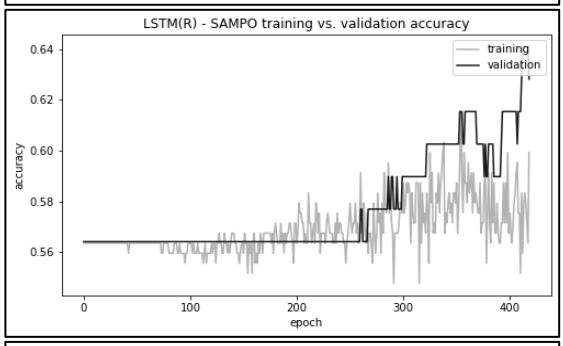
APPENDICES

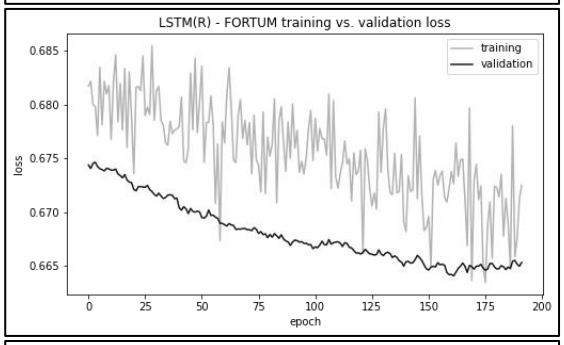
Appendix 1. LSTM (R) training history of the final study period

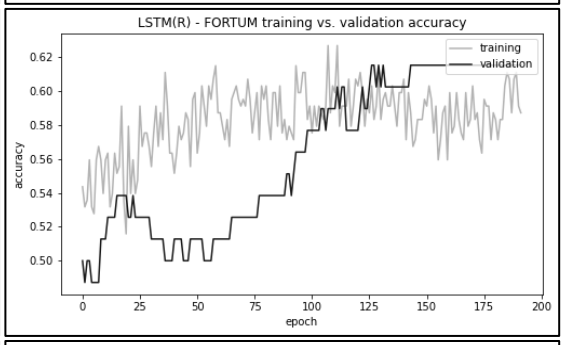


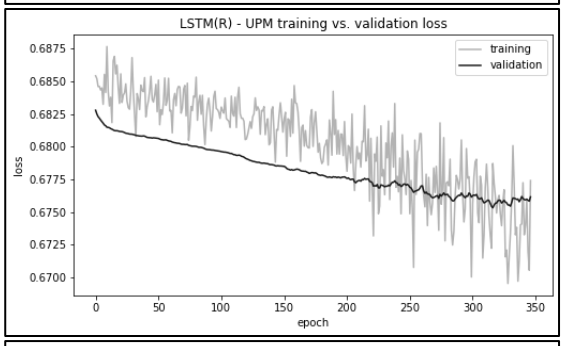


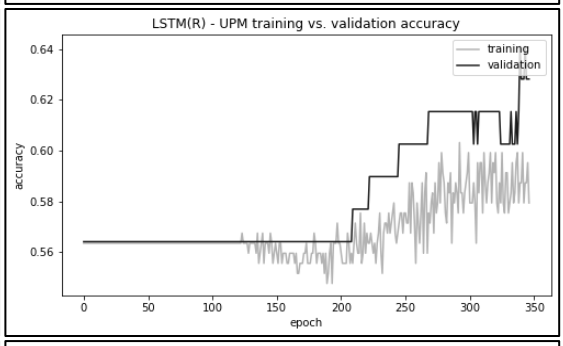


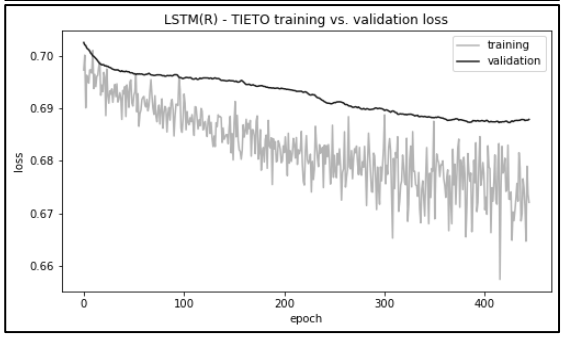


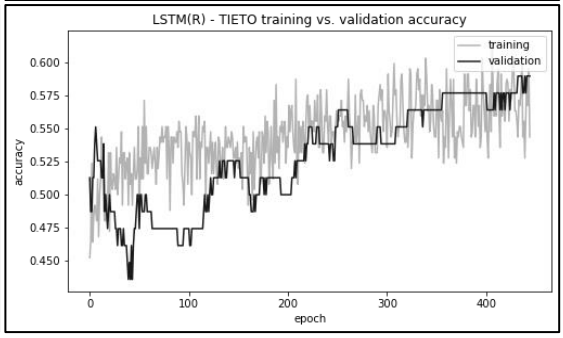


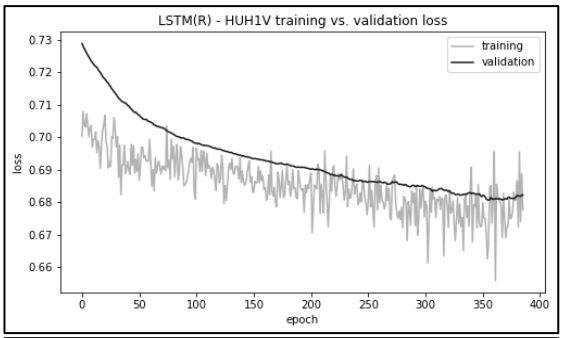


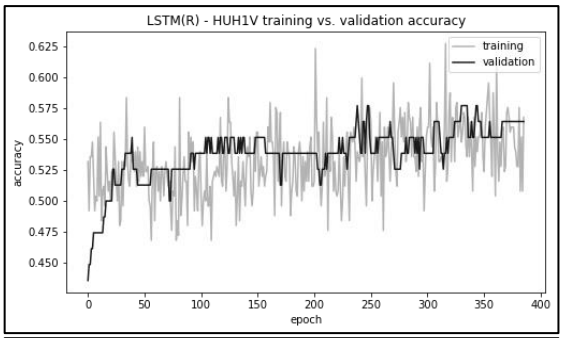


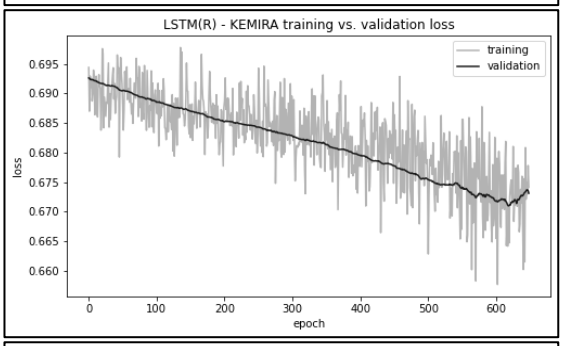


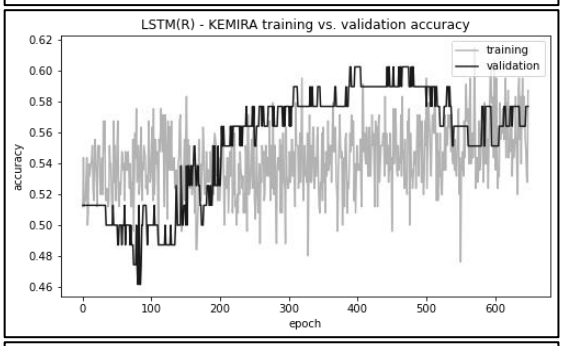


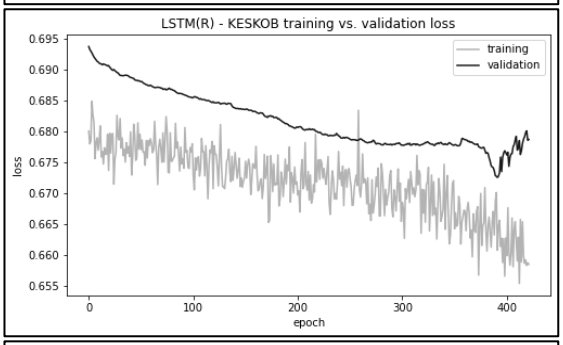


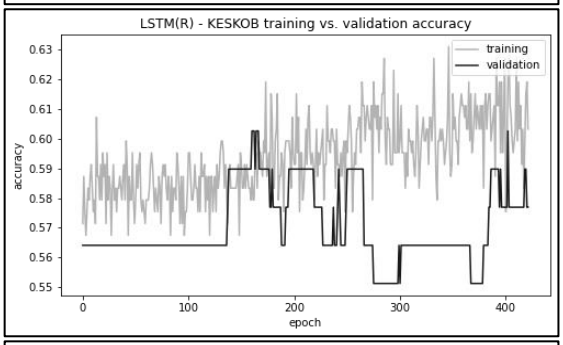

LSTM(R) - HUH1V training vs. validation loss
training
validation
loss
epoch

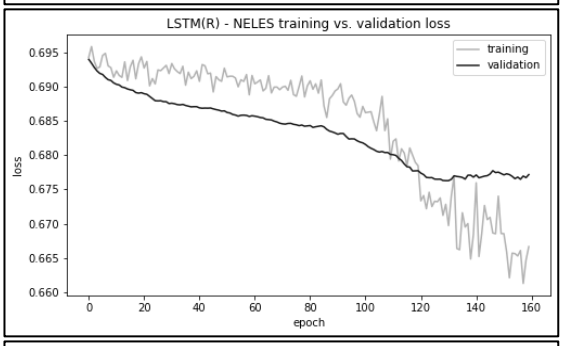

LSTM(R) - HUH1V training vs. validation accuracy
training
validation
accuracy
epoch

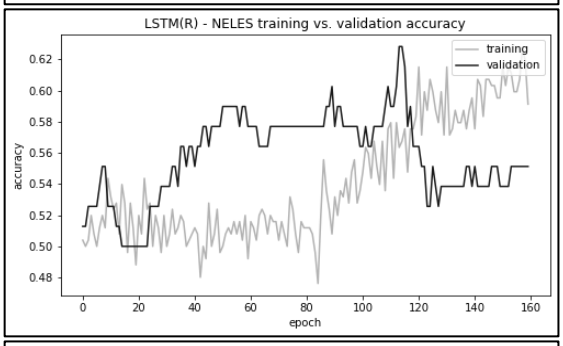

LSTM(R) - KEMIRA training vs. validation loss
training
validation
loss
epoch

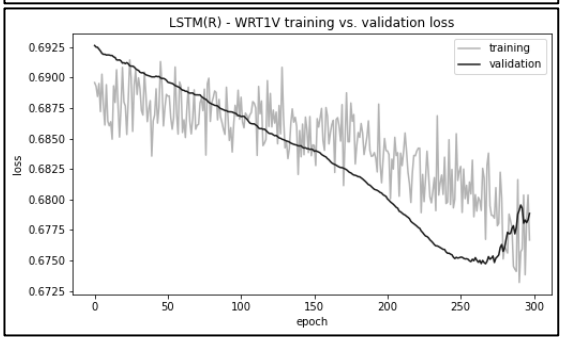

LSTM(R) - KEMIRA training vs. validation accuracy
training
validation
accuracy
epoch

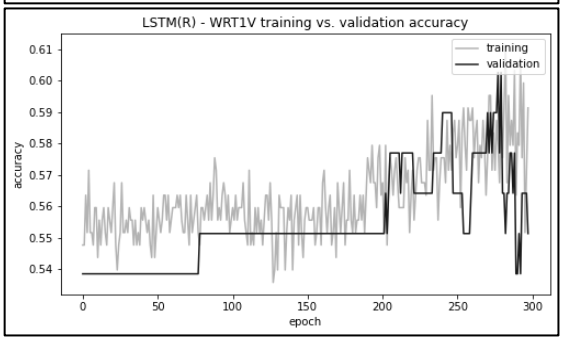

LSTM(R) - KESKOB training vs. validation loss
training
validation
loss
epoch

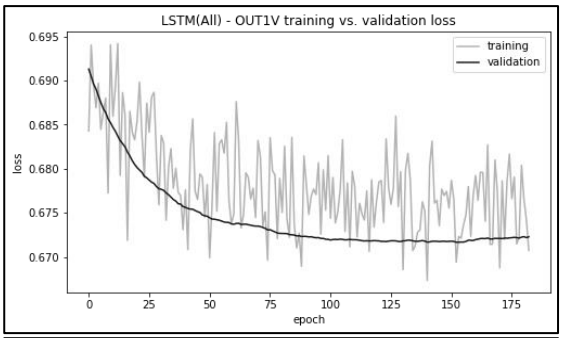

LSTM(R) - KESKOB training vs. validation accuracy
training
validation
accuracy
epoch

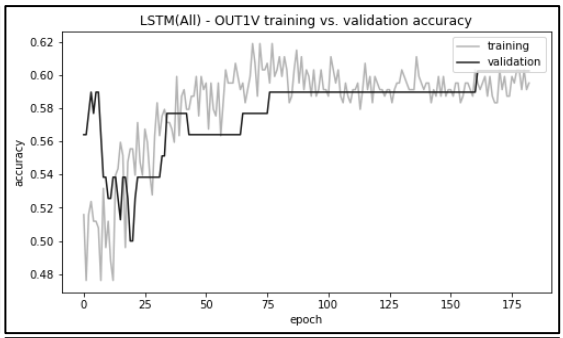

LSTM(R) - NELES training vs. validation loss
training
validation
loss
epoch

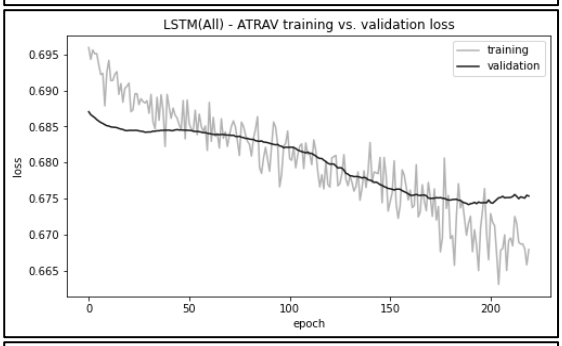

LSTM(R) - NELES training vs. validation accuracy
training
validation
accuracy
epoch

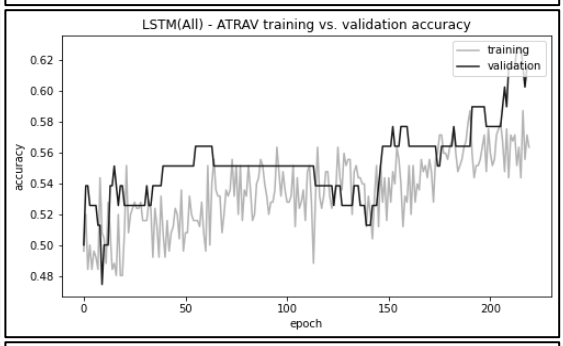

LSTM(R) - WRT1V training vs. validation loss
training
validation
loss
epoch

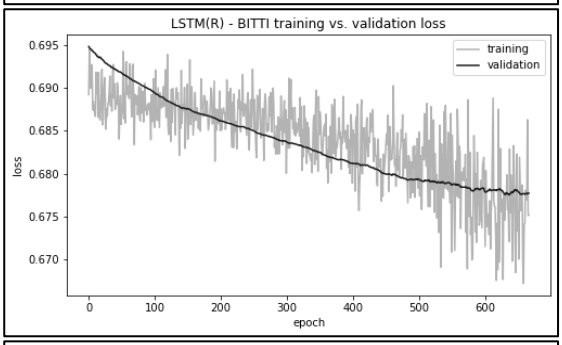

LSTM(R) - WRT1V training vs. validation accuracy
training
validation
accuracy
epoch

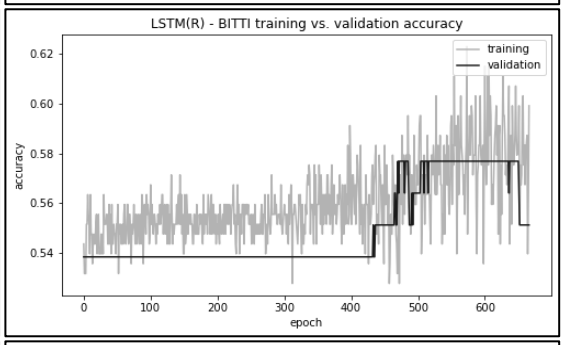

LSTM(All) - OUT1V training vs. validation loss
training
validation
loss
epoch

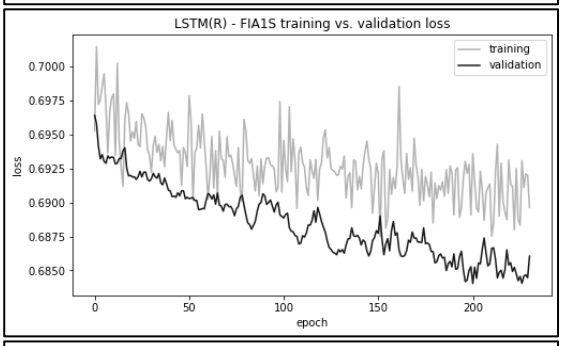

LSTM(All) - OUT1V training vs. validation accuracy
training
validation
accuracy
epoch

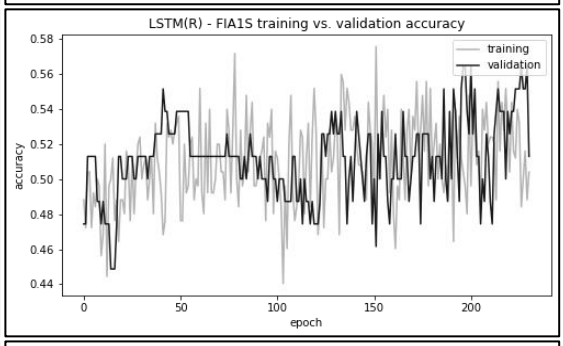

LSTM(All) - ATRAV training vs. validation loss
training
validation
loss
epoch

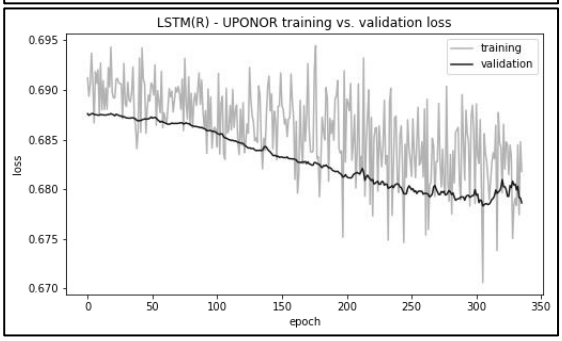

LSTM(All) - ATRAV training vs. validation accuracy
training
validation
accuracy
epoch

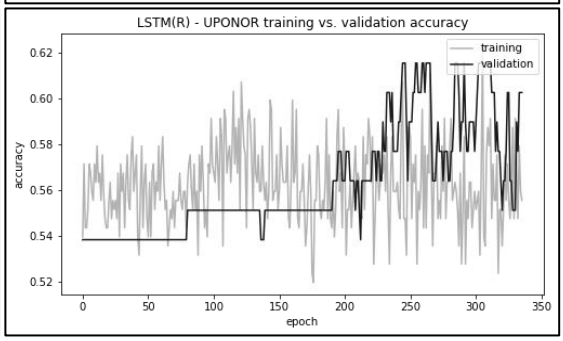

LSTM(R) - BITTI training vs. validation loss
training
validation
loss
epoch


LSTM(R) - BITTI training vs. validation accuracy
training
validation
accuracy
epoch

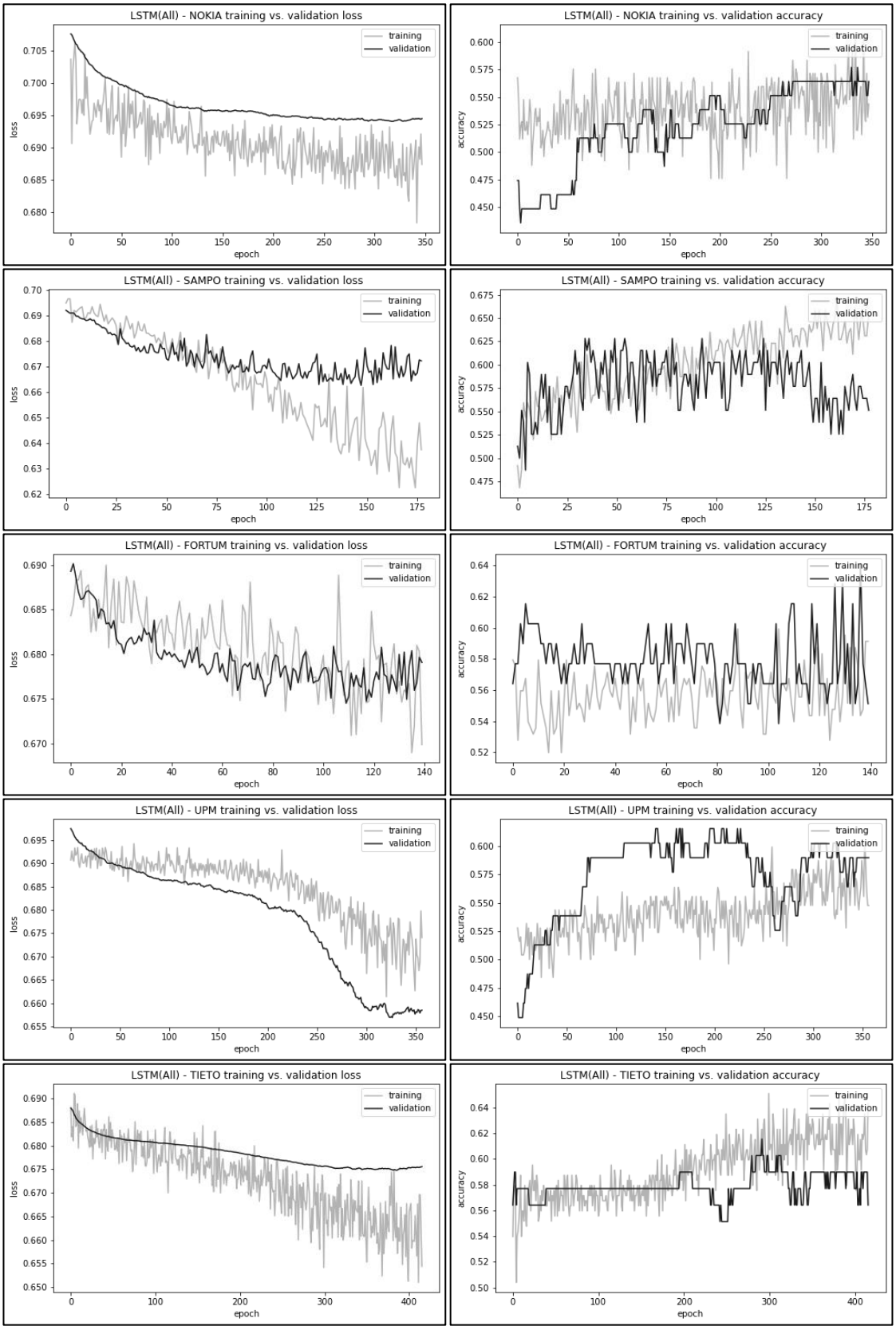

LSTM(R) - FIA1S training vs. validation loss
training
validation
loss
epoch

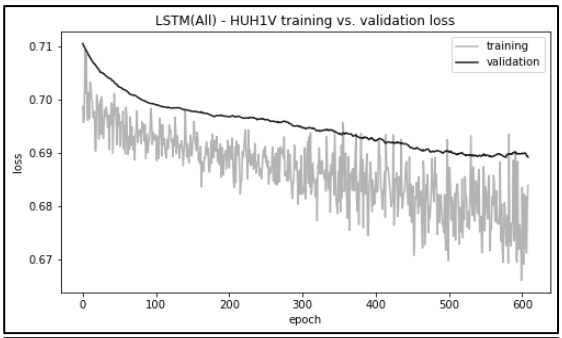

LSTM(R) - FIA1S training vs. validation accuracy
training
validation
accuracy
epoch

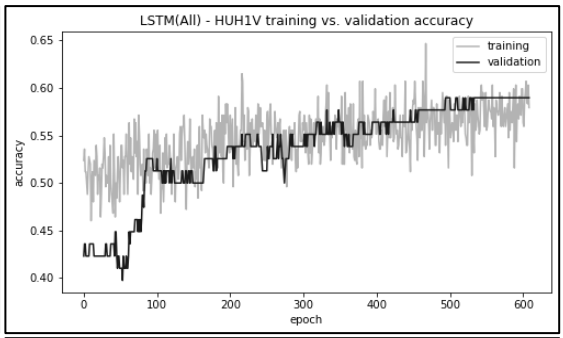

LSTM(R) - UPONOR training vs. validation loss
training
validation
loss
epoch

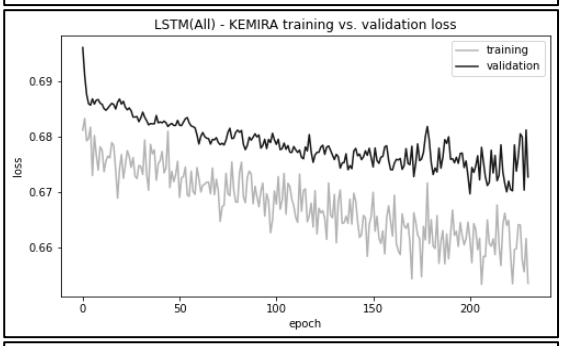

LSTM(R) - UPONOR training vs. validation accuracy
training
validation
accuracy
epoch

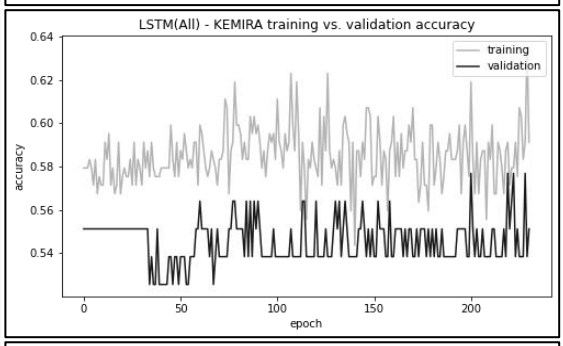
Appendix 2. LSTM (All) training history of the final study period

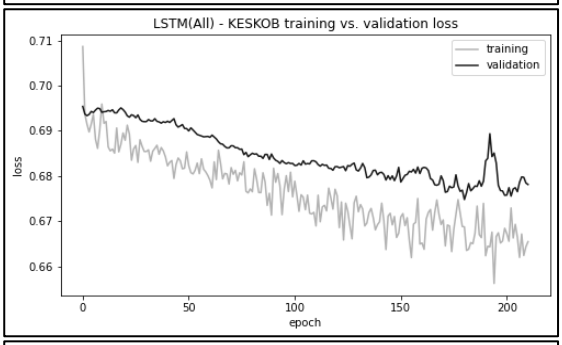


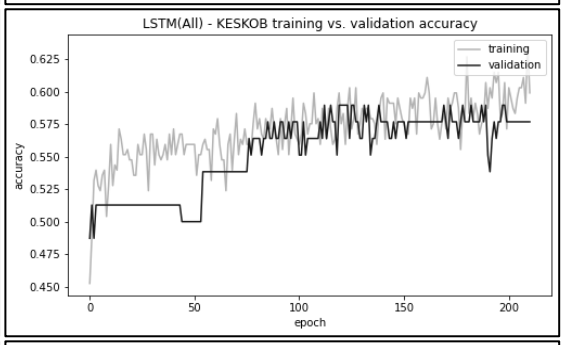

LSTM(All) - HUH1V training vs. validation loss
training
validation
loss
epoch

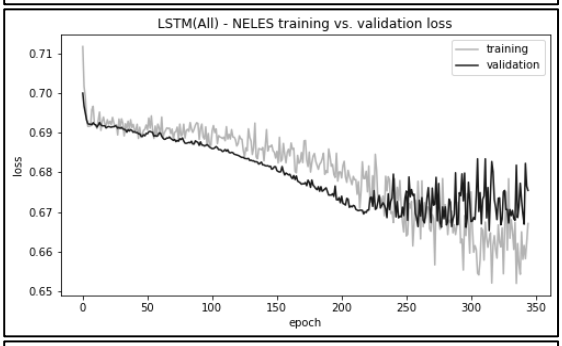

LSTM(All) - HUH1V training vs. validation accuracy
training
validation
accuracy
epoch

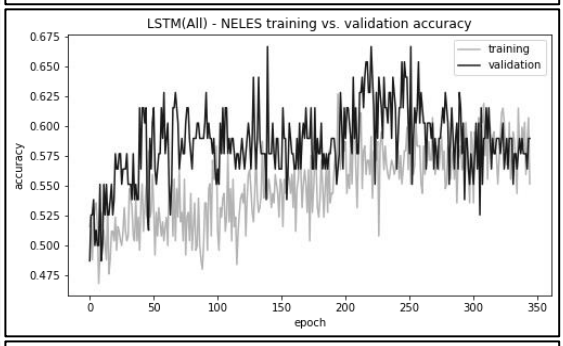

LSTM(All) - KEMIRA training vs. validation loss
training
validation
loss
epoch

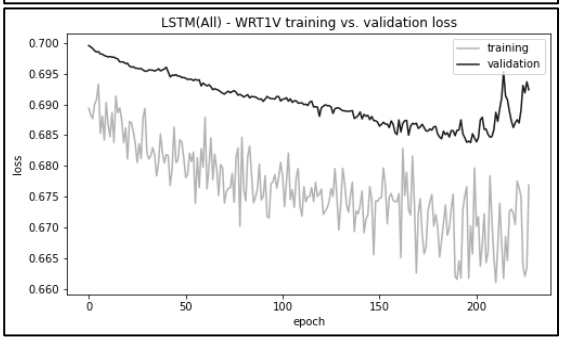

LSTM(All) - KEMIRA training vs. validation accuracy
training
validation
accuracy
epoch

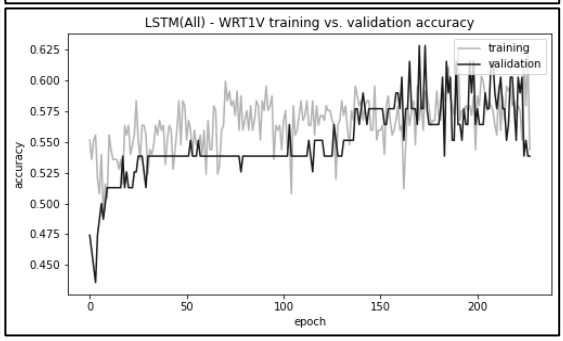

LSTM(All) - KESKOB training vs. validation loss
training
validation
loss
epoch

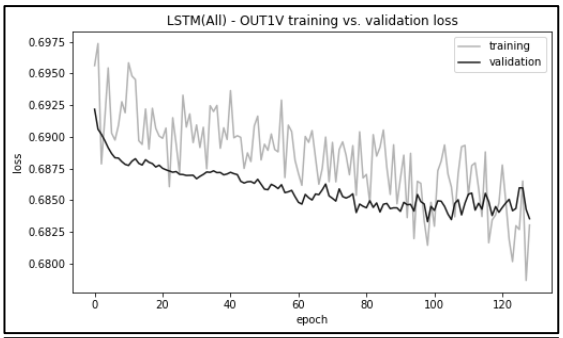

LSTM(All) - KESKOB training vs. validation accuracy
training
validation
accuracy
epoch

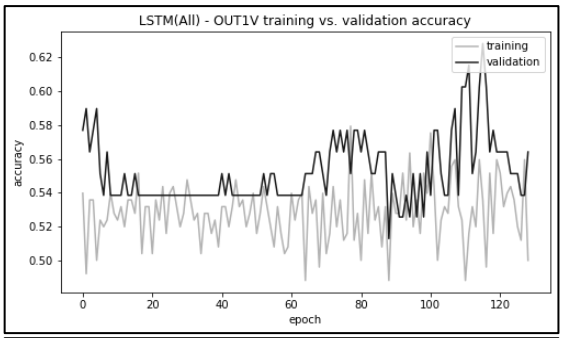

LSTM(All) - NELES training vs. validation loss
training
validation
loss
epoch

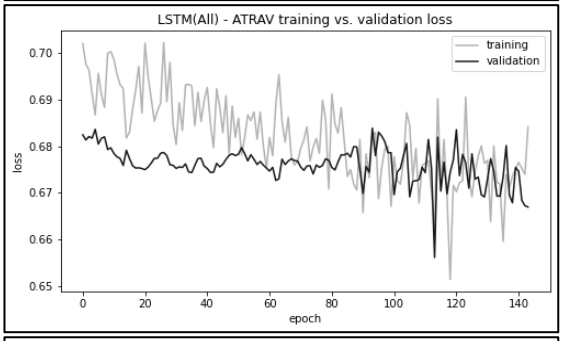

LSTM(All) - NELES training vs. validation accuracy
training
validation
accuracy
epoch

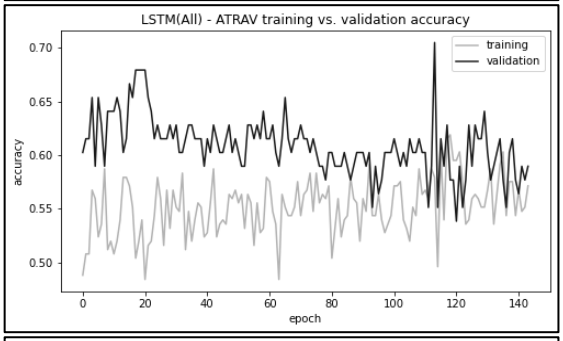

LSTM(All) - WRT1V training vs. validation loss
training
validation
loss
epoch

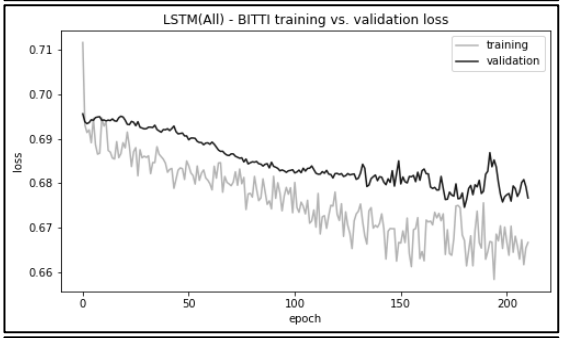

LSTM(All) - WRT1V training vs. validation accuracy
training
validation
accuracy
epoch

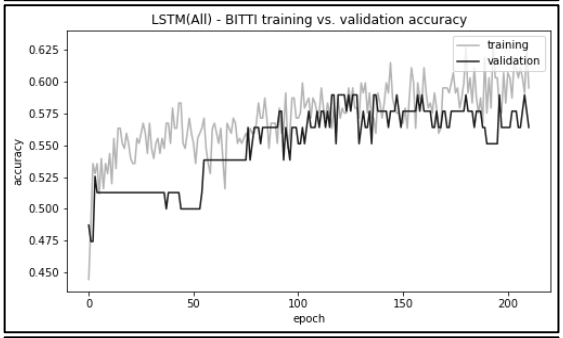

LSTM(All) - OUT1V training vs. validation loss
training
validation
loss
epoch

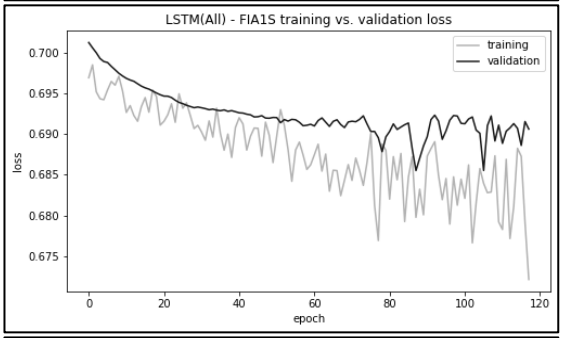

LSTM(All) - OUT1V training vs. validation accuracy
training
validation
accuracy
epoch

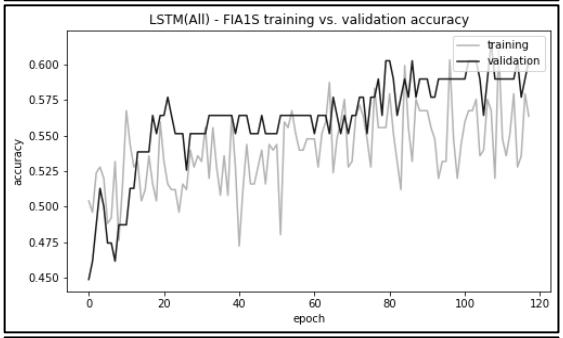

LSTM(All) - ATRAV training vs. validation loss
training
validation
loss
epoch

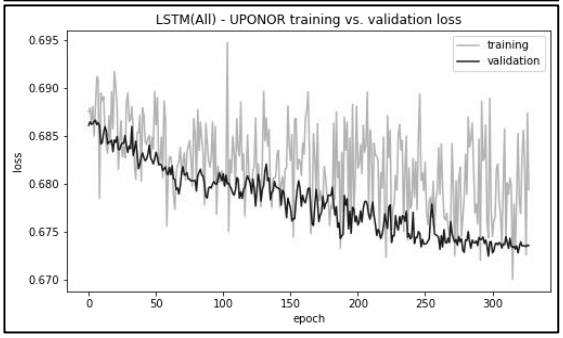

LSTM(All) - ATRAV training vs. validation accuracy
training
validation
accuracy
epoch

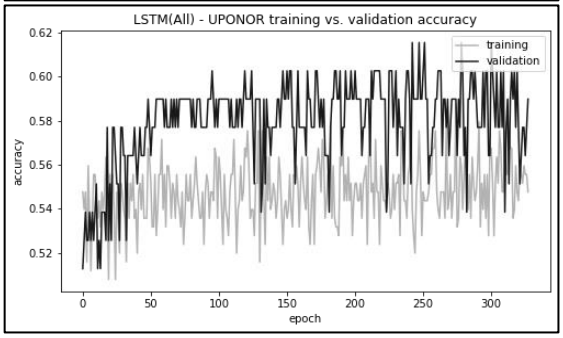

LSTM(All) - BITTI training vs. validation loss
training
validation
loss
epoch


LSTM(All) - BITTI training vs. validation accuracy
training
validation
accuracy
epoch


LSTM(All) - FIA1S training vs. validation loss
training
validation
loss
epoch


LSTM(All) - FIA1S training vs. validation accuracy
training
validation
accuracy
epoch


LSTM(All) - UPONOR training vs. validation loss
training
validation
loss
epoch


LSTM(All) - UPONOR training vs. validation accuracy
training
validation
accuracy
epoch