On Comparison of ARIMA and RNN Models in Predicting Stock Indices

Matti Staudinger

School of Science

Bachelor's thesis

Espoo 2.12.2021

Supervisor

Assoc. Prof. Pauliina Ilmonen

Advisor

Assoc. Prof. Pauliina Ilmonen

Aalto University
School of Science

Copyright © 2022 Matti Staudinger

The document can be stored and made available to the public on the open internet pages of Aalto University.
All other rights are reserved.

Aalto University
School of Science

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the bachelor's thesis

Author Matti Staudinger

Title On Comparison of ARIMA and RNN Models in Predicting Stock Indices

Degree programme Bachelor's Programme in Science and Technology

Major Mathematics and Systems Sciences **Code of major** SCI3029

Teacher in charge Assoc. Prof. Pauliina Ilmonen

Advisor Assoc. Prof. Pauliina Ilmonen

Date 2.12.2021 **Number of pages** 26 **Language** English

Abstract

The behavior of stock indices is difficult to predict because of their erratic nature. When predicting the value of stock indices, it is possible to try to take many variables into account, such as political or economical changes or the previous values of the index itself.

The purpose of this thesis is to compare autoregressive integrated moving average (ARIMA) models and recurrent neural networks (RNN), when predicting S&P 500 and OMXH25 stock indices in a reasonably stable time period. Short-term forecasts of the indices are produced and compared. Only the information about the index close values are used when making the predictive models. The different predictions are then compared and their use as a supportive tool is assessed when making an investment decision.

For an investor an ability to make good predictions would result in monetary gains and resulting mathematical models can be used in automated trading algorithms. It is also beneficial to understand what kind of methods are suitable for stock index prediction.

Based on the results of this thesis, the ARIMA and RNN models produce similar predictions. Neither can be said to be better than the other, when predicting the development of a stock index. Neither method was able to make good predictions as the resulting average errors were near or over the average daily fluctuations of the index. Only some one day predictions achieved better results and the largest average errors were too large for the models to be used as a reliable decision making tool for investments. None of the forecasts were unable to anticipate future changes of indices.

Keywords Time series, ARIMA model, Recurrent Neural Network, Stock index

Aalto University
School of Science

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi
Tekniikan kandidaatintyön tiivistelmä

Tekijä Matti Staudinger 58344S

Työn nimi ARIMA- ja neuroverkkomallien vertailua osakeindeksejä ennustettaessa

Koulutusohjelma Teknistieteellinen kandidaattiohjelma

Pääaine Matematiikka ja systeemitieteet **Pääaineen koodi** SCI3029

Vastuuopettaja Prof. Pauliina Ilmonen

Työn ohjaaja Prof. Pauliina Ilmonen

Päivämäärä 2.12.2021 **Sivumäärä** 26 **Kieli** Englanti

Tiivistelmä

Osakeindeksien käyttäytymistä on vaikea ennustaa, koska ne vaihtelevat epäsäännöllisesti. Kun indeksin arvoa ennustetaan, voidaan yrittää ottaa huomioon monien muuttujien vaikutuksia, kuten politiikan ja talouden muutoksia sekä indeksin omia aiempia arvoja.

Tämän kandidaatintutkielman tavoitteena on kehittää ja vertailla perinteisiä tilastollisia autoregressiivisiä integroituja liukuvakeskiarvoisia (ARIMA) malleja ja takaisinkytkettyihin neuroverkkoihin (RNN) perustuvia malleja ennustettaessa S&P 500- ja OMXH25-indeksejä kohtalaisen vakaalla ajanjaksolla. Osakeindeksien lyhyen aikavälin ennusteita ja ennustemalleja tehtäessä käytetään tietoa vain indeksin omista aiemmista päätösarvoista. Tämän jälkeen eri ennusteita vertaillaan toisiinsa. Lopuksi arvioidaan niiden hyödyntämismahdollisuuksia sijoituspäätöstä tehtäessä.

Sijoittajille on merkittävää taloudellista hyötyä kyvystä ennakoida indeksin kehitystä, ja ennustamisen yhteydessä syntyviä matemaattisia malleja voidaan käyttää automatisoiduissa kaupankäyntialgoritmeissa. On myös hyödyllistä ymmärtää, millaiset menetelmät sopivat osakeindeksien ennustamiseen.

Tulosten perusteella ARIMA- ja RNN-mallit tuottavat samankaltaisia ennusteita. Kumpaankaan tapaan perustuvan mallin ei voida sanoa olevan parempi osakeindeksin kehitystä ennakoitaessa. Kumpikaan menetelmä ei kyennyt tekemään hyviä ennusteita, vaan suurin osa keskimääräisistä virheistä oli yli indeksin keskimääräisen päivävaihtelun. Vain osa yhden päivän ennusteista tuotti parempia tuloksia, ja osassa mallien tuottamat virheet olivat liian suuria, jotta niitä voisi käyttää luotettavana työkaluna investointipäätöksiä tehtäessä. Mikään ennusteista ei myöskään kyennyt ennakoimaan indeksien tulevia muutoksia.

Avainsanat Aikasarjat, ARIMA-malli, Takaisinkytketty neuroverkko, Osakeindeksi

Contents

1 Introduction

Values of stock indices are predicted using a variety of different methods, because there are significant financial benefits for investors from the ability to anticipate the development of a stock index. Another motivation for stock index prediction is forecasting the future state of the economy, as the changes in stock indices are usually followed by changes in the economy (Bosworth et al., 1975).

The methods of stock index prediction are divided into technical analysis, which is based on the past values of the index and fundamental analysis, which in turn looks at the financial information and future prospects of the companies included in the index, such as turnover, profitability and order backlog. Forecasting models can be utilized in automated trading and for example, software robots that are able to quickly utilize vast amounts of data are controlled by mathematical models.

Family of autoregressive integrated moving average (ARIMA) models are used in the statistical analysis of time series. ARIMA models can be used for describing the behavior of time series or for predicting its future values. Similar predictions can be done with recurrent neural networks (RNN), that are a type of artificial neural network (ANN). ANNs are inspired by biological neural networks and they consist of several layered neurons or computation nodes that use series subsequent computations to produce output values from input values. The models used for computations are learned from examples.

This thesis compares traditional statistical ARIMA models and RNN models on predicting stock indices when their movements are reasonably stable, so for example stock market crashes are not considered. Only short term predictions are made and no external explanatory variables will be used, meaning that only the previous close values of the index itself were used for predicting. The Standard& Poor's 500 (S&P 500) index, which consists of the 500 largest companies in the United States, and the OMX Helsinki 25 (OMXH25) index, which consists of the 25 largest companies on the Helsinki Stock Exchange, were chosen as indices to forecast.

The background of ARIMA and RNN models and previous research about stock indices will be discussed in chapter 2. Datasets, methods, mathematics used in ARIMA and RNN models and settings for analysis are presented in chapter 3. Chapter 4 consists of prediction results. Finally conclusions and future prospects are discussed in chapter 5.

2 Background

Autoregressive moving average (ARMA) processes are discrete time processes that are used when analyzing and predicting time series. There has been research in this area beginning with Yule (1927) describing concepts of autoregressive processes while studying the number of sunspots. Autoregressive integrated moving average models, which try to transform the non-stationary time series to stationary through differencing, are a generalization of the ARMA models. The processes are named seasonal autoregressive integrated moving average (SARIMA) processes when possible seasonality is taken into account (Brockwell and Davis, 2009; Box et al., 2015).

Also other methods can be used for making predictions. Some of them rely on artificial neural networks. ANNs are a subfield of machine learning, which in turn is a subfield of artificial intelligence. Artificial neural networks began with the mathematical modelling of human neurons (McCulloch and Pitts, 1943). One of the first applications of neural networks was in binary pattern recognition and prediction of the next bit (Widrow and Hoff, 1960). Recurrent neural networks and backprobagation have been developed by Rumelhart et al. (1986) and they are used for processing sequences of values, such as a time series of a stock index. Currently recurrent neural networks are often used in natural language processing and speech recognition (Goodfellow et al., 2016).

Stock indices measure the performance of the stocks of the market in question. Large cap indices, such as S&P 500 and OMXH25, consist of stocks of some or all of the largest companies in the corresponding stock exchange (S&P Global, 2021b; Nasdaq, Inc., 2020). Generally stock indices also measure the economic growth or decline, but that is not always the case (Bosworth et al., 1975).

There are different ways to construct an index. For example, in market capitalization weighted or market-cap-weighted indices larger companies have more impact on the value of the index. Float adjusted indices take the amount of shares publicly available into account when calculating the value. There can be additional constraints to the construction of indices such as individual companies cannot constitute for more than a certain amount of the index or companies must be selected from all business areas (S&P Global, 2021b; Nasdaq, Inc., 2020).

There have been many previous studies about stock price prediction. Ariyo et al. (2014) describe building ARIMA models for stock prediction and conclude that ARIMA models can be effective in short term stock price prediction. Mondal et al. (2014) write about the effectiveness of ARIMA models on stock price forecasting and find that ARIMA models achieve 85 % accuracy on predictions when comparing predicted prices to actual ones. And Selvin et al. (2017) compare different neural network models on predicting stock prices. They have come to the conclusion that deep learning models, such as RNNs, are good at identifying underlying patterns in time series of stocks.

3 Datasets and methods

This section discusses Standard and Poor's 500 and OMX Helsinki 25 indices, how they are constructed and what information can be spotted by inspecting the series. Then the methods used for building models for computing from 1 to 5 step predictions are described and the section ends with the specification of analysis settings.

3.1 Standard and Poor's 500 index

S&P 500 index includes 500 large companies that are listed in one of the U.S. primary stock exchanges that include the New York Stock Exchange (NYSE) and Nasdaq for example. Companies are picked in the index to represent 11 different sectors of industry. Generally the largest companies by their turnover are included, but selection is ultimately done by the Index committee. S&P 500 is a market capitalization weighted index, meaning that larger companies have more impact on the index. Each company's weight is calculated by market cap/total of all market caps, where market cap $=$ price $\cdot$ available shares. Weighting is done quarterly and index is reconstituted yearly meaning that without any extraordinary situation companies can be removed from the index only on that occasion (S&P Global, 2021b).

3.2 OMX Helsinki 25 index

OMXH25 index includes 25 largest companies by their turnover that are listed in the Nasdaq Helsinki stock exchange. Index measures the performance of these stocks and market capitalization weighting is used. Additionally each weight is constrained to a maximum of 10 %. Companies including to index and their weights are determined twice a year (Nasdaq, Inc., 2020).

3.3 Datasets

In this thesis forecast models are built for two reasonably stable time periods, meaning that major shocks in the economy are excluded. The aim is to find and compare models that predict future values of indices in a stable time period, not to predict economic turnarounds. Chosen periods are the year 2015 (1.1.2015 - 31.12.2015) and from early 2016 to the end of 2017 (1.3.2016 - 31.12.2017). In addition to these, models are built for the whole time series from 1.1.2015 to 31.12.2019.

Datasets have been preprocessed so that if any value is missing, due to a bank holiday or a saving error, the previous day's observation is used in place of the missing value. This ensures that there are no gaps in the time series and used tools work as planned.

Figure 1 shows the time series of closing values of the S&P 500 and OMXH25 indices from each trading day. Series have been scaled so that both indices begin from the

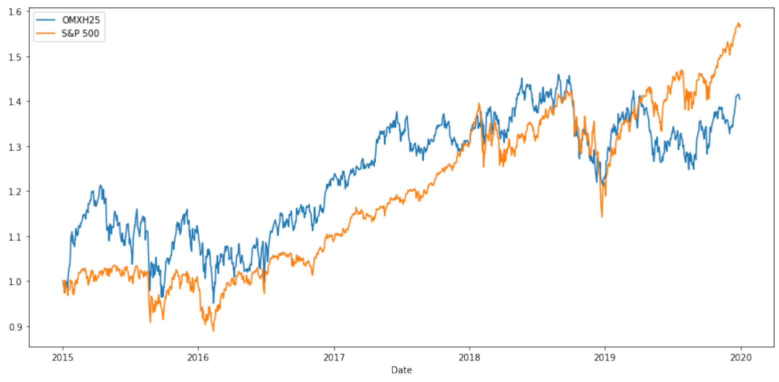

Figure 1: Time series of S&P 500 and OMXH25 indices from 2015 to 2019. Both time series are scaled to begin from 1 on 1.1.2015. There is a downward trend visible in the year 2015, an upward trend from early 2016 to the end of 2017 and a downward trend from mid 2018 to the end of 2018.

value of 1 at the beginning of 2015. The final value of the both time series is the last trading day of 2019. The data was downloaded from S&P Global (2021a) and Nasdaq, Inc. (2021) websites.

Both time series have strong visible changes around the imaginary trendline. In the year 2015 both indices have a slight downward trend, but the OMXH25 index has greater variance and a steep rise in the beginning of 2015. Then from early 2016 to the end of 2017 both indices are in a stable upward trend. OMXH25 gained value faster in mid 2016, early 2017 and mid 2017. There is a dip followed by a recovery in early 2018 and a strong downward trend is visible from mid 2018 to the beginning of 2019. This drop of indices seems to be similar for both. Then there is a strong upward trend in the early months of 2019 following with a drop. While the gain in the indices is similar, the drop is greater in the OMXH25 index. Rest of the year 2019 indices rise and the rise is steeper in the S&P 500 index.

To conclude there are three major changes in the trend for both indices: in the early 2016, mid 2018 and in the beginning of 2019. It seems possible that the OMXH25 index has greater variance than the S&P 500 index. There is no clear seasonality visible in the indices.

3.4 Seasonal autoregressive integrated moving average models

Autoregressive or AR models explain the future values of the variable in question using a linear combination of the past values. AR processes can be of different orders. The order defines how many past values are picked as explanatory variables. AR process of order p, AR(p) is defined

$$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + ... + \phi_p x_{t-p} + \epsilon_t, \tag{1}$$

where x_t is the t:th value in the time series, $\phi_1 ... \phi_p$ are the parameters of AR model and ϵ_t is the error of the model.

Moving average or MA models explain the future values of the variable in question using a linear combination of the past error terms. As with AR processes, MA processes can be of different orders and the order defines how many past error terms are selected as explanatory variables. MA process of order q, MA(q) is defined

$$x_t = \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + ... + \theta_q \epsilon_{t-q}, \tag{2}$$

where x_t is the t:th value in the time series, $\theta_1 ... \theta_q$ are the parameters of the MA model and ϵ_i is the i:th error of the model.

ARMA(p, q) process is a linear combination of AR(p)-, and MA(q)-processes and is defined

$$x_t = \sum_{i=1}^{p} \phi_i x_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-q} + \epsilon_t. \tag{3}$$

Only ARMA processes with certain parameter values are stationary (Box et al., 2015).

ARIMA(p, d, q) process represents possibly a non-stationary time series. This time series may be transformed into a stationary one by differencing it d times

$$z_t = x_t - x_{t-1}, \tag{4}$$

where z_i are the differences of two consecutive values. For example when removing a linear trend from the time series, it is differenced once. With a polynomial trend more differentiations are needed. Differencing cannot transform all non-stationary time series into stationary ones. If after the time series is differenced d times and a stationary time series is obtained, an ARMA(p, q) model can represent the transformed time series.

Seasonal ARMA or SARMA(P, Q)$_s$ is an ARMA process where dependencies between its values are s steps apart, where s is the length of season and P and Q are the order of the process

$$x_t = \sum_{i=1}^{P} \Phi_i x_{t-is} + \sum_{i=1}^{Q} \Theta_i \epsilon_{t-is} + \epsilon_t. \tag{5}$$

Seasonal differencing is required when taking into account seasonal changes in time series. Like differencing, this is sometimes needed to perform D times to produce stationary time series. Seasonal differencing transforms the original time series to a new one

$$z_t = x_t - x_{t-s}. \tag{6}$$

SARIMA(p, d, q)(P, D, Q)$_\mathrm{s}$ process combines all previous models.

$$x_t = \sum_{i=1}^{p} \phi_i x_{t-i} + \sum_{i=1}^{P} \Phi_i x_{t-is} + \sum_{i=1}^{q} \theta_i \epsilon_{t-q} + \sum_{i=1}^{Q} \Theta_i \epsilon_{t-Qs} + \epsilon_t \tag{7}$$

A SARIMA model gives interpretable results when the time series it describes is or can be transformed into a stationary one. A time series is stationary when it's expected value and variance don't depend on time. Variance also needs to be finite and the autocovariance of the time series needs to depend only on the difference between the values. This can be examined when looking at the plot of the time series. Stationary time series doesn't show any trend, seasonality or systematic changes in variance.

SARIMA models have several parts to be identified. First the time series needs to be stationary. If it is not, differencing or other methods are needed. If the time series can be transformed into a stationary by differencing d times, d determines the I-part of the model. Model might also require seasonal differencing. It is possible that the time series needs other types of transformations, such as logarithmic transformation, to produce stationarity or that the time series cannot be transformed into stationary one.

Figure 2 illustrates an example from S&P 500 from 1.3.2016 to 31.12.2017, where the original time series (upper part) shows a clear trend and is non-stationary. When it is differenced once, a stationary time series is produced (lower part). No more transformations are required and I-part is identified as $d = 1$.

If the time series is stationary, possibly after the differencing, the order of AR- and MA- parts are identified by inspecting the autocorrelation function (ACF) and partial autocorrelation function (PACF). If the time series is non-stationary, analysis of ACF and PACF should not be used.

ACF ρ of x_t at lag h is

$$\rho_x = \gamma_x(h)/\gamma_x(0) = \mathrm{Corr}(x_{t+h}, x_t), \tag{8}$$

where

$$\gamma_x(h) = \gamma_x(h, 0) = \mathrm{Cov}(x_{t+h}, x_t). \tag{9}$$

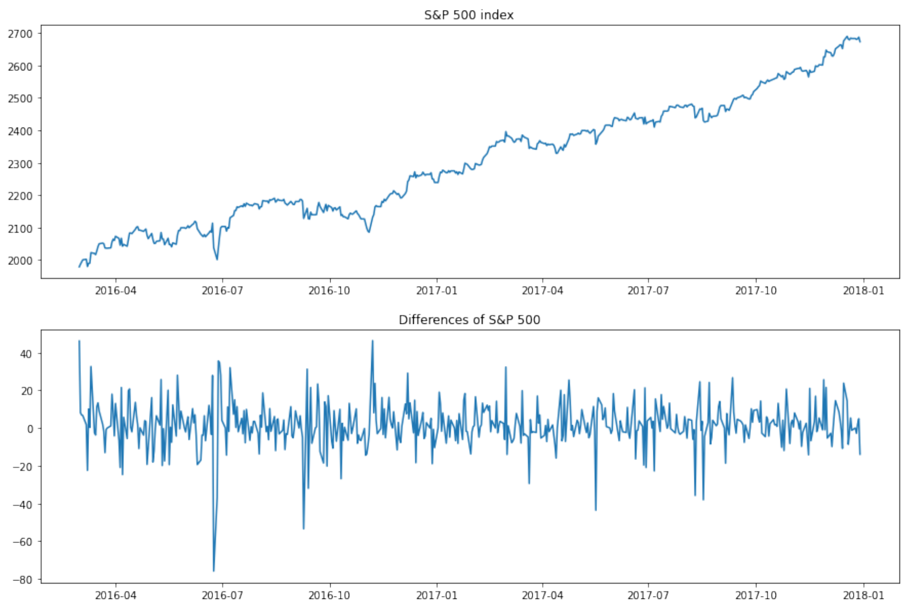

Figure 2: Time series of S&P 500 index from 1.3.2016 to 31.12.2017 both original and differenced. Original shows a clear upward trend, while there is no trend visible in the differenced one.

PACF α of x_t at lag k is

$$\alpha(k) = \phi_{kk}, k \geq 1, \tag{10}$$

where ϕ_{kk} is

$$\begin{bmatrix} \rho(0) & \rho(1) & \rho(2) & \dots & \rho(k-1) \\ \rho(1) & \rho(0) & \rho(1) & \dots & \rho(k-2) \\ \vdots & & & & \vdots \\ \rho(k-1) & \rho(k-2) & \rho(k-3) & \dots & \rho(0) \end{bmatrix} \begin{bmatrix} \phi_{k1} \\ \phi_{k2} \\ \vdots \\ \phi_{kk} \end{bmatrix} = \begin{bmatrix} \rho(1) \\ \rho(2) \\ \vdots \\ \rho(k) \end{bmatrix}. \tag{11}$$

AR(p) processes have the values of PACF equal to zero after lag of p and the values of ACF decay exponentially. MA(q) processes have the values of ACF equal to zero after lag of q and the values of PACF decay exponentially. For the ARMA(p,q) processes both ACF and PACF decay exponentially. Seasonal AR(P) process has the values of PACF equal to zero after lag of $P \cdot s$ with spikes s units apart. Similarly seasonal MA(Q) process has the values of ACF equal to zero after lag of $Q \cdot s$ with spikes s units apart (Brockwell and Davis, 2009).

The autoregressive and moving average orders and seasonality of a SARIMA model can be selected by examining the ACF and PACF of a stationary time series. For example Figure 3 shows the ACF and PACF plots of a once differenced S&P 500 time during training 1.3.2016 - 2.12.2016. Values of ACF and PACF show possible spikes at the lag of 3. This would result in the SARIMA(0, 1, 0)(1, 0, 1)$_3$ to be selected as the model. However, there are multiple possible spikes: at lags of 1, 3,

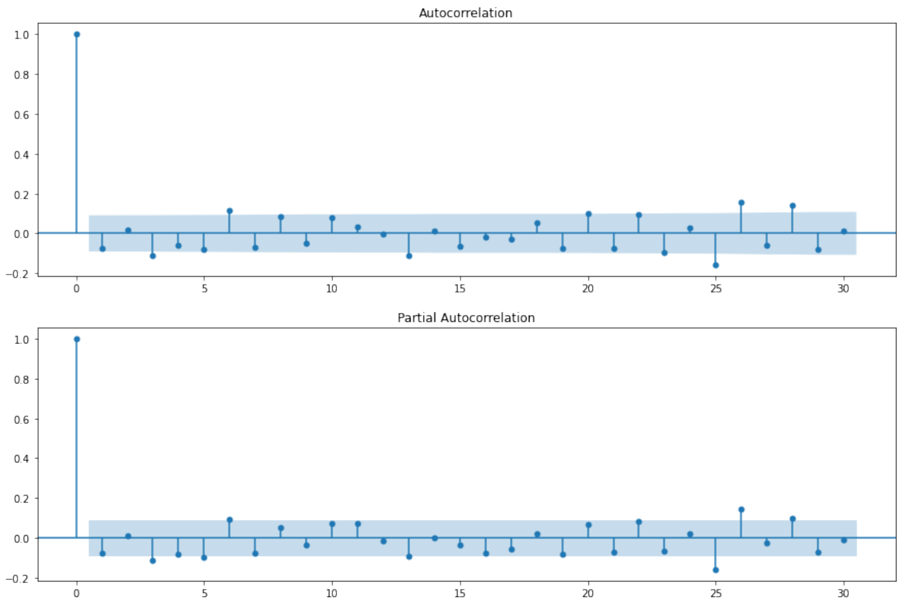

Figure 3: ACF and PACF values of differenced S&P 500 during the training perioid of 1.3.2016 - 2.12.2016. Both show values that are near the border of condifence interval at the lags of 1, 3, 5 and 6.

5 and 6. They are near the border of the confidence interval and none of them are clearly significant. It suggests that another model could also be selected to describe the time series. Possible spikes at large lag values are not considered.

For determining if autocorrelations and partial autocorrelations differ significantly from zero, confidence intervals are computed from standard deviation, which are in turn computed with Bartlett's formula (Brockwell and Davis, 2009). For these computations level α is given and it results in $(1-\alpha)\cdot 100\%$ confidence intervals. An example of such a confidence interval can be seen as the light blue area in Figure 3.

The parameters, ϕ_i, Φ_j, θ_k and Θ_l, of the obtained ARIMA model need to be estimated. This can be done by using for example linear or non-linear least squares method, maximum likelihood method or Bayes' theorem (Box et al., 2015). Parameter estimation is done in this thesis using Kalman filtering in Statsmodels library for Python (Fulton, 2015). There are also other available methods for parameter estimation in Statsmodels.

Once specific parameters for the model are obtained, it needs to be validated. This includes the calculation of residuals. Residual or residual error e_i is the difference between real value x_i and predicted value $\hat{x}_i$

$$e_i = x_i - \hat{x}_i, \tag{12}$$

Then it is checked that residuals are not correlated. This can be done by visual inspection of the residual plot. Residuals of a prediction should be evenly spread around zero and there shouldn't be any patterns visible.

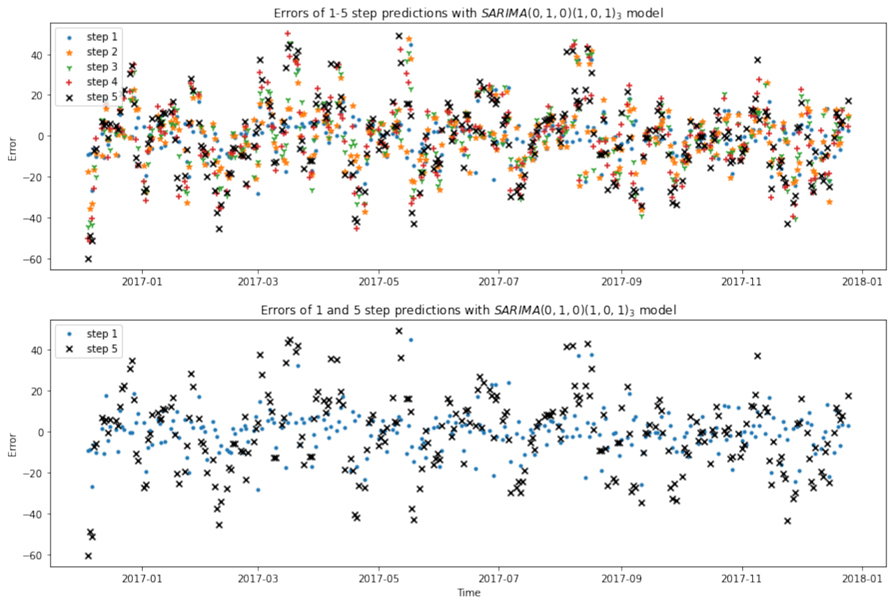

Figure 4: Errors of 1 to 5 step predictions of S&P 500 index with SARIMA(0,1,0)(1,0,1)$_3$ model. Errors are evenly spread on both sides of zero and there are no patterns visible.

For example when predicting S&P 500 index with SARIMA(0,1,0)(1,0,1)$_3$ model, the prediction errors for steps one to five in Figure 4 are evenly distributed and show no signs of autocorrelation. This suggests a valid model.

3.5 Analysis setting for predictions using ARIMA

In this thesis ARIMA models are used to produce from one to five step predictions of S&P 500 and OMXH25 indices. One step prediction corresponds with a one business day and 5 step prediction forecasts index value one week ahead. Mean squared error is used as the error term, since it is desired to avoid large errors in the forecast as large errors can result in large financial loss.

For shorter time series 1.1.2015 - 31.12.2015 and 1.3.2016 - 31.12.2017 first 200 observations and for the longer series from 1.1.2015 to 31.12.2019 the first 800 observations are used for training the model. These correspond to time spans of around 10 months for the shorter time series and 3 years and 4 months for the longer time series.

If data seems to have a trend, it is removed by differencing. Autocorrelation functions and partial autocorrelation functions are plotted with a confidence level of $\alpha = 0{,}2$ for determining the ARIMA models. ACFs, PACFs and ARIMA models are calculated with Python using Statsmodels library (Seabold and Perktold, 2010).

Table 1: ARIMA models used for predicting each time series. Models are selected based on need for differencing and ACF and PACF plots.

time series	model
S&P 500 2015	SARIMA(0,1,0)(1,0,1)$_4$
OMXH25 2015	SARIMA(0,1,0)(1,0,1)$_4$
S&P 500 2016-2017	SARIMA(0,1,0)(1,0,1)$_3$
OMH25 2016-2017	SARIMA(0,1,0)(1,0,1)$_5$
S&P 500 2015-2019	SARIMA(0,1,0)(1,0,1)$_6$
OMXH25 2015-2019	SARIMA(0,1,0)(1,0,1)$_5$

After differencing and plotting autocorrelation and partial autocorrelation functions, the models can be determined. The resulting models are listed in Table 1. In addition to these ARIMA models, a model making predictions based on trendline or ARMA(0,0) model is chosen as a comparison in all time series.

The ARIMA models are used for making predictions from one to five steps ahead. When predicting multiple steps ahead the previous predictions will be used in the prediction of the next value. The current prediction will be temporarily added into the time series before making the next prediction. For example in the case of 5 step prediction, the 1 to 4 step predictions will be the last values of the time series. After one iteration of predictions from 1 to 5 steps, the next business days real index value is added to the data and parameters ϕ_i, Φ_j, θ_k and Θ_l of the ARIMA model are re-estimated. Model itself, meaning the order of autoregressive, moving average and seasonality, remains the same. This continues until all data is used. This means that as more and more data is added, more information is available for estimation.

When predictions are ready, squared errors e_i^2 are calculated for all predictions and mean squared errors

$$\text{mse} = \frac{\sum_{i=1}^{n} e_i^2}{n}, \tag{13}$$

where n is the number of predictions made, are calculated to describe the accuracy of the model when predicting the time series in question.

3.6 Recurrent Neural Networks

In machine learning neural networks consist of several layers of neurons, nodes or predictors that map the input vector to a predicted output label.

Neuron in Figure 5 is a mathematical function that takes input values $x_1, \ldots x_n$, and

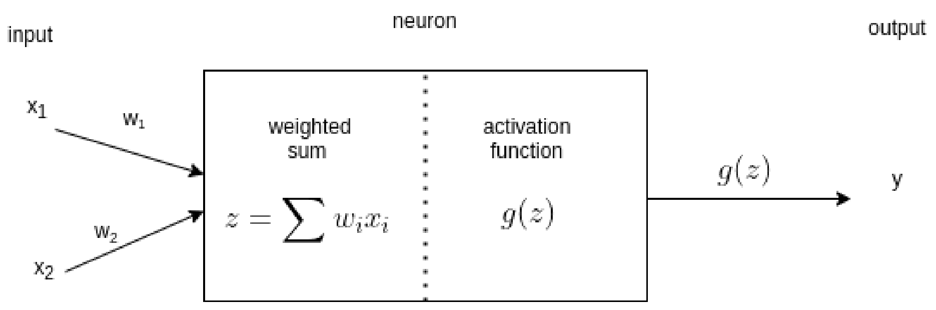

Figure 5: Neuron takes an input vector and maps it to an output using linear combination and a possibly non-linear activation function.

produces a weighted sum

$$z = \sum_{i=1}^{n} w_i x_i, \tag{14}$$

where w_i are the weight parameters used for mappings between neurons. Then an activation function g is applied to this sum and it produces the output value $\hat{y}$

$$\hat{y} = g(z) = g(\sum_{i=1}^{n} w_i x_i). \tag{15}$$

Activation function allows the model to non-linearly map the input values to output values (Jung, 2018). In this thesis a rectified linear unit or ReLU and hyperbolic tangent or tanh activation functions are used in the hidden layers. ReLU activation function

$$g(z) = \max\{0, z\} \tag{16}$$

outputs the value of variable or zero. Tanh activation function

$$g(z) = \tanh z = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{17}$$

maps the variable between -1 and 1. In the final output layer a linear activation function is used, which leaves the weighted sum of inputs untouched. It is possible to choose any activation function even though there is a habit of using specific functions in certain situations (Goodfellow et al., 2016; Géron, 2019).

When neurons are connected with each other an artificial neural network is formed. ANN presented in the Figure 6 consists of several layers of neurons. A layer takes the outputs from the neurons of the previous layer as input, maps them to output and passes them to the next layer until final output values are produced.

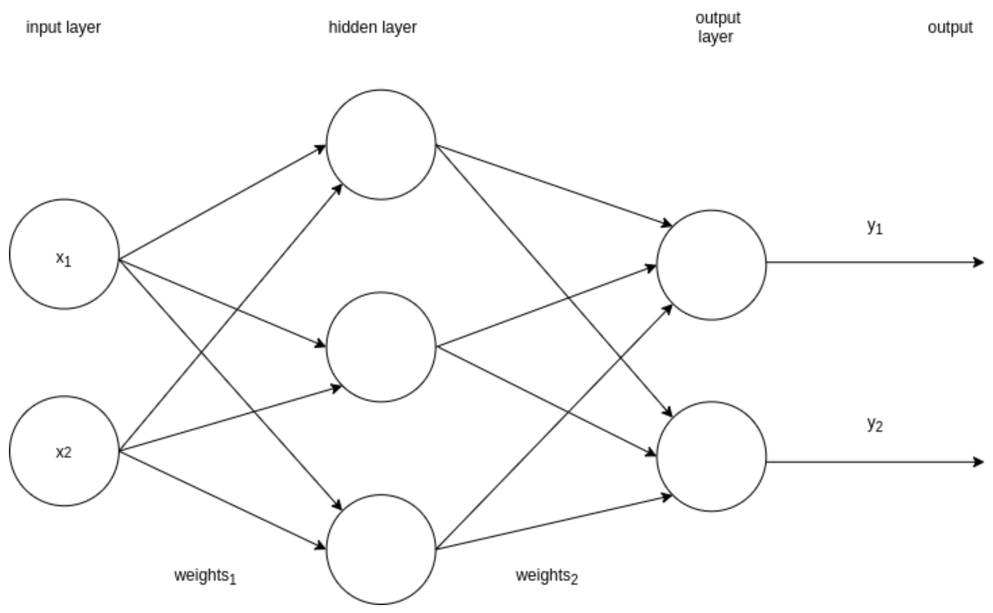

Figure 6: Neural network maps the feature vector or input into a label vector or an output using several layers of neurons. Each neuron is connected to all neurons of the previous and next layer.

First layer of an ANN is called the input layer and it consists of neurons containing the values of the feature vector. That is, it contains the attributes that are used for learning. In this thesis the input vector contains differences between two consecutive close values of a stock market index. This is followed by one or more hidden layers. They contain layers of neurons that map the input received from the previous layer to an output. The activation function in hidden layers is typically non-linear and it is often efficient to have more than one hidden layer (Thomas et al., 2017). The final layer of an ANN is the output layer. It contains the neurons producing final output values of the model. In this thesis output values will be the 1 to 5 step predictions for the change in index values and linear activation function is used as a real value is needed as an output.

If an ANN is used for a sequence of mappings and the information of all mapping is wanted to store, recurrent neurons can be used. A recurrent neuron stores information about previous timesteps in memory cells and its value is called cell state

$$h_t = f(h_{t-1}, x_t), \tag{18}$$

where h_t is the current cell state, h_{t-1}, is the previous cell state and x_t are the input values in each time step. The output of a recurrent neuron depends on the weighted sum, as with neuron (Figure 5), cell state and activation function.

When these recurrent neurons are stacked into layers, a sequence to sequence RNN in Figure 7 is formed. It takes a sequence as an input and produces a sequence as an

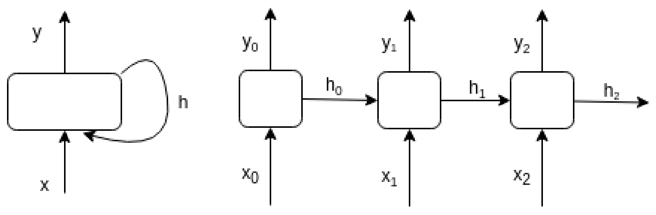

Figure 7: In RNN each layer takes also the previous time steps cell states of the memory cells as input.

output. As RNN consists of recurrent neurons it makes the learning of sequential patterns in training data possible (Goodfellow et al., 2016; Géron, 2019).

For an ANN to be useful, its weight parameters need to be selected in a meaningful way. Here the concept of loss helps. Loss is given by the loss function that evaluates the precision of the model, that is how large of a penalty is perceived when the estimated value differs from the true value. Loss and loss function are used for training the model and ANNs weights are optimized so that there is a minimal loss for the training data.

Loss function is not fixed and its selection is a design choice. For prediction of real values, loss functions relating to distance measures can be a good choice. In this thesis mean squared error is used as the loss function. This is used also for evaluating the precision of the model just as with ARIMA models, making the comparison of models possible.

Often the optimization problem of finding weights that minimize the loss function is not trivial and this optimization is done by using a gradient descent algorithm. Generally these algorithms involve taking a gradient of the loss function to find a direction in which the loss increases fastest and then taking a step, of a size η or learning rate, to the opposite direction. Then the weights are updated correspondingly so that the loss becomes smaller. This is iterated until a local minimum is found or another condition for ending the search is met.

Computing of the gradients can be done through backpropagation. This involves computing the partial derivatives one layer at a time, beginning from the last layer. Then this intermediate result is used for calculating the partial derivatives for the next layer. This is repeated until the gradient is computed. When calculating gradients for RNNs it is called backpropagation through time. Here the weights connecting recurrent neurons between time steps are taken into account (Goodfellow et al., 2016).

In this thesis adaptive moment estimation or Adam (Kingma and Ba, 2014) is used for optimizing the weights. Adam is a gradient based method for stochastic optimization. It helps to solve the possible issue with diminishing or exploding gradients when training the neural network with backpropagation. There are also several other well-established optimization methods and it is a design choice what method to use (Goodfellow et al., 2016).

Lastly, some sort of regularization is needed. Regularization is a term that is used when talking about methods to avoid overfitting. One simple regularization method is called dropout (Srivastava et al., 2014). With dropout in all training steps, all other neurons than the ones in the output layer, have a probability of p to be ignored or dropped out during that step. This p is called the dropout rate. The dropout forces the network not to rely on just a few connections.

3.7 Analysis setting for predictions using RNN

Recurrent Neural Networks used in this thesis were programmed with Python using Keras (Chollet et al., 2015) a high level API of TensorFlow (Abadi et al., 2015). Finding suitable models was a trial and error process that was supported by examples (Goodfellow et al., 2016; Géron, 2019). Differentiated time series of the indices was used for training the RNN models as was with the ARIMA models.

Table 2: Hyperparameter selection for the two RNN models was a trial and error process that resulted in two working models.

model	hidden layers	neurons in hidden layer	activation function	optimizer	loss	batch size	epochs
RNN-2	2	10	tanh	Adam	mse	5	5
RNN-3	3	20	ReLU	Adam	mse	5	10

Two different RNNs were selected as models. Hyperparameters were selected through an experimentation process and results can be seen in Table 2. In addition to these selections a dropout rate of 0,1 was used in all models. When choosing the hyperparameters, one parameter at a time was altered until the validation error of the chosen model didn't get any smaller. When tuning the hyperparameters 20 % of the training data was used as validation data.

Both models have an output layer of 5 neurons. The outputs predict the daily change of the stock market index in question from 1 to 5 steps ahead. The output layer uses linear activation function as the predicted value is a real number (Géron, 2019).

Hyperparameter batch size gives the number how many training samples from the training data are used at a time for training the model or updating the weights. Hyperparameter epoch gives the number how many times the whole training data is looped through during the training process.

4 Results

When making one to five step predictions of the six time series, four different types of models are used: trendline or ARMA(0,0), a SARIMA, a RNN with 2 hidden layers and a RNN with 3 hidden layers. This totals in 24 sets of predictions and 30 sets of comparisons as all four models are compared in each time series with each prediction length. The performance of these predictions is assessed by comparing their mean squared errors that can be found in Tables 3 - 8. The smallest error in each comparison is indicated with a green cell.

Table 3: Table containing the mean squared errors for all predictions for S&P 500 index from 1.1.2015 to 31.12.2015.

models	1 step	2 step	3 step	4 step	5 step
SARIMA $(0,1,0)(1,0,1)_4$	363,60	699,87	855,76	936,61	1112,39
ARMA(0, 0)	346,52	666,17	812,43	896,12	1072,37
RNN 3-layers	375,58	690,52	792,05	901,46	1082,45
RNN 2-layers	350,52	678,63	803,20	889,77	1078,19

Table 4: Table containing the mean squared errors for all predictions OMXH25 index from 1.1.2015 to 31.12.2015.

models	1 step	2 step	3 step	4 step	5 step
SARIMA $(0,1,0)(1,0,1)_4$	1413,91	2699,99	3361,37	4317,10	5467,46
ARMA(0, 0)	1410,75	2717,20	3347,14	4263,94	5386,73
RNN 3-layers	1496,39	2902,22	3536,63	4459,72	5772,70
RNN 2-layers	1491,34	2844,64	3528,28	4526,19	5748,91

Tables 3 and 4 contain the mean squared errors for the time series of S&P 500 and OMXH25 indices from 1.1.2015 to 31.12.2015. Errors are considerably smaller for models predicting S&P 500 as its values are smaller and it shows less variation. All types of models win at least one comparison. However, the ARMA(0,0) or trendline performs the best on seven occasions out of ten.

Table 5: Table containing the mean squared errors for all predictions for S&P 500 index from 1.3.2016 to 31.12.2017.

models	1 step	2 step	3 step	4 step	5 step
SARIMA $(0,1,0)(1,0,1)_3$	102,38	185,26	261,28	323,55	366,91
ARMA(0, 0)	103,67	185,90	264,40	325,47	369,61
RNN 3-layers	110,40	185,08	243,96	310,56	374,62
RNN 2-layers	109,48	192,72	265,39	337,28	400,89

Tables 5 and 6 contain the mean squared errors for the time series of S&P 500 and OMXH25 indices from 1.3.2016 to 31.12.2017. These errors are smaller than with previous predictions (Tables 3 and 4) as the time series are more stable. In Table

Table 6: Table containing the mean squared errors for all predictions OMXH25 index from 1.3.2016 to 31.12.2017.

models	1 step	2 step	3 step	4 step	5 step
SARIMA $(0,1,0)(1,0,1)_5$	688,74	1341,74	1911,97	2514,91	2976,72
ARMA(0, 0)	663,85	1290,58	1848,58	2440,88	2929,24
RNN 3-layers	678,81	1329,41	1892,12	2502,50	2972,88
RNN 2-layers	678,33	1383,60	1829,09	2451,42	2880,24

5 when predicting S&P 500, the one step prediction with SARIMA-model has the smallest prediction error of all predictions made and the largest relative difference of 9,3 % between the predictions is with five step predictions. All models are quite evenly matched as ARMA(0,0) and RNN 3-layers have best predictions in three sets and SARIMA and RNN 2-layers in two sets.

Table 7: Table containing the mean squared errors for all predictions for S&P 500 index from 1.1.2015 to 31.12.2019.

models	1 step	2 step	3 step	4 step	5 step
SARIMA $(0,1,0)(1,0,1)_6$	664,69	1347,57	1921,41	2592,77	3281,06
ARMA(0, 0)	658,94	1333,97	1899,50	2557,96	3227,18
RNN 3-layers	661,42	1327,94	1895,60	2567,06	3224,11
RNN 2-layers	661,96	1337,69	1907,36	2562,89	3238,95

Table 8: Table containing the mean squared errors for all predictions OMXH25 index from 1.1.2015 to 31.12.2019.

models	1 step	2 step	3 step	4 step	5 step
SARIMA $(0,1,0)(1,0,1)_5$	1369,17	2722,44	4104,36	5535,02	6818,85
ARMA(0, 0)	1356,66	2740,18	4139,46	5577,27	6899,29
RNN 3-layers	1358,08	2748,78	4136,44	5565,92	6875,51
RNN 2-layers	1355,98	2742,90	4143,52	5566,18	6878,56

Tables 7 and 8 contain the mean squared errors for the time series of S&P 500 and OMXH25 indices from 1.1.2015 to 31.12.2019. These longer time series are unstable when compared to the shorter ones and this results in larger errors. Again all types of models win at least one comparison, SARIMA model winning the most times. Again all types of models win at least one comparison, SARIMA model winning the most times. The average errors for 5 step prediction in Table 8 correspond to 1,9 - 2,9 % change in index values, when compared to index value range in Table 9.

In all predictions it is evident that the longer to the future the predictions are made, the less accurate they become. This is intuitive since new information becomes available after each time step. Predictions of OMXH25 have greater errors than those of S&P 500 as OMX25 has greater average daily changes and overall values shown in Table 9.

Table 9: Table containing the smallest mean squared errors, the average squared daily changes and value ranges of all time series. These give some reference to the errors of the predictive models.

Time series	Best average prediction error	Average squared daily change	Index value range
S&P 500 2015	346,52	372,60	1867,61 - 2130,82
OMXH25 2015	1410,75	1782,10	2899,11 - 3647,13
S&P 500 2016-2017	102,38	150,17	1978,35 - 2690,16
OMH25 2016-2017	663,85	969,89	2997,29 - 4139,81
S&P 500 2015-2019	658,94	405,07	1829,08 - 3240,02
OMXH25 2015-2019	1355,98	1343,61	2858,41 - 4387,70

When comparing the average one step prediction errors to squared average daily changes of index values in Table 9, the average errors are smaller than average changes with shorter time series. This is explained by the fact that the models take the trend into account. When the trend has multiple changes in direction as with the longer time series, the models perform worse results than just predicting the next value to be the same as the last value in the current time series. Predicting the next value to be the same as last would give the same average errors as with daily changes.

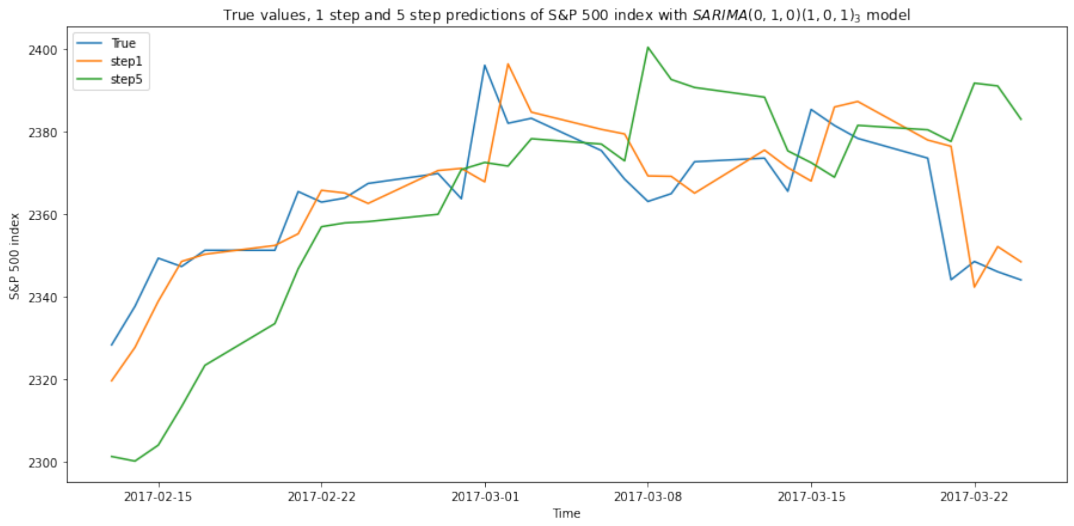

Figure 8: True values, 1 step and 5 step predictions for S&P 500 index with SARIMA(0,1,0)(1,0,1)$_3$. Predicitions follow the true values with a lag correspoding to how many steps to the future is being predicted.

Visual inspection of the predictions illustrate that the predictive models are not able to capture the behavior of the time series. Figure 8 shows one example of such a situation. Predictions of SARIMA(0,1,0)(1,0,1)$_3$ model follow the observed movements of S&P 500 index with a lag and are not able to predict what is about to happen. The lag is evident from the similar spikes at 1.3.2017, 2.3.2017 and 8.3.2017.

First being the true value, second 1 step prediction and the last 5 step prediction. This also means that when predicting longer in the future, as with 5 step predictions, the movements of the index are followed with lag corresponding to how many steps to the future is being predicted. This results in deteriorating predictions. Same behavior is present with all predictive models and time series.

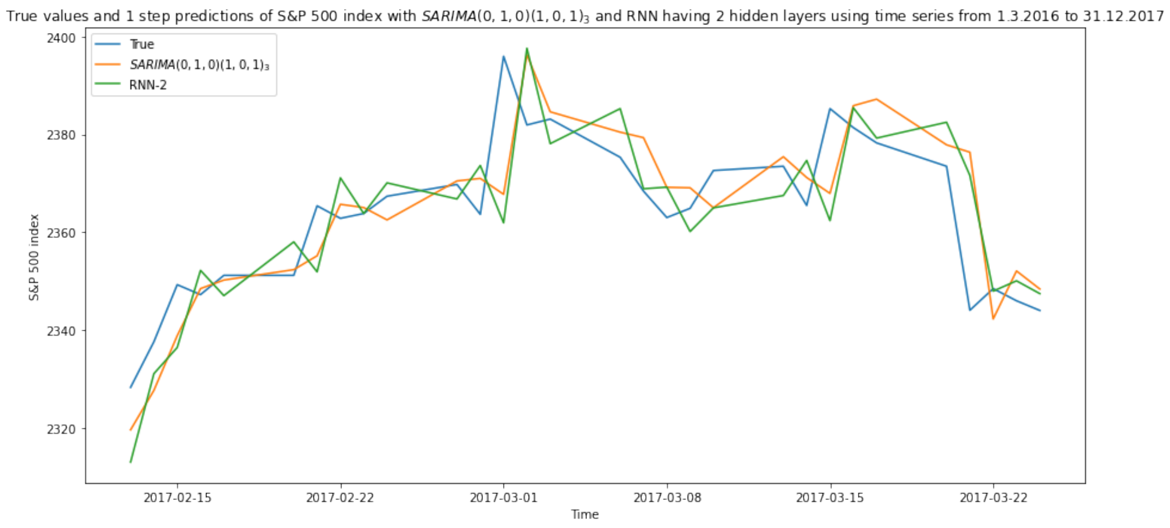

Figure 9: One step predictions with SARIMA(0,1,0)(1,0,1)$_3$ and RNN 2-layer models and true values when forecasting S&P 500 index. Predictions are similar but delayed.

Further analysis reveals that predictions made with all models are similar. It cannot be said that there is a difference between ARIMA or RNN models or models that have more or less parameters. Figure 9 contains one step predictions of S&P 500 during 2016-2017 time series from SARIMA(0,1,0)(1,0,1)$_3$ and RNN 2-layer models. It shows that predictions are similar and delayed. This can be seen from the spikes at 1.3.2017 for true index value and 2.3.2017 for predictions made with SARIMA(0,1,0)(1,0,1)$_3$ and RNN 2-layer models. Again this behavior is present with all predictive models and time series.

5 Summary and future prospects

In this thesis two autoregressive integrated moving average or ARIMA models and two recurrent neural network or RNN models were compared when predicting stock indices. The S&P 500 index, which consists of the 500 largest companies in the United States, and the OMXH25 index, which consists of the 25 largest companies on the Helsinki Stock Exchange were chosen as indices to forecast and only their close values were used for prediction. Three time series from both indices were analyzed: 1.1.2015 - 31.12.2015, 1.3.2016 - 31.12.2017 and 1.1.2015 - 31.12.2019. First two time spans contained no changes in trend, while the last longer time series had multiple. One to five step predictions were made for each time series with each predictive model and the goodness of predictions was evaluated with mean squared errors.

Results indicate that the ARIMA and RNN models produce similar predictions. This is intuitive as models use the same information for building the predictive models. Furthermore, any clear rankings between the four models cannot be formulated as all models give best predictions in some comparisons. None of the predictive models are able to capture the behavior of the predictive time series as they are not able to predict any rapid changes. The trendline predictions were hard to beat, but differences were small. However, the largest average errors of five step predictions corresponded to 1,9 % - 2,9 % of the index values. Models with errors of this magnitude are not ideal, if used as tools when making investment decisions.

In future it would be beneficial trying to add some external explanatory variables into the predictive models. If chosen wisely they could help bring the prediction errors down. Also the long short-term memory RNN architecture could be used for trying to capture patterns of time series better. Grid search could be implemented for tuning the hyperparameters of both ARIMA and RNN models as it would be a more systematic approach and it would give assurance about used hyperparameters. Finally, models could be used for predicting values of individual stocks as their time series might contain underlying patterns to discover.

References

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Available from: `https://www.tensorflow.org/`.

A. Ariyo, A. Adewumi, and C. Ayo. Stock price prediction using the arima model. In *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, pages 106–112. IEEE, 2014.

B. Bosworth, S. Hymans, and F. Modigliani. The stock market and the economy. *Brookings Papers on Economic Activity*, 1975(2):257–300, 1975.

G. Box, G. Jenkins, G. Reinsel, and G. Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

P. Brockwell and R. Davis. *Time series: theory and methods*. Springer Science & Business Media, 2009.

F. Chollet et al. Keras, 2015. Available from: `https://github.com/fchollet/keras`.

C. Fulton. Estimating time series models by state space methods in python: Statsmodels, 2015. Available from: `http://www.chadfulton.com/files/fulton_statsmodels_2017_v1.pdf`.

A. Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

A. Jung. Machine learning: Basic principles. *arXiv preprint arXiv:1805.05052*, 2018.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

P. Mondal, L. Shit, and S. Goswami. Study of effectiveness of time series modeling (arima) in forecasting stock prices. *International Journal of Computer Science, Engineering and Applications*, 4(2):13, 2014.

Nasdaq, Inc. Nasdaq index methodology: Omx helsinki 25 index, 2020. Available from: https://indexes.nasdaqomx.com/docs/Methodology_OMXH25.pdf Last accessed 12.8.2021.

Nasdaq, Inc. Omx helsinki 25 index, 2021. Available from: https://indexes.nasdaqomx.com/Index/History/OMXH25 Retrieved 9.7.2021.

D. Rumelhart, G. Hinton, and R. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

S. Seabold and J. Perktold. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.

S. Selvin, R. Vinayakumar, E. Gopalakrishnan, V. Menon, and K. Soman. Stock price prediction using lstm, rnn and cnn-sliding window model. In *2017 international conference on advances in computing, communications and informatics (icacci)*, pages 1643–1647. IEEE, 2017.

S&P Global. S&p 500 index, 2021a. Available from: https://www.nasdaq.com/market-activity/index/spx/historical, Retrieved 9.7.2021.

S&P Global. S&p u.s. indices methodology, 2021b. Available from: https://www.spglobal.com/spdji/en/documents/methodologies/methodology-sp-us-indices.pdf, Last accessed 12.8.2021.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

A. Thomas, M. Petridis, S. Walters, S. Gheytassi, and R. Morgan. Two hidden layers are usually better than one. In *International conference on engineering applications of neural networks*, pages 279–290. Springer, 2017.

B. Widrow and M. Hoff. Adaptive switching circuits. Technical report, Stanford Univ Ca Stanford Electronics Labs, 1960.

G. Yule. Vii. on a method of investigating periodicities disturbed series, with special reference to wolfer's sunspot numbers. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 226(636-646):267–298, 1927.