Spe-fucking-ciale

Andreas Borup Jørgensen - 20164559
Mette Koch Møller - 20164146
Robert Høstrup - 20166322
10/4. semester - Speciale

AALBORG UNIVERSITET

4. Juni 2021

Titelblad

- Uddannelse
- Semester
- Gruppenummer
- Fulde navne
- Evt. anslag

Abstract

- Et abstract er et kort resumé på engelsk, der placeres forrest i produktet.
- Er produktet skrevet på engelsk, kan abstractet skrives på dansk.
- Et abstract må max. være 2 normalsider

Contents

1 Introduction

2 Problem statement

How do machine learning perform compared to econometric models, when forecasting the Danish stock market?

wuhu

3 Literature review

This chapter is split into four sections. The search strategy is presented in the first section (3.1). In the second section (3.2) only existing literature concerning econometric methods will be presented. The third section (3.3) presents literature that focuses on machine learning alone. The last section (3.4) presents articles that compare econometric and machine learning models.

3.1 Search strategy

In this section, the overall search strategy for finding literature is explained. The search strategy was divided into four different stages: brainstorming, searching, reading, and selection.

The first stage was a brainstorm to find keywords for the actual search. The brainstorming of keywords took basis in the problem statement and preexisting knowledge. The keywords from the brainstorm were examined and split up into three groups: 'Econometric', 'machine learning', and 'comparative'. The keywords 'forecast' and 'stock' were included in all three groups.

The second stage, where the actual search was done, took its base in the first stage keywords. The entire search was made using the advanced search mechanism on Google Scholar. The different searches always included the words 'forecast' and 'stock' to limit the search to relevant material. To find the most relevant articles, the search was restricted to the first two pages on Google Scholar.

The articles were distributed in the third stage following the grouping created in stage one. Each article in the different groups was then thoroughly read.

At the fourth and final stage, the articles were discussed. The final articles were selected based on the discussions. The selected articles are presented in the following sections.

Author	**Year**	**Resume**	**Conclusion**
Mondal et al.	2014	Examination of different fitting period lengths when building an ARIMA	No signigicant difference between an ARIMA created on 23-, 18-, 12- or 6-months' ability to forecast
Afeef et al.	2018	ARIMA(1,1,1) forecast of Pakistani oil company's stock price	Strong ability to forecast in the short run
Guiao	2019	Forecasting Philippian stocks by ARIMA(1,1,2)-ARCH-(T,N)	Strong forecast with MAPE between 0,77 and 1,87 per cent
Mustapa and Ismail	2019	Comparison of static/dynamic forecasts using ARIMA(2-1-2)-Garch(1,1) on the SP500	Dynamic forecast outperformed static.
Virtanen and Yli-Olli	1987	Examination of uni-, multivariate and combined models' abilities to forecast	The Combined model proved to predict stronger forecasts with lower RSME
Purwa et al.	2020	Comparison of ARIMA, VAR and Transfer function for forecasting	Transfer function proved to do more precise forecast based on RSME.
Dingli and Fournier	2017	Comparison of different SVM models for Tech and Finance industry	Mixed result. Often Linear regression and SVM were the better model
Shen et al.	2012	Comparison of different SML models. Noget med portfolio	SMV gave the better forecast and outperformed the benchmark trading model
Reddy	2018	Forecasting stock by a SVM model with a RBD kernel	The model gave a higher profit than the benchmark model.
Abe and Nakayama	2018	Comparison of RF, SVR and, deep learning models	A deep learning model with more layers were the better model
Naik and Mohan	2019	Comparison of RF, SVR, ANN, and a deep learning	The deep learning model outperformed the other models.
Beyaz et al.	2018	Comparison of SVR and ANN models	SVR had the lowest RMSE
Tsantekidis et al.	2017	Comparison of SVR, ANN, and CNN models	A CNN model were the superior model
Mehtab et al.	2020	Comparison of SVM, ANN, and LSTM models	A LSTM model were the superior model
Balaji et al.	2018	Comparison of CNN, LSTM, GRU, and ELM models	GRU were better for shorter forecasts and ELM were better for longer forecasts
Qiu et al.	2020	Comparison of GRU, LSTM, WLSTM and WLSTM + attention models	WLSTM + attention models was the better model

Liu et al.	2019	Comparison of C1D-LSTM, C1D-ROC and a baseline model (WSAEs-LSTM)	Both outperformed the baseline model. C1D-LSTM were slightly better
Kim and Kim	2019	Comparison of CNN, LSTM and LSTM-CNN. The CNN model is used for image recognition	The LSTM-CNN were the superior model
Du	2018	Comparison of ARIMA, ANN and Hybrid models	ANNs outperform ARIMA and Hybrid models outperform both
Siami-Namini et al.	2018	Comparison of LSTM and ARIMA in financial and general economic data	LSTM achieves 84-87 per cent lower RMSE
Isenahd and Olubusoye	2014	Comparison of ARIMA and ANN on Nigerian stock market	The ANN outperform the ARIMA
Tiwari et al.	2017	Comparison of ARIMA, ANN, and a host of other model types	On general data ANN performs best

3.2 Econometric

3.2.1 ARIMA models

ARIMA models are often used to forecast financial assets. Mondal et al. (2014) and Afeef et al. (2018) both studies different ARIMA models and their ability to forecast.

In their article *Study of effectiveness of time series modeling (ARIMA) in forecasting stock prices* Mondal et al. (2014) examines the Indian stock market index NSE using an ARIMA(1,0,2). Mondal et al. uses the time period 2012-2014 for their studies. They test four different fitting period lengths: 23-, 18-, 12-, and 6-months. No significant difference was found in the models' abilities to forecast stock prices across different sectors. The models in the study all had an accuracy[1] of at least 85 per cent (Mondal et al., 2014).

In Afeef et al.'s (2018) study *Pakistan Oil & Gas Development Company Limited*, they focus on one company. They create different ARIMA models to forecast the Pakistani oil company *Pakistan Oil & Gas Development Company Limited*. Based on a time period from 2004 to 2018, they find that an ARIMA(1,1,1) have the strongest fit. Afeef et al. conclude that the model's ability to forecast is strong in the short run (Afeef et al., 2018).

[1] Accuracy is calculated using MAE

3.2.2 ARIMA-(G)ARCH models

Mustapa and Ismail (2019), and Guiao (2019) all studies forecasting of stocks by using ARIMA-(G)ARCH models.

Guiao (2019) investigates the Philippine stock market. In her article *Forecasting Philippine stock market prices using ARIMA-GARCH models*. She models three different Philippine stocks from different sectors. The goal was to forecast the stock price using an ARIMA-GARCH model. A model is created for each stock, all with an ARIMA(1,1,2) base. Two of the stocks' volatility is modelled with an ARCH-T, and one is modelled with an ARCH-N. Guiao creates the models in a time period from 2014 to 2019. The three models all produced strong forecast with MAPE between 0,77 and 1,87 per cent. (Guiao, 2019)

In their article *Modelling and forecasting S&P 500 stock prices using hybrid ARIMA-GARCH model* Mustapa and Ismail (2019) examines the American stock index S&P500. Mustapa and Ismail model the S&P500 in the time period 2001-2017, and use the model to forecast the following year 2018. The model used to forecast follows an ARIMA(2,1,2)-GARCH(1,1) structure. Two different methods are used when forecasting, dynamic and static. The methods are compared using RSME. Mustapa and Ismail concludes that the dynamic(RSME=120) forecast outperformed the static(RSME=125) one (Mustapa and Ismail, 2019).

3.2.3 Comparison of econometric models

Virtanen and Yli-Olli (1987) wrote an article named *Forecasting stock market prices in a thin security market*, where they examined different econometric ways to forecast the Finish stock market. The study examined the time frame 1975-1986, where the years 1975-1984 were used as a data frame to build the models, while the years 1985-1986 were used to forecast and test the forecasts. Virtanen and Yli-Olli build three different types of models; a univariate ARIMA model, a multivariate time series model and a combination of the two. The univariate method led to the construction of two different models: A classic ARIMA$(0, 2, 1)$ and an ARIMA with a seasonal component ARIMA$(0, 2, 1)(0, 1, 1)^4$. Both models were significant, with p-values smaller than one per

cent. The multivariate model was created with six explanatory variables: Lagged versions of the stock, anticipated future cash flow, return on Finish state bonds, money supply (Finish Mark), inflation, and the Swedish stock market (Stockholm Stock Exchange). The coefficients of all explanatory variables proved to be significant at a five per cent level. Two combined models were created: One where the standard ARIMA ($f(A)$) was combined with the multivariate model ($f(M)$) and one where the ARIMA with a seasonal component ($f(AS)$) were combined with the multivariate model ($f(M)$). The combined models had the formula:

$$y_t = \alpha + \beta_1 f(M) + \beta_2 f(A) + \epsilon \tag{1}$$

$$y_t = \alpha + \beta_3 f(M) + \beta_4 f(AS) + \epsilon \tag{2}$$

β_2 proved to be insignificant in the model with no season al component, as to why the model was not used for forecasting. Virtanen and Yli-Olli used many different methods to examine the strength of the model forecasts. The different methods all pointed to the same conclusion. The combined model (RMSE = 0,069) outperformed both the univariate ARIMA (RMSE = 0,095) and the multivariate model (RMSE = 0,082) (Virtanen and Yli-Olli, 1987).

In their article *Comparison of ARIMA, Transfer function and VAR models for forecasting CPI, stock prices, and Indonesian exchange rate: Accuracy vs. Explainability* Purwa et al. (2020) examined different time series models' abilities to forecast. They forecast three different variables: CPI, CSPI(Indonesian stock market) and the Indonesian exchange rate. Since our thesis primary focus is the stock market, only conclusions concerning the CSPI will be presented. They construct the models on a time period from 2006-2016, used 2017 as out of sample testing, and forecast in 2018. Purwa et al. conclude that Transfer function is better at forecasting than ARIMA- and VAR-models. Furthermore, they conclude that CPI is a good predictor of CSPI, while the exchange rate is an ineffective predictor (Purwa et al., 2020).

3.3 Machine Learning

3.3.1 Supervised Machine Learning

Dingli and Fournier (2017) wrote the article *Financial Time Series Forecasting – A Machine Learning Approach* where they created several Supervised Machine Learning (SML) models to predict the stock price for Tech and Finance industry. They created three different models: A classification that predicts whether the price goes up or down, a regression that predicts the change in the price, and a regression that predicts the actual price. Each of the models is estimated daily, weekly, monthly, quarterly, and yearly. The feature[2] variables are 200 different technical indicators: Currency exchange rates, world indices, and commodity prices. All the features are not included in each model; only the features with statistical importance were selected. Therefore each model is created with a different number of features (Dingli and Fournier, 2017).
Dingli and Fournier test different models' ability to forecast e.g. logistic regression, Support Vector Model (SVM), Decision Trees(DT), and a Random Forest(RF). The classification is evaluated using accuracy, and the regressions are evaluated using Root Mean Square Error (RMSE). Different models performed best, depending on time period and stock. However, linear regression and SVM were often the better models.

Shen et al. (2012) also tested several SML models in their article *Stock Market Forecasting Using Machine Learning Algorithms*. They created classification and regression models as well and found that SVM was a better model at forecasting the NASDAQ, DJIA and, S&P 500. When predicting whether the stock price rose or fell, the models had an accuracy of 74.4, 77.6, and 76 per cent, respectively. Different models were tested, and they found only including four features lead to a higher accuracy compared to models that included several. Additionally, they found that SVM is sensitive to the training set's size, where a longer training period leads to a higher accuracy. The article differs by creating three different trading models: One based on the results from the SVM model and two benchmark models. In most cases, the trading model containing the SVM results outperformed the benchmark models(Shen et al., 2012).

[2]In the machine learning terminology, explanatory variables are described as features.

An SVM model was also used by Reddy (2018) in his article *Stock Market Prediction Using Machine Learning*. Where Dingli and Fournier tested multiple different SML models, Reddy only uses an SVM, where he uses a Radial Basis Function (RBF) as a kernel. Furthermore, he delimits by only trying to forecast the daily stock price of IBM. Just like Shen et al. (2012), the final model generates a higher profit than the benchmark (Reddy, 2018).

3.3.2 Artificial Neural Network

In the article *Deep Learning for Forecasting Stock Returns in the Cross-Section*, the writers Abe and Nakayama (2018) create several machine learning models in order to forecast the stock price. The purpose of the models is to forecast the one-month-ahead stock price for the MSCI Japan index. The features are 25 different factors, e.g. book-to-market ratio, market beta and volatility (Abe and Nakayama, 2018).
Abe and Nakayama creates different models consisting of an RF, an SVR[3], and 16 different deep learning models. The different deep learning models are differentiated by their number of layers, units and percentage of dropout. The models are evaluated by their correlation coefficient, directional accuracy, and mean square error (MSE). Generally, the deep learning models had better results, where a greater amount of layers led to a better model (Abe and Nakayama, 2018).

Naik and Mohan (2019) found the same results in their article *Stock Price Movements Classification Using Machine and Deep Learning Techniques-The Case Study of Indian Stock Market*. Comparing an RF, SVM, Artificial Neural Network (ANN), and a deep learning model, they found that deep learning outperformed the remaining models for a classification (Naik and Mohan, 2019).

Beyaz et al. (2018) also compared an ANN and an SVR model in their article *Comparing Technical and Fundamental indicators in stock price forecasting*. They tried to combine fundamental and technical analysis in a machine learning model. Opposite Naik and Mohan (2019), Beyaz et al. found that the SVM had a lower RSME compared to the ANN model regardless of the combination of features (Beyaz et al., 2018).

[3]Same principle as an SVM model but for a regression problem.

In the article *Forecasting Stock Prices from the Limit Order Book using Convolutional Neural Networks*, Tsantekidis et al. (2017) create both SML and ANN models. Just like Abe and Nakayama (2018), and Beyaz et al. (2018), Tsantekidis et al. create an SVM and ANN model. However, they also introduce Convolutional Neural Networks (CNN). The ANN model uses a single hidden layer, while the CNN model consists of several pooling and convolution layers. They use three different evaluations methods and determine that the CNN model is the superior model (Tsantekidis et al., 2017).

Mehtab et al. (2020) create eight different SML and ANN models in their article *Stock Price Prediction Using Machine Learning and LSTM-Based Deep Learning Models*. They differ by creating four different Long Short Term Memory (LSTM) models. They found that the LSTM models were superior to the SML and ANN models Mehtab et al. (2020).

In their article *Applicability of Deep Learning Models for Stock Price Forecasting An Empirical Study on BANKEX Data*, Balaji et al. (2018) focus on deep learning models. They create fourteen different models based on the techniques: CNN, Gated Recurrent Unit (GRU), LSTM, and Extreme Learning Machines (ELM). The models differ by the number of layers and how many steps ahead the models forecast.They tested the models on different stock prices from the bank sector. All of the deep Learning models had a great result. The GRU model tends to have better results for a shorter forecast, while the ELM model tends to be the better model for longer forecasts (Balaji et al., 2018).

Qiu et al. (2020) investigate the LSTM model in their article *Forecasting stock prices with long-short term memory neural network based on attention mechanism*. They propose that a Wavelet transform LSTM with attention (WLSTM + attention) would result in more precise forecasts. They compare their suggested model with a regular LSTM, a regular WLSTM and a GRU model across three different stocks (S&P500, DJIA, and HSI). The RMSE, MAE, MSE and R^2 show that the WLSTM + Attention was the best model regardless of the stock (Qiu et al., 2020).

Liu et al. (2019) combines a CNN and an LSTM model in their article *Stock Prices Prediction using Deep Learning Models*. They found that too much noise in the data can lead a model to overfit. Therefore they create two models: A CNN model to denoise

and an LSTM to forecast. They found that denoising the data decreases overfitting and reduces dimensionality. Furthermore, they create a C1D-ROC to forecast the rate of change in prices. MAPE, Theil U, and the linear correlation between the forecasted price and the actual price are used to evaluate the models. The models were also compared to a baseline model (WSAEs-LSTM). Both models had better results than the baseline model, though C1D-ROC had slightly better results (Liu et al., 2019).

In *Forecasting stock prices with a feature fusion LSTM-CNN model using different representations of the same data*, Kim and Kim (2019) combine an LSTM and CNN model as well. However, the intention behind introducing a CNN model is entirely different from Liu et al. (2019) as CNN is used for image recognition on stock chart images.
They created three models: a CNN using stock charts pictures, an LSTM using stock prices, and a CNN-LSTM, which is a combination of the two models. The CNN-LSTM turned out as the superior model, while the CNN model was the worst (Kim and Kim, 2019).

3.4 Comparitive studies

In their article, *Application and analysis of forecasting stock price index based on combination of ARIMA model and BP neural network*, Du (2018) uses forecasting of the Shanghai Securities composition stock index to compare ARIMA, ANNs, and a hybrid of the two. The data consist of daily data points from January to December 2017. The models created are an ARIMA (2, 1, 1), a feed-forward neural network, and a hybrid ARIMA-Neural network model that uses the residual values from the ARIMA model as a feature for the neural network model.
The testing data show that the ARIMA was the worst model (RMSE = 89.21), the neural network was the second-best (RMSE = 65.73), and the hybrid was the best model by far (RMSE = 18.02) (Du, 2018).

Siami-Namini et al. (2018) compares ARIMA and LSTM models' ability to forecast both financial and general economic data in their article *Forecasting economic and financial time series: ARIMA vs. LSTM*. The data used consist of six stocks collected monthly from 1985 to 2018 for the financial forecasts. A range of different consumption metrics,

M1 money supply, and trade data was used for the general economic forecasts. The models compared are an ARIMA(5, 1, 0) and an LSTM model with four units in its LSTM layer.

The results presented by Siami-Namini et al. show that the LSTM model on average has an 87 per cent lower RMSE in financial forecasts and 84 per cent lower RMSE in general economic forecasts. They also show that the amount of iterations in the LSTM-model does not affect the final forecasting accuracy (Siami-Namini et al., 2018).

In their article *Forecasting nigerian stock market return using arima and artificial neural network models*, Isenahd and Olubusoye (2014) similarly conclude that ANNs (RMSE = 0.0150 - 0.1055) outperform ARIMA (RMSE = 5) models when used for forecasting. The neural networks used was feed-forward neural networks, and the ARIMA was a non-seasonal (3, 0, 1) (Isenahd and Olubusoye, 2014).

In their article *Stock price prediction using data analytics*, Tiwari et al. (2017) expand their comparison to include Holt winter, linear, and seasonal trend loess models, in addition to ARIMA and ANN models. These models are compared when forecasting the opening price of the Indian NIFTY 50 stock index. Tiwari et al. find that ANNs perform the best (RMSE = 165), but if a polynomial or time series trend is added to the data, the results change. For the polynomial trend, an ARIMA performs the best (RMSE = 558), while a Holt winter model performs best (RMSE = 181) with a time series trend (Tiwari et al., 2017).

4 Linear models

Time series regression models are built on different types of equations, all with stochastic components. These models are often used to forecast the future based on historic data.

4.1 Stationary

In order to be able to forecast a time series, it is often necessary that the series is stable and predictable. This means that the series needs to be stationary. Several conditions need to be met in order to deem a time series stationary. These conditions state that a time series' mean, unconditional variance, and autocovariance needs to be independent of time.

Time series that stems from raw data are often not stationary. The raw data tend to have a trend or are seasonally correlated, making it hard to model and forecast. Non-stationary time series, however, can become stationary by differentiating the data.

4.2 AutoRegressive Integrated Moving Average

A model often used to forecast a stationary time series is the AutoRegressive Moving Average (ARMA) model. The ARMA model consists of two components; an AR- and an MA-component.

The AR component is a linear model that makes the forecast based on lagged versions of the dependent variable. The model is expressed by:

$$AR(p) \quad y_t = \alpha_0 + \sum_{i=1}^{p} \alpha_i y_{t-i} + \varepsilon_t \tag{3}$$

p indicates the number of lags, y_t is the dependent variable, α_0 is a constant, α_i are parameters, and ε_t is a white noise component.

The second part of an ARMA model is the MA component. The MA component describes a model where forecasts are made based on the earlier periods' error term. The model is

expressed by:

$$MA(q) \quad x_t = \sum_{i=0}^{q} \beta_i \varepsilon_{t-i} \tag{4}$$

q denotes the number of lags, ε_t is a white noise component, and β_i are parameters.

As explained, the ARMA model is a model that combines the two models AR and MA. The ARMA model is therefore expressed by: En ARMA kombinerer de to modeller til følgende model:

$$ARMA(p,q) \quad x_t = \alpha_0 + \sum_{i=1}^{p} \alpha_i y_{t-i} + \sum_{i=0}^{q} \beta_i \varepsilon_{t-i}. \tag{5}$$

As explained in section 4.1, is stationarity in the data set a requirement for forecasting using ARMA models. Stationarity can be secured by differentiating the data. If the data used for the ARMA model is deemed non-stationary and therefore differentiated, the ARMA is denoted as an AutoRegressive Integrated Moving Average (ARIMA) model. In an ARIMA *(p,d,q)* denotes p and q still the number of AR and MA components, while the d denotes how many times the data set is differentiated.

4.2.1 Auto correlation and partial auto correlation functions

Lidt om modellens egenskaber

5 Machine learning

Machine learning and AI is a broad field of study and contains many different models. For this study, supervised machine learning is used, more specifically, Artificial Neural Networks (ANN). The following section explains the theory needed to understand and create both a feed-forward and a recurrent neural network.

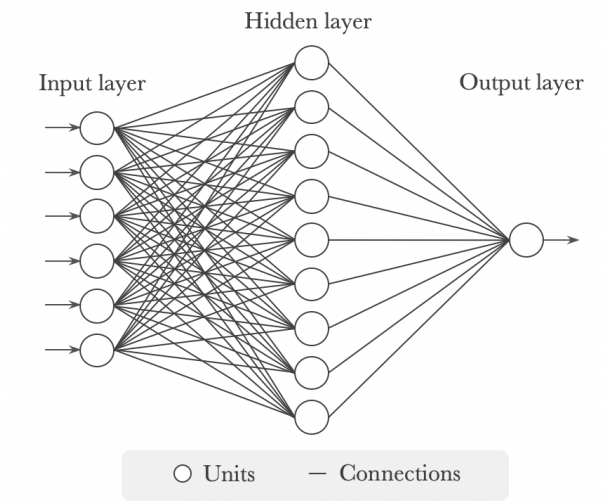
5.1 Out-of-sample testing

Machine learning does not use the same tests and methods as econometrics to determine whether a model's results are significant. Instead, out-of-sample testing is used to generalise the model in machine learning. This method entails dividing the data into two parts, one for training and one for testing. The model is first trained using the training part of the data, and then to generalise the model, it is tested using new data from the testing part of the data.

5.2 Artificial Neural Networks

An Artificial Neural Network (ANN) is a machine learning method that is able to deal with more complex systems than traditional supervised machine learning methods. An ANN works by creating units and layers that do computations based on input and relay those computations further down the network. A simple feed-forward ANN (FNN) is illustrated below in figure 1:

Figure 1: Simple feedforward ANN



Reference: Mette et al.

The first part of an FNN is the input layer. This layer consists of one unit for each feature in the data. Each unit receives the data of one feature and assigns it a weight to determine how important this feature is to the model. Aside from the weights, each unit also contains a bias and an activation function. This is presented in equation 6:

$$\sigma(w_1 a_1 + b), \tag{6}$$

Where σ is the activation function, w is the weight, a is the data, and b is the bias.

The bias regulates whether the unit sends data through the network. The activation function is explained at the end of this section. The bias, input, and weights in each unit are different while they share the same activation function. Each unit's results in the input layer are then sent through the network to the next layer, which is the hidden layer.

The hidden layer differs from the input layer in how the number of units is decided. Instead of being decided based on the number of features in the data, the number of units is decided solely at the researcher's discretion. The chosen amount of units is often a trade-off between predictive power and overfitting the model. The units in the hidden layer also differ from the units in the input layer in the data they receive. Instead of

receiving raw data, they receive the results from all the units in the previous layer. Since they receive more data, they also have to assign more weights. This is presented in equation 7:

$$\sigma(w_1a_1 + w_2a_2 + w_3a_3 + ... + w_ia_i + b), \tag{7}$$

The notations are the same as in equation 6

The weights and bias are different in each unit, but the units in the hidden layer not only share an activation function but also shares the same data. The results from the units in the hidden layer are then sent to the output layer.

The number of units in the output layer depends on the problem that is being investigated. There would be one unit for a regression, and for a classification, there would be units equal to the number of classes predicted. The units in the output layer work the same way as the units in the hidden layer. Since the output layer is the last layer in an FNN, its results are not sent to another layer but are instead the entire model's result.

Activation function:
The units' activation function is used to normalise the output and determine whether the units' output is sent further down the network. Since this study uses a regression model, the Rectified Linear Unit (ReLU) function is used. This activation function changes all negative values to zero and does nothing to positive values. This means data is only sent through the network if the unit's output is above 0. The advantage of using ReLU is that it is an efficient activation function that also allows for back-propagation.

5.2.1 Training an FNN

Initially, the weights and biases are randomly selected in the model. However, they can be changed to improve the model's performance. The process of changing the weights and biases is often referred to as training the model.

When training an FNN, it is important to have a loss function to evaluate the model's performance. In this study, the loss function is the RMSE measurement described in section 6.1.

The loss function describes how far off the correct answer the model's predictions are.

The goal of training an FNN is to find a local minimum of the loss function. Reaching this local minimum is commonly done by using back-propagation. To understand back-propagation, it is helpful to start by considering the output layer, which consists of only one unit. When running a neural network, each observation is sent through all the layers, and the final prediction is the output of the last unit, as shown in equation 7. When an FNN makes a prediction, it is possible to calculate the loss of that prediction and then calculate how the model should be changed to reach the local minimum. A model's output can be changed by changing the variables shown in equation 7, namely w, a and b. Changing the units' weight and bias is relatively simple since their initial value is randomly chosen, but changing the unit's input is more complicated. For the first unit in a model, the input is simply the data given, but for all the following units, the input is the output of all units in the previous layer. To change the input of the last unit, the output of the second to the last layer needs to be changed. This change is done by back-propagation, which is a method of changing the weights, bias and input in a unit with respect to how it will change the following units and layers.

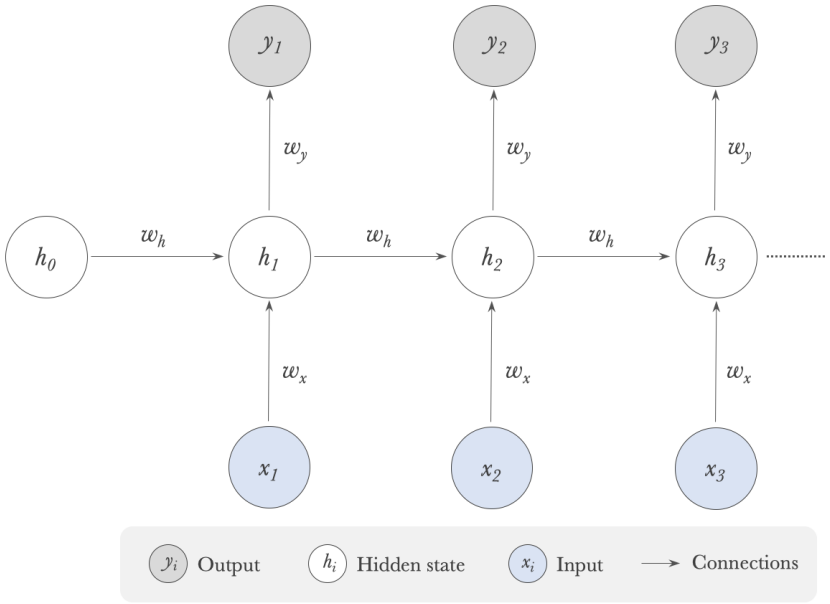
In theory, the most accurate way to reach a local minimum of the loss function is to calculate the loss individually for all observations. This method is often considered too time-consuming and demands too much computational power. To solve these problems, an optimiser is often used to speed up the process of back-propagation. One solution is to gather the data in batches and send them through the network together, thereby calculating their combined loss. This method is not as accurate but is significantly faster and reaches the same local minimum in the end. Too further increase the efficiency of the training process, momentum can be added to the model. In this project, the Adaptive Momentum Estimation (Adam) method is used to optimise the training. This method has been chosen since it offers the best balance of computational ease and accuracy.

5.3 Recurrent Neural Networks

A Recurrent Neural Network (RNN) is a type of ANN that deals with sequential data like time-series. The major difference between an RNN and an ANN is that the former incorporates memory about earlier instances in the data. This is done by adding a

hidden state to the units. This hidden state transfers what has been learned earlier in the sequence to the later parts (Pi, 2018b). An example of an RNN can be seen in figure 2.

Figure 2: Recurrent neural network



Reference: Own creation based on (Venkatachalam, 2019)

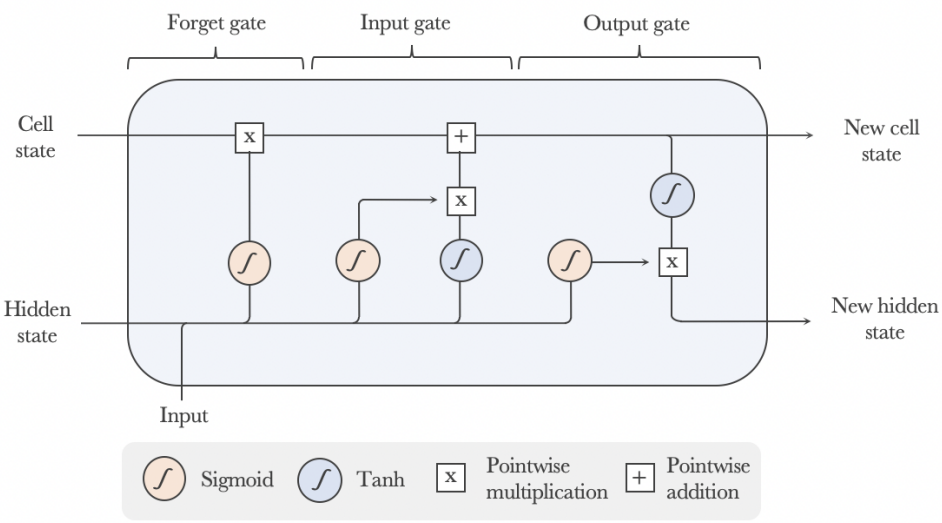
As illustrated in figure 2 the units assign weights to not only the output of the previous unit, but also the output of the earlier step in the sequence. The weight w_h represents the memory transfer and is what allows the RNN to retain a memory of the sequence. One drawback of this simple way of creating RNNs is that it suffers from short-term memory. The vanishing gradient problem causes this short-term memory. The vanishing gradient problem occurs because the earliest layers of an ANN often experiences the least amount of change during back-propagation. Thus, the last layers are often weighted higher when the ANN makes its final prediction, and for RNNs, only the most recent data hold any significant weight in the prediction (Pi, 2018b).

5.3.1 Long-short-term memory

A way to alleviate the vanishing gradient problem is to use a type of RNN called Long-Short-Term Memory (LSTM) architecture. The LSTM architecture utilises a gate struc-

ture that makes it possible to control what is remembered and forgotten by the network (Pi, 2018a). An example of this gate structure is illustrated below in figure 3.

Figure 3: LSTM gate



Reference: Own creation based on (Pi, 2018a)

In the LSTM architecture, the cell state is used to transport information through the sequence. It acts as the LSTM architecture's memory.

The first part of the LSTM is the forget-gate. This gate receives the cell and hidden state from the previous part of the sequence. It also receives input from the current part of the sequence. The input and hidden state are passed through a sigmoid function that creates values between 0 and 1. Values closer to 0 should be forgotten, and values closer to 1 should be remembered. After being passed through the sigmoid function, it is multiplied by the cell state. Thereby the model chooses how much of the earlier cell state should be kept.

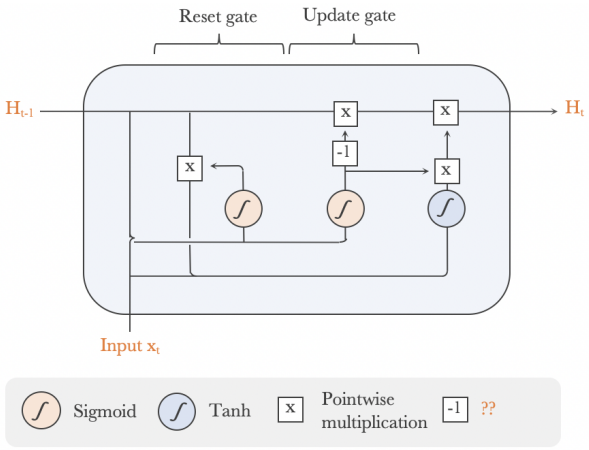
The next part of the LSTM is the input gate. In this gate, the hidden state and input are passed through both a sigmoid and tanh function. The sigmoid function again controls what is kept and what is forgotten. The tanh function ensures that the input values do not become unmanageable by limiting them to be between -1 and 1. The sigmoid and tanh function results are then multiplied to determine what the model should keep from the current input and then added to the cell state.

The last part of the LSTM is the output gate. In this gate, the hidden state and input are first passed through a sigmoid function. Then the cell state is sent through the tanh function. The transformed input and hidden state are multiplied by the cell state to determine what should be kept for the output. The result is a new hidden state that is passed on to the next part of the sequence. The cell state is similarly passed on to the next part of the sequence. The new hidden state also acts as the output for the current part of the sequence.

5.3.2 Gated recurrent unit

Another way of dealing with the vanishing gradient problem is to use a Gated Recurrent Unit (GRU). A GRU is similar to an LSTM, but i foregoes the cell state and instead uses the hidden state to transfer information. A GRU also only has two gates: a reset gate and an update gate.

Figure 4: GRU gate



Reference: (Pi, 2018a)

The GRU works by first sending both the new input and a hidden state through a sigmoid function in the reset gate. The result of this is then multiplied to the hidden state to decide what information from the past should be used. Next is the update gate, here the input and hidden state is passed through another sigmoid function and added to both the past hidden state and to the input. The update gate acts as a way of updating the current and new hidden states with new information from the input.

6 Evaluation methods

Different evaluation methods will be used to evaluate and compare econometric and machine learning models' ability to forecast financial assets. The following section describes the chosen evaluation methods.

6.1 Root Mean Square Error

Root Mean Square Error (RMSE) is used to test the models' abilities to forecast. RMSE is also used in the out-of-sample methodology when choosing the best ANN model.

RMSE describes an average deviation between the foretasted and the actual values, with a heavier weight on large deviations and a lighter weight on smaller deviations. The formula for calculating RMSE is shown in equation 8.

$$RMSE = \sqrt{\sum_{t=1}^{n} \frac{(y_t - \hat{y}_t)^2}{n}} \tag{8}$$

where y_t denotes the actual value at time t, $\hat{y}_t$ denotes the forecasted value at time t, and n denotes the number of forecasts.
The deviation is squared in the formula, which secures that RMSE always is presented in absolute values. This also has the before mentioned effect that large deviations get a heavier weight.

6.2 Mean Absolute Error

Like RMSE, Mean Absolute Error (MAE) describes an average deviation between the foretasted and the actual values. MAE does, unlike RMSE, not put a weight on the deviations, which means all the deviations get the same weight. The formula for calculating MAE is shown in equation 9.

$$MAE = \sum_{t=1}^{n} \frac{|y_t - \hat{y}_t|}{n} \tag{9}$$

6.3 Mean absolute percentage error

Another way of calculating an average deviation between forecasted and actual values is the Mean Absolute Percentage Error (MAPE). It differs by calculating the difference in percentage instead of in level. The formula for calculating MAPE is shown in equation 10.

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}|\frac{y_t - \hat{y}_t}{y_t}| \tag{10}$$

6.4 Diebold-Mariano test

The Diebold-Mariano (DM) test is used to investigate whether the difference in accuracy across models is significant. The DM test tests the difference by an ordinary students' t-test. Since the models in this project always use one step ahead forecasting, the DM test statistic can be calculated as:

$$DM = \frac{\bar{d}}{\sqrt{\frac{\gamma_0+2\gamma_1+...+2\gamma_q}{(H-1)}}} \tag{11}$$

Where H represents the number of forecasts used to calculate $\bar{d}$, γ_i is the i'th autocovariance of the d_t sequence, $\bar{d}$ represents the mean loss difference between model 1 and model 2:

$$\bar{d} = \frac{1}{H}\sum_{i=1}^{H}[g(e_{1i}) - g(e_{2i})] \tag{12}$$

Where $g(e_{1i})$ and $g(e_{2i})$ is the errors at time i, in model 1 and 2 respectively.
As shown in equation 11, The DM statistic is calculated by the mean of the loss differences between model 1 and 2, and the standard deviation of that mean ($\sqrt{\frac{\gamma_0+2\gamma_1+...+2\gamma_q}{(H-1)}}$). It is clear from the formula that if one or more γ_i are negative, the standard deviation of $\bar{d}$ could become negative. In order to counter such circumstances, Newey and West standard deviation is calculated and used.

The last step is to compare the DM statistic with critical values obtained from a student t-test with 1-H degrees of freedom. The null hypothesis in the DM methodology is $\bar{d} = 0$,

which means that there is no difference in the forecasted errors between model 1 and 2.

Table 1: Precision and sensitivity for all three models

		FNN	RNN	LSTM	bi-LSTM	GRU	biGRU
Carlsberg	Excl.	**7.00**	9.81	11.60	12.42	10.82	10.39
	Incl.	40.13	13.23	13.43	16.10	**12.12**	13.52
Genmab	Excl.	29.19	41.01	44.62	57.77	40.97	48.33
	Incl.	87.56	56.04	69.62	70.43	**50.39**	62.34
Jyske Bank	Excl.	**4.72**	6.33	6.63	8.58	6.28	8.59
	Incl.	13.59	9.26	8.44	11.56	**7.97**	10.38
Mærsk B	Excl.	**177.28**	264.39	286.84	287.69	259.63	264.84
	Incl.	**233.46**	290.04	316.98	332.42	275.74	280.32
Vestas	Excl.	**8.05**	11.95	12.21	14.85	11.97	12.58
	Incl.	**12.94**	22.84	23.12	25.67	18.61	19.26

(a) RMSE

		FNN	RNN	LSTM	bi-LSTM	GRU	biGRU	Mean
Carlsberg	Excl.	**0.71**	1.02	1.19	1.32	1.13	1.10	**1.89**
	Incl.	4.47	1.45	1.44	1.74	**1.34**	1.49	1.99
Genmab	Excl.	**1.75**	2.52	2.84	4.04	2.48	3.30	**2.82**
	Incl.	6.81	3.79	4.83	4.81	**3.34**	4.40	4.66
Jyske Bank	Excl.	1.02	1.49	1.59	2.02	1.45	1.86	**1.57**
	Incl.	3.28	2.20	2.10	2.79	**2.00**	2.49	2.48
Mærsk B	Excl.	**1.75**	2.51	2.82	2.84	2.52	2.56	**2.50**
	Incl.	**2.37**	2.91	3.19	3.32	2.68	2.83	2.88
Vestas	Excl.	**1.44**	2.03	2.04	2.48	2.00	2.14	**2.02**
	Incl.	**2.60**	4.13	4.02	4.47	3.02	3.28	3.57
Mean	Excl.	**1.89**	2.06	2.18	2.63	1.96	2.30	3.12
	Incl.	3.91	2.90	3.09	3.43	**2.50**	2.92	**2.16**
	Total	2.89	2.46	2.65	3.03	**2.22**	2.60	2.64

(b) MAPE

References

Abe, M. and Nakayama, H. (2018). Deep learning for forecasting stock returns in the cross-section. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 273–284. Springer.

Afeef, M., Ihsan, A., and Zada, H. (2018). Forecasting stock prices through univariate arima modeling. *NUML International Journal of Business & Management*, 13(2):130–143.

Balaji, A. J., Ram, D. H., and Nair, B. B. (2018). Applicability of deep learning models for stock price forecasting an empirical study on bankex data. *Procedia computer science*, 143:947–953.

Beyaz, E., Tekiner, F., Zeng, X.-j., and Keane, J. (2018). Comparing technical and fundamental indicators in stock price forecasting. In *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 1607–1613. IEEE.

Dingli, A. and Fournier, K. S. (2017). Financial time series forecasting-a machine learning approach. *Machine Learning and Applications: An International Journal*, 4(1/2):3.

Du, Y. (2018). Application and analysis of forecasting stock price index based on combination of arima model and bp neural network. In *2018 Chinese Control And Decision Conference (CCDC)*, pages 2854–2857. IEEE.

Guiao, J. E. (2019). Forecasting philippine stock market prices using arima-garch models.

Isenahd, G. M. and Olubusoye, O. E. (2014). Forecasting nigerian stock market returns using arima and artificial neural network models. *CBN Journal of Applied Statistics*, 5(2):25–48.

Kim, T. and Kim, H. Y. (2019). Forecasting stock prices with a feature fusion lstm-cnn model using different representations of the same data. *PloS one*, 14(2):e0212320.

Liu, J., Chao, F., Lin, Y.-C., and Lin, C.-M. (2019). Stock prices prediction using deep learning models. *arXiv preprint arXiv:1909.12227*.

Mehtab, S., Sen, J., and Dutta, A. (2020). Stock price prediction using machine learning and lstm-based deep learning models. *arXiv preprint arXiv:2009.10819*.

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

Mustapa, F. H. and Ismail, M. T. (2019). Modelling and forecasting s&p 500 stock prices using hybrid arima-garch model. In *Journal of Physics: Conference Series*, volume 1366, page 012130. IOP Publishing.

Naik, N. and Mohan, B. R. (2019). Stock price movements classification using machine and deep learning techniques-the case study of indian stock market. In *International Conference on Engineering Applications of Neural Networks*, pages 445–452. Springer.

Pi, M. (2018a). Illustrated guide to lstm's and gru's: A step by step explanation. `https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21`. Accessed 16. March 2021.

Pi, M. (2018b). Illustrated guide to recurrent neural networks. `https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9`. Accessed 16. March 2021.

Purwa, T., Nafngiyana, U., and Suhartono, S. (2020). Comparison of arima, transfer function and var models for forecasting cpi, stock prices, and indonesian exchange rate: Accuracy vs. explainability. *Statistics*, 2020.

Qiu, J., Wang, B., and Zhou, C. (2020). Forecasting stock prices with long-short term memory neural network based on attention mechanism. *PloS one*, 15(1):e0227222.

Reddy, V. K. S. (2018). Stock market prediction using machine learning. *International Research Journal of Engineering and Technology (IRJET)*, 5(10):1033–1035.

Shen, S., Jiang, H., and Zhang, T. (2012). Stock market forecasting using machine learning algorithms. *Department of Electrical Engineering, Stanford University, Stanford, CA*, pages 1–5.

Siami-Namini, S., Tavakoli, N., and Namin, A. S. (2018). A comparison of arima and lstm in forecasting time series. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1394–1401. IEEE.

Tiwari, S., Bharadwaj, A., and Gupta, S. (2017). Stock price prediction using data analytics. In *2017 International Conference on Advances in Computing, Communication and Control (ICAC3)*, pages 1–5. IEEE.

Tsantekidis, A., Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M., and Iosifidis, A. (2017). Forecasting stock prices from the limit order book using convolutional neural networks. In *2017 IEEE 19th Conference on Business Informatics (CBI)*, volume 1, pages 7–12. IEEE.

Venkatachalam, M. (2019). Recurrent neural networks. `https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce`. Accessed 16. March 2021.

Virtanen, I. and Yli-Olli, P. (1987). Forecasting stock market prices in a thin security market. *Omega*, 15(2):145–155.