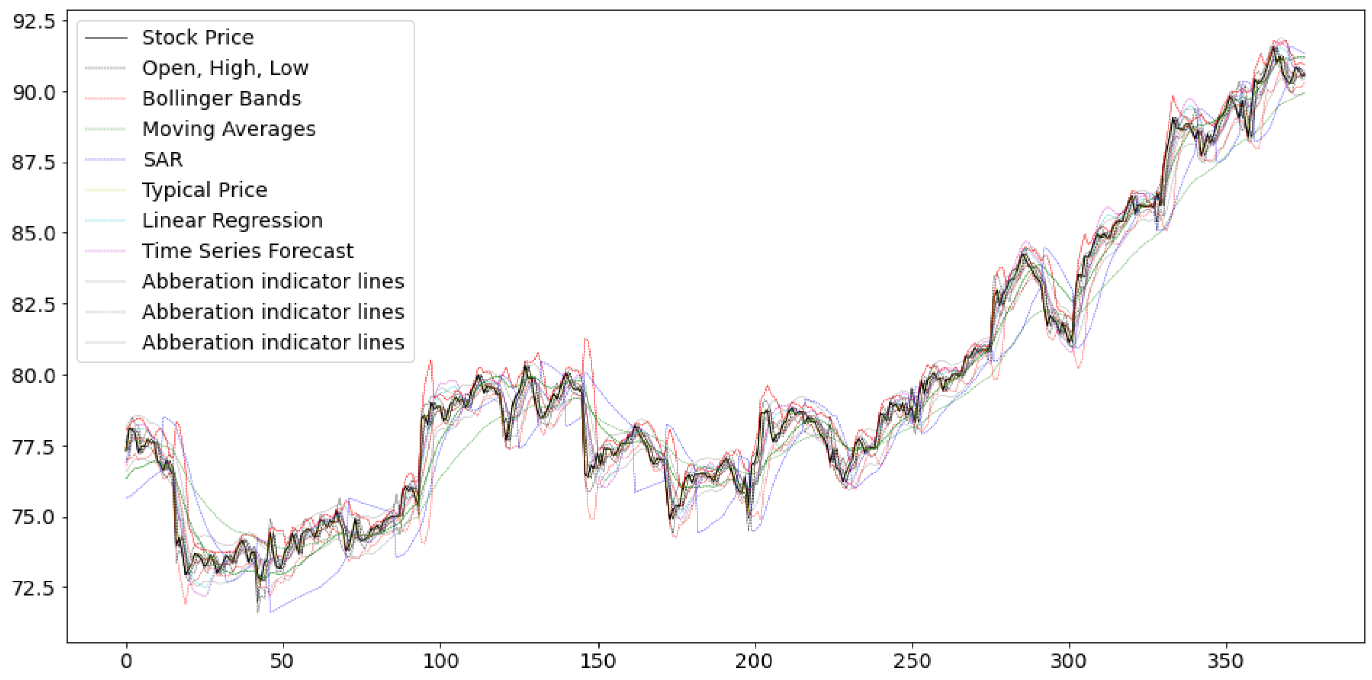


# Stock Market Prediction using a Deep Neural Network with A Large Set of Financial Indicators

## An Error Based Analysis on the Impact of Financial Indicators Using A Distributed Approach

Master's thesis in Computer science and engineering

IVISA STANIČIĆ

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2023

Master's thesis 2023

# Stock Market Prediction Using a Deep Neural Network with a Large Set of Financial Indicators

An Error Based Analysis on the Impact of Financial Indicators Using A Distributed Approach

IVISA STANIČIĆ

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2023

Stock Market Prediction Using a Deep Neural Network with a Large Set of Financial Indicators
An Error Based Analysis on the Impact of Financial Indicators Using A Distributed Approach
IVISA STANIČIĆ

Supervisor: Philippas Tsigas, Department of Computer Science and Engineering. Division of Computer and Network Systems.
Examiner: Marina Papatriantafilou, Department of Computer Science and Engineering. Division of Computer and Network Systems.

Master's Thesis 2023
Department of Computer Science and Engineering
Division of Computer and Network Systems.
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Stock visualization plots constructed in Python showing a selection of the data used for the machine learning. The x-axis showing the data points, each being 15 minutes apart in time. The y-axis showing the value of the stock and illustration of some of the indicators used in the project.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2023

Stock Market Prediction Using a Deep Neural Network with a Large Set of Financial Indicators
An Error Based Analysis on the Impact of Financial Indicators Using A Distributed Approach
IVISA STANIČIĆ
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

## Abstract

Correctly predicting the trends or values of different stocks within the stock market has been of general interest for many years. This thesis investigates the possibilities for further increasing the Root Mean Squared Error accuracy of current stock market prediction methods using Long Short-Term Memory. Specifically, the derivative as a function of the stock price as well as the time of day during the trade is investigated closer. The hypothesis behind these two approaches is that the relative difference of the stock price is important as well as the time of day which is assumed to impact the stock market. The investigation of time is examined by predicting and training the model on specific times of the day. Furthermore, the main method which is attempted is training the model on a much larger amount of indicators than has been traditionally used. In addition to the stock data, 92 stock market indicators were introduced to the model. The many different methods also required a task distribution system to be implemented for faster execution. A scalable distribution system was then created, increasing execution speed by up to 7.25 times. Results show that using a lower window size is beneficial for a problem this large. Furthermore, that time is an important factor in stock price prediction. The highest directional accuracy obtained was 55.4 % for afternoon predictions on the Advanced Micro Devices stock using all of the indicators. The best Root Mean Squared Error value was 0.53 for this stock and obtained without any indicators and only using the closing price for training. These results are based on 8 months of data with 15 minute intervals.

Keywords: Machine Learning, Stock Market Prediction, Long Short-Term Memory, Time Series

# Acknowledgements

I first want to express my gratitude for the support and interesting discussions from my supervisor Philippas Tsigas. You have pointed me in the right direction when I was in the most need for it. Furthermore I would like to thank Marina Papatriantafilou for agreeing to be an examiner for this work and for her comments and feedback. This thesis would not have been possible without either of you.

Finally I would like to thank my family and my friends for their support. You are the reason for why I am where I am today. I consider myself very lucky to have you in my life.

Ivisa Stanicic, Gothenburg, 2023

# List of Acronyms

Below is the list of acronyms, listed in alphabetical order, that have been used throughout this thesis:

| | |
|---|---|
| AMD | Advanced Micro Devices, Inc. |
| ANN / DNN | Artificial Neural Network / Deep Neural Network |
| API | Application Programming Interface |
| LSTM | Long Short-Term Memory |
| OHLC | Open / High / Low / Close - Stock Values |
| OHLCV | Open / High / Low / Close / Volume - Stock Values |
| RMSE | Root Mean Squared Error |
| TSF | Time Series Forecast |

# Contents

# 1
# Introduction

Stock market prediction has long been of interest from a financial point of view. Succeeding in predicting the stock market correctly could potentially lead to significant profit. Using machine learning to achieve this prediction is a promising idea. While there are many machine learning models which can be used for stock market prediction, a comparative study made by M. Nabipour et al. indicate that deep learning algorithms such as recurrent neural networks, from here on referred to as RNNs, perform the best [1].

Non-linear time series prediction using RNNs was made as early as 1991 by N.Z.Hakim et al [2]. However, as discussed by Y. Bengio et al., RNNs are not suitable for long-term dependencies as they lack control structures. The authors found that a system in a long-term context was either not efficiently trainable or not resistant to noise [3]. As the time series prediction in this project is the stock market where there is a possibility of long-term dependencies, standard RNNs will not be used.

S. Hochreiter and J. Schmidhuber proposed Long Short-Term Memory (LSTM) as a solution to this problem. LSTM is a recurrent neural network with control structures which should solve this issue. A more recent advance in the field of stock market prediction using LSTM, is a work by Y. Wang et al. which manages to achieve a prediction accuracy of over 60% on stock price forecasting [4].

## 1.1 Background and Thesis Layout

Estimation of the stock market has been done for many years with many different machine learning methods with the end goal of successful investments. This will likely continue as the world market value continues to increase. To give a quantitative number, as of writing this thesis the world market value has reached almost 100 trillion dollars [5]. Formerly created stock market prediction classifiers have managed an accuracy of up to 65% using a collaboration of classifiers such as neural networks and decision trees already in 2007 [6]. This thesis examines whether it would be possible to create an even better model, with a focus on keeping the Root Mean Squared Error (RMSE) as low as possible. The suggested improvement is to examine a plentiful amount of various financial indicators. In addition to this, the derivative of the stock market price and the time of day will be examined in more detail. This is done with the assumption that the stock market behaves differently depending on the time of the day, furthermore that the LSTM algorithm will react better to the derivative of the normalized price rather than only normalized price. The derivative will emphasize the rate of change to the algorithm, which is expected to in combination with the other indicators improve predictions.

The reasoning behind using a larger model than previously tested is that a deeper correlation between these indicators is expected and that they may have dependencies which are imperceptible for the smaller models and only realized through complex ones.

The main idea is that the algorithm will be given the same information as an experienced trader and be able to learn the different patterns which, in succession, increases the model's accuracy. This is explained further in section 1.3.1. This is also the main difference compared to other related work in the field. An additional difference is the span of predictions. The stock market can be predicted in different ranges. The focus of this study will be with a time span of 15 minutes, which is a common trading span used among intra-day traders. The same concepts, ideas and discussions could be applied to other intervals as well. Note that it is specifically the closing price of the 15 minute time frame that will be predicted to ensure that there is not a correlation between the opening price of one time frame and the closing price of the previous time frame.

During the time of the project, as a demonstration of how active the stock market is, the total number of trades made in the United States stock market on one day was almost 70 million with a total volume of over 10 billion [7]. The specific date chosen regarding the numbers was the 26th of August 2022. The idea of this display is to illustrate that with such a large number of trades it is probable that an algorithm can be constructed that can indicate whether a trade is predictable, and ultimately, whether a trade is profitable. This concept is the motivation for this work.

To establish a clear understanding of the thesis and its aim, a few concepts will have to be introduced, starting with an overview of the stock market. In addition to this overview, a subsection on technical and fundamental analysis of the stock market as well as a section on artificial neural networks and their relation to LSTM will be presented. A more detailed description of LSTM will be included as well. A section on theory will later follow, which will include the various indicators used in the project as well as the different model parameters for the neural network. Thereafter the method and details regarding the execution of the project will be presented. Data collection and processing is a part of this section as well. After this, the findings of the project will be presented and discussed. There will also be a section on future work and finally a section on conclusion.

## 1.2 Recurrent Neural Networks

Artificial neural networks, a term often heard in combination with deep learning, is a collection of artificially created and interconnected neurons. There are a few different kinds, such as convolutional neural networks, Long Short-Term Memory which will be used in this project and generative adversarial network to name a few. In deep learning applications, a multi-layered approach is commonly used where the first and last layers of the model are called visible layers. These are the output and input layers. The rest are intuitively called hidden layers as we do not have access to them from the output or input side. An increase of these hidden layers means that more complex patterns can be learned by the model [8].

For many applications, using the most recent information is sufficient for making accurate predictions. An example of this is a simple image classifier where the previous image is irrelevant to a prediction of the current one. For some applications however it is important to have a context to the prediction which is obtained by remembering previous information. Stock prediction is such an application, where previous patterns may repeat and need to be remembered. Standard RNNs are theoretically capable of remembering long sequences, but not practically as the complexity of introducing the RNN to the longer and more complex problem emerges [9].

Other common problems with standard RNNs are the exploding gradients which in turn leads to oscillating weights, as well as vanishing gradients which lead to longer training times or training with no improvement at all. It is for this reason, that the weights during back-propagation can vanish or explode, that a limitation on the RNN to learn longer problems becomes apparent [10]. A standard RNN layer is illustrated in figure 1.1. The encapsulated blue symbol represents a tanh function.

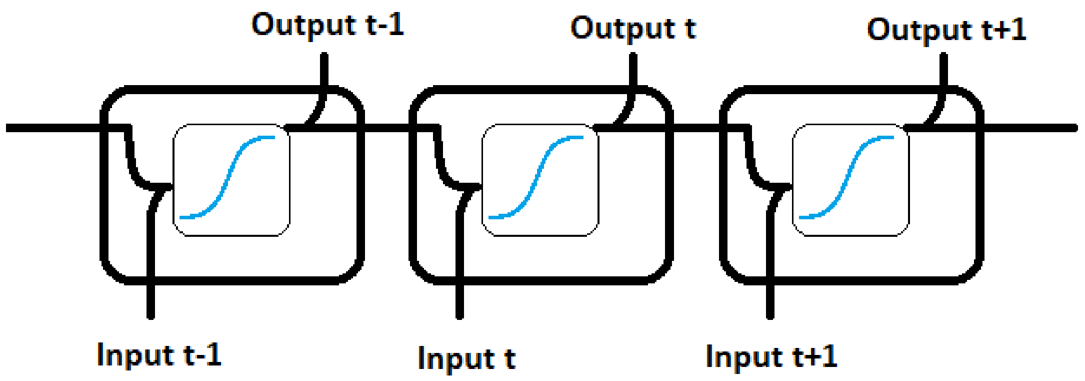


**Figure 1.1:** A visualization of a standard RNN layer.

### 1.2.1 Back-Propagation

When training a neural network, back-propagation is used to update the weights of the model. This is done with the help of the loss function which decides whether the prediction of the model is accurate and in turn how the weights should be updated. The weights are updated in a way which tries to minimize the chosen loss. The loss function therefore serves a purpose of indicating the correct function of the model by giving the algorithm a number to quantify how good of a prediction it made. The computational section is specifically calculated using the chain rule. This is done one layer at a time, iterating backwards. The problems associated with this is commonly the issue of exploding or vanishing gradients, where the model weights can be extreme the further back the propagation goes.

The vanishing and exploding gradient issues are caused by different properties of the RNN. Activation functions such as Sigmoid and Tanh within the network are one of the reasons for this issue. These functions have gradients which are between 0 and 1. Because of the use of the chain rule in back-propagation the gradient is multiplied and decreased exponentially the further back the propagation goes. This leads to a vanishing gradient. The exploding gradient however, is not caused by these functions as they are never larger than 1, it can instead be caused by the weights of the features used as these can be very large unless normalization is applied. Additionally, a more complex RNN network such as LSTM introduces addition to the neural network cell block which can lead to a larger gradient [11].

### 1.2.2 Long Short-Term Memory

Many of these issues caused during back-propagation can be handled using a variant of RNN called LSTM. LSTM enables a constant error flow through back-propagation which allows the gradient to flow unchanged, essentially meaning that the problem of the vanishing gradient is handled. This leads to an effective decrease in training time as the weights are updated functionally faster, in turn causing the end goal of minimizing the loss to occur faster as well [9].

The problem of having too large gradients is however still relevant. This is the reason for applying normalization to the data. This way, the impact of the gradient explosion problem will be greatly reduced and the exploding gradient problem is addressed as well [12]. Furthermore, another benefit of normalization is that features will be weighted equally within the algorithm rather than based on the size of the feature. See section 3.3 for a more detailed explanation of this.

The ability to remembering complex patters for a longer period of time could be considered the main advantage of using LSTM. This would be an issue for a standard neural network, as these lack the ability to store information for long. Using standard RNNs instead of LSTM would therefore be inappropriate for a stock market prediction application [8].

#### 1.2.2.1 LSTM Components

An LSTM cell consists of 3 sigmoid and 2 tanh activation functions which are used to calculate the cell state and hidden state, or rather output. A standard RNN cell in contrast to an LSTM cell is much simpler. It uses only one tanh function for the input concatenated with the previous state. The LSTM cell is divided into three gates called input, output and forget gate.

In figure 1.2, “t” denotes the order of the input and output sequence. “X” denotes point-wise multiplication and “+” denotes point-wise addition. The larger blue functions are tanh functions and the smaller red ones are sigmoid functions. The four trained neural networks are the four lower sigmoid and tanh functions. The fifth tanh function acts as a gate and is not a trained neural network, but simply forces values between -1 and 1 before being updated with the previous state.

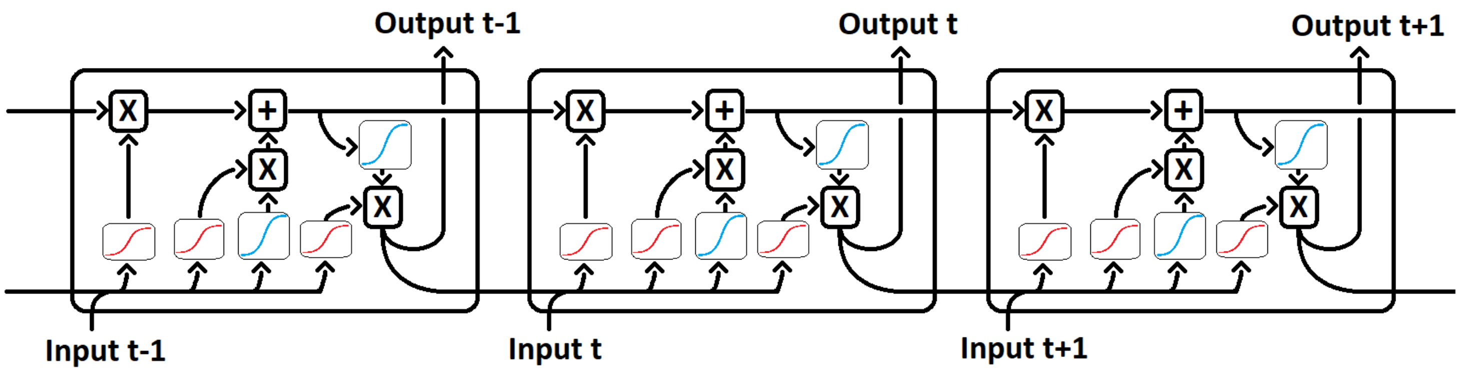


**Figure 1.2:** A visualization of an LSTM network layer.

One of the main differences between a standard RNN and LSTM is the cell state which lets the gradient flow unchanged. It is because of this that the gradient is more resistant to vanishing. The cell state is the straight line going through all of the cells, being updated with point-wise multiplication and addition in each of the cells.

The forget gate updates the cell state by deciding which information to forget, leaving only relevant information to aid prediction. This is why point-wise multiplication is used. The input gate updates the cell state after the forget gate, adding relevant information from the current step. This is done by point-wise addition. Finally, the output gate calculates the value used for the output and hidden state [11].

## 1.3 Stock Market

A stock can be seen as ownership in a company. Depending on how people value this company, the price of the stock naturally increases or decreases. Stocks can therefore be considered as a product of supply and demand. The reasons for valuing a stock at a certain price could be different, but one of the common ways are looking at the income and expenses of a company. With this said, speculation of a company can drive prices both up and down. This is why we can see success in an LSTM network with news sentiment analysis incorporated as performed by Y. Guo [13]. This is an example relating to fundamental analysis rather than technical analysis. These two concepts will be explained in the two following sections.

To illustrate an example of a good trade would be, the purchase of an undervalued stock which later is sold when the stock is overvalued for a larger sum than it was bought for. A stock is considered undervalued when the market price is presumed to be below the true value of the stock. An overvalued stock is in the same way considered overvalued when the price is presumed to be above the true value. This can happen for example if the company's share holders are in need of money and are forced to sell their stocks to a price which is lower than the true value. If at a later time there are a lot of buyers because of for example a well known investor's recommendation, the stock can become overvalued. These kinds of scenarios are sought after when investing. They can happen on a small scale as well as a larger scale. These scenarios can be realized both using fundamental analysis, and technical analysis. Successfully managing to analyze the stock market, meaning knowing whether a stock is undervalued or overvalued, could result in large capital gain if the analysis turns out to be a correct one.

When buying and selling stocks, they are generally presented with a chosen trading span. The stocks are then included with the information of what the lowest and highest traded value was during the span, the opening and closing value as well as the amount of trades made during the span. This information is provided when obtaining stock market data with the notation OHLCV, which stands for Open, High, Low, Close and Volume.

### 1.3.1 Technical Analysis

The main idea of technical analysis is that historical trading activity impacts future trading activity. The trading discipline analyzes historical data in combination with data processing in various ways to identify a trade or an opportunity. These data processing methods often return a value of indication, which are referred to as financial indicators. One of the most common indicators used is the momentum indicator RSI, introduced as early as 1978 by J. W. Wilder [14]. The indicators are used and analyzed to, in turn, decide on an action whether to buy or sell a certain stock. This method is generally used for shorter time-spans compared to the contrasting method of fundamental analysis, see the following section 1.3.2. Technical analysis therefore seems to be a more fitting method for an intra-day trading application where a deep learning model can be created in an attempt to replicate this strategy.

### 1.3.2 Fundamental Analysis

The stock market is a place where people buy and sell their stocks. The prices of these stocks are based on many factors. Fundamental analysis is a method of trying to estimate this price as it should be, or rather estimate the so-called intrinsic value. The way this is achieved is based on financial statements such as annual reports but also industry trends and external influences as the price is governed not only by the financial state of a company but also the possibility for future growth and the overall public view of the specific industry. The assumption made is then that the market price does not necessarily reflect the true value of the stock, and furthermore that the true value will be reflected on the market price given enough time. For this reason, fundamental analysis is often used when investing for longer periods of time.

## 1.4 Problem Definition

A correct prediction of the stock market is a desirable achievement as the outcome could lead to significant financial profit. This thesis aims to investigate further approaches to improve state of the art machine learning prediction. This is done by introducing a large amount of financial indicators to a machine learning regression model using LSTM. The objectives of this thesis more specifically are:

- Understand how an LSTM model with heavy data information using a large amount of indicators behaves with different parameters.
- Identify whether the introduction of a large amount of indicators aid prediction, or whether the indicators simply contribute to over-fitting of the data and act as noise.
- Investigate how the influence of time and the use of the derivative impact prediction.
- Explore a solution to the demanding computational requirement that is needed by the project.

The project will be created with a focus on performance as suggested by the last objective. This will be achieved using a parallel distributed system where computations will be divided between multiple workers called nodes.

## 1.5 Limitations

In comparison to previous and related work, this thesis examines the behaviour and accuracy of a model using a larger amount of indicators. Achieving the highest possible accuracy is desirable, however this assumes that the increased amount of indicators can provide the model with relevant data to enable this higher accuracy. There is a possibility that this is not the case.

Root Mean Squared Error (RMSE) will be used to calculate the error of the prediction:

$$RMSE = \sqrt{1/n \sum e^2}$$

where $n$ is the number of predictions made and $e$ is the error, or difference, of the prediction from the true value. The main reason for having this is to simply compare the different models. The same error calculation will be used on all different models for a fair comparison. Other methods such as Mean Absolute Error, Mean Squared Error and Mean Absolute Percentage Error are available, although RMSE is still commonly used in recent work for these types of applications and is therefore suited for this work as well [15]–[18].

Directional accuracy will be included as well for comparison of the different models and is defined as how often the prediction is in the correct direction. This can then be viewed or translated into a buy or sell signal. Although, as the models are trained to specifically have a lower RMSE with a regression model, a lower directional accuracy compared to related work is to be expected. The evaluation method is only included as a point of interest and aids as another dimension in discussion. Furthermore, it is difficult to compare the RMSE values between stocks as they depend greatly on the stock price and these vary significantly. This is another reason for having introduced the directional accuracy.

This thesis will use the Advanced Micro Devices (AMD) stock to train and evaluate the different models. Comparison of different models using the same data is made to enable discussions and evaluations between the models. The choice of AMD was made arbitrarily as using one stock for training was desired to better understand the specific patterns of the specific stock as these are assumed not to be exactly the same for different stocks. However, other stocks will be available to confirm that the trends and results we see for the different models on the AMD stock are the same. These stocks are the most traded stocks in their specific category as specified by Yahoo finance. They are all American stocks and they are all part of the same high-volume stocks of over 500 000 USD [19]. The availability of other stocks also serve as a possibility to make a general model as well which enables the creation of more training data to the model if needed. This could be important to deal with over-fitting.

Profit analysis will not be a part of this thesis, as the main goal is to improve and analyze the reason for the prediction accuracies.

# 2
# Theory

This chapter will introduce a few concepts which are necessary to understand the reasoning behind the choices made for the methods used in this project as well as to understand the discussions and observations regarding the results that the work has contributed with. The main focus will be an overview of the different indicators used as well as the model parameters and their impact on training. Details regarding the indicators used in the project, including the indicator for derivative and time of day, as well as a deeper explanation on over-fitting will be included. In addition, a graph of the data used in the project will be provided to further clarify what data the machine learning model was trained with.

To illustrate the complexity of the problem, the stock price and all of the indicators used in the project is shown below in figure 2.1. The horizontal axis represents time where each data point is 15 minutes apart, ultimately displaying a total span of 5 days. The vertical axis represents the normalized values of the stock price and indicators. These indicators in addition to the stock market data are used as input to the algorithm to predict the next stock price. The complete list of these indicators is included in Appendix A.

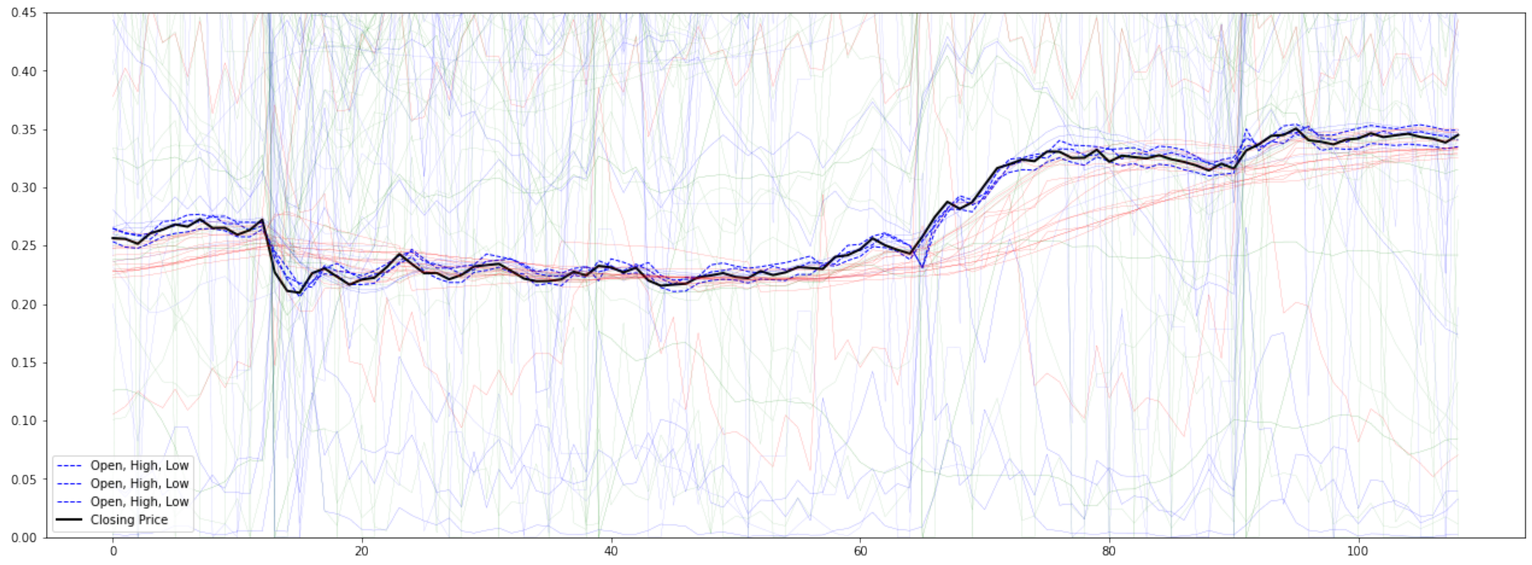


**Figure 2.1:** A visualization of the different indicators and their normalized values.

Making a prediction for a person given this information might be quite overwhelming. Although for a neural network, it is not as difficult. The main difficulty therefore lies in creating and optimizing the neural network in order to do a prediction well enough and understand the reasons for why the prediction is the way that it is.

## 2.1 Indicators

More information is generally better, however, the information should correlate to the data. If it does not, it is considered noise. In a machine learning application, noise is undesirable as it can contribute to over-fitting. The indicators in general were assumed to be able to contribute to a more accurate prediction and not just be noise or random numbers. This assumption is a similar one as the assumption of technical analysis and its trading discipline where past data is assumed to impact current trading patterns. See section 1.3.1 for a description of technical analysis.

Principally, the indicators were used because they are also used by other day traders. This means in turn that the predictions are affected by the buy and sell patterns by other day traders. This is why a better model accuracy is expected when using the indicators as the machine learning model is assumed to see these patters as well which would aid prediction. The most common values for the different indicator spans were used in this project as they are naturally the most common among traders as well. These spans are generally calculated using 14 or 10 values back from the OHLCV data. The longest ones, however, go to 88 values back, meaning over three days of data is required for some indicators to be processed. Many of these indicators are commonly used for day-to-day basis calculations, although all of the used indicators were scaled for use in 15-minute intervals.

Further details regarding the exact calculations and how the financial indicators work, except for a general overview, will not be presented. The main importance is that they are widely used for technical analysis, thereafter the algorithm will decide how to weigh them without any human input. The general overview of the indicators is given in table 2.1.

### 2.1.1 Derivative and Time

An aspect of using the derivative instead of only the actual value is that there is a larger relative difference of derivated data compared to non-derivated data. For example the derivative value of 0 to 0.121 compared to the price of 121 to 121.121. This could be a beneficial way of presenting the data to the algorithm as an emphasis is put on price fluctuations rather than only price.

Although not generally considered a technical indicator, the project also investigates the effect of time on the stock market. One hypothesis is that people trade differently during the morning or afternoon of the market.

## 2.2 Over-fitting

In the creation of a predictive model, there is a concept called over-fitting. Over-fitting is a form of modeling error that needs to be explained in order to understand the choices made in this work, such as the creation of the general model. The concept essentially means that the algorithm is too closely adapted to the training data. So close that even what would be considered noise is trained on and remembered for future predictions. This can happen when we let the algorithm train for a long time or use a complex model which is trained on a small amount of data. This leads to a highly accurate algorithm, but only for the training data. The result of which being that the algorithm loses accuracy for future data.

The problem of over-fitting within neural networks is generally a common one but there are a few methods of dealing with this issue. One way, is by stopping training as soon as over-fitting has begun. An indication of this is seen when the validation data used in training begins to decrease rather than increase in accuracy, meaning that even though that accuracy increases for the data as seen by the model during training, accuracy decreases for data which comes afterwards. This is one reason for having the validation data.

Over-fitting is expected to be a common issue for this specific work as there are a lot of indicators that could have a poor correlation to the output data and for this reason be seen as noise. Noise to this extent is undesirable as there is no improvement of the model that can be obtained by having it.

A visualization of over-fitting is seen in figure 2.2. This figure was created by increasing the validation set and by removing the dropout-layers introduced in the model to reduce over-fitting. The concept of dropout is explained further in section 2.3.3. The amount of LSTM cells was set to 50 to visualize the concept and it is observed that the validation loss increases after about 120 epochs in this specific example. The RMSE after 400 epochs was 7.08, compared to where the lowest validation loss and least over-fitting was obtained at 3.40 RMSE. This is more than double the error.

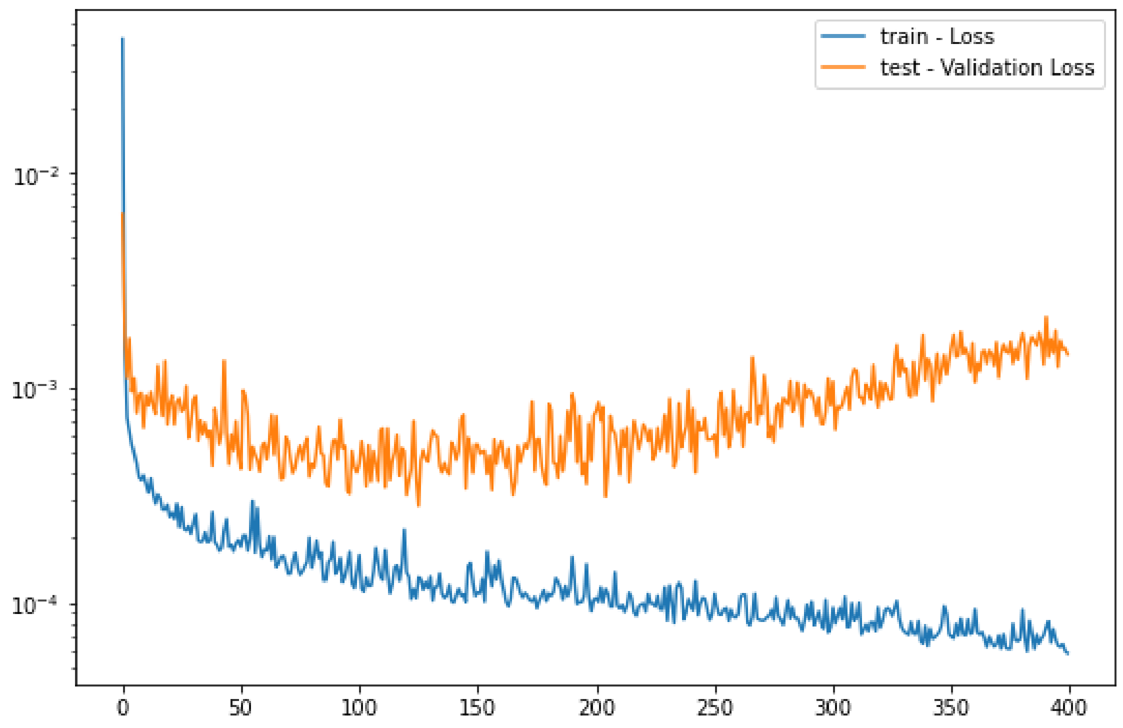


**Figure 2.2:** The effect of over-fitting on OHLC data with all indicators. Trained for 400 epochs.

To prevent this, the model used in this thesis was introduced with dropout layers. Furthermore, to verify that there was enough data to train on, a general model was created, which is an extension of the normal model that uses 87 stocks for training instead of using only one. The general model had the advantage of having more data. Although, more data from the same stock would ideally be better for prediction as stocks do not necessarily correlate well enough with each other to improve accuracy. The goal of the general model was to reduce over-fitting as using more training data lessens this effect. Further discussion of the general model will be made in section 3.4.

## 2.3 Model Parameters

The following section will discuss the various parameters of the machine learning model and their relation to the project as well as the impact they have on the model, its input and ultimately how the predictions are affected.

### 2.3.1 Learning Rate and Batch Size

A lower learning rate is generally better for improving the accuracy of the model as weights are updated more carefully. Using a too small learning rate however, lengthens training time as well as increases the possibility for training to get stuck in a local minima. The same problem happens for larger batch sizes where using a too large batch size could lead to the algorithm getting stuck in local minimas [20]. One of the advantages of using larger batch sizes is the shorter training time. Therefore, a trade-off between training time and accuracy has to be made. Another aspect of this is the ability to change learning rate or batch size during training. There are many advantages to such an approach, however, it contributes to an increase in model complexity.

### 2.3.2 Layers

A model could be made using only one layer. This would lead to the dependencies between the indicators to be viewed as much simpler compared to using multiple layers. More layers will therefore enable more complex patterns to be found. The assumption is that one layer therefore is insufficient to create a well performing model.

Having too many layers is problematic as well. If more layers are used than necessary, the algorithm is introduced to a problem which is assumed to be more complicated. If the complexity of the problem is simple in relation to this, it ultimately leads to over-fitting. A way of dealing with over-fitting is therefore by not using more layers than necessary to match the complexity of the problem.

### 2.3.3 Dropout

Dropout is a way of avoiding over-fitting by cancelling out some nodes in the model, or rather from the name, letting them be "dropped out". This effectively removes them from the network by setting their value to zero which in turn makes the algorithm forced to weigh features less dependently. This is done randomly to ensure that training would not only effect specific nodes and features but the model as a whole. The effect of this makes the model harder to over-fit as the model constantly changes. Since we have a very large amount of features and not a very large amount of data relative to this, using dropout is a suitable choice. When introducing dropout, it is done so as a layer. The layer then takes the output from the previous layer and applies dropout before sending it to the next one.

### 2.3.4 Loss Function

The loss function of the model decides how the algorithm updates its weights. There are a few to choose from such as mean squared error, mean absolute error and mean absolute percentage error as a regression model is used. The most intuitive choice of loss function was mean squared error as the Root Mean Squared Error (RMSE) was used to evaluate the model. The reasoning being that using a loss function which is closely related to the evaluation method would optimize for higher model accuracy. There is still the possibility that using a different loss function could have a positive impact on directional accuracy.

## 2.4 Features

In addition to OHLCV data, table 2.1 describes an indicative set of indicators used in the project. The sources are hyperlinks which contain a more detailed description of the indicators. The full list of all indicators used is more complete and included in Appendix A. A thing to note is that the indicators are introduced to the algorithm as features in the same way as the OHLCV data. A reference to the features of the project therefore means all indicators of the project a well as OHLCV data. The indicators are varying in category as a wide range of indicators was attempted to be introduced.

**Table 2.1:** A general overview of the project's indicators and their descriptions

| Indicator | Description | Source / Reference |
|---|---|---|
| Volume | Based on the volume of traded stock and included in the data collection. | Investopedia - Volume |
| Derivative | The difference between the current closing price and the previous closing price. | Manually coded using np.diff |
| Bollinger Bands (BBANDS) | Trendlines, or bands, which are two standard deviations away from the Simple Moving Average. | Investopedia - Bollingerbands |
| Moving Averages (MA, SMA, EMA, etc.) | The average of prices from the last numbers of values. These help smooth the price to show an average price representation. | Investopedia - Moving Average |
| Transforms (Hilbert Transform, Fisher Transform) | A form of digital signal processing which helps indicate turning points and trends. | Investopedia - Fisher Transform + [21] |
| Parabolic Stop and Reverse (SAR) | Indicates the price direction and also the changing of the price direction. | Investopedia - Parabolic SAR |
| Average Directional Index (ADX) | Determines the strength of a trend. | Investopedia - ADX |
| Oscillators (Rate of Change, Money Flow, Relative Strength and others) | Returns values close to the stock price which is used to determine whether a position is oversold or overbought. | Investopedia - Oscillators + [14] |
| Moving Average Convergence Divergence (MACD) | Shows a relationship between two moving averages, this indicates buy and sell signals. | Investopedia - MACD |
| Momentum (Many indicators fall in this category such as Relative Strength Index, True Strength Index, Inertia, etc) | Describes the rate of acceleration of the price and ideally indicating when it is possible to follow the trend. | Investopedia - Momentum |

| | | |
|---|---|---|
| Sentiment Indicator (BRAR) | Represents the current beliefs and feelings of traders which is in turn is assumed to affect future market behaviour. | Investopedia - Sentiment Indicator |
| On Balance Volume (OBV) | A Momentum indicator type which incorporates volume in its calculation to predict if there is a change of price on the way. | Investopedia - On Balance Volume |
| Volatility (MASS_INDEX, RVI, BBANDS, etc.) | A measure of fluctuation. A high volatility therefore should indicate a higher risk and vice versa. Indicators such as Bollinger Bands, Mass Index, Relative Volatility Index, etc. fall under this category. | Investopedia - Volatility |
| Sinewave (EBSW) | An oscillating waveform used to identify oscillating patterns. The project uses the specific indicator called "even better sinewave". | Investopedia - Sinewave |
| Elder-Ray Index (ERI) | Measures the amount of pressure of buying and selling, effectively indicating trends. | Investopedia - Elder Ray |
| Ease of Movement (EOM) | Presents a combined value for momentum and also volume, in theory indicating the strength of a trend. | Investopedia - Ease Of Movement |
| Regression (LINREG, TIME_SERIES_FORECAST) | Establishes a relationship between different values. In the case of Linear regression this creates a line which can be used to estimate future prices in a linear fashion. | Investopedia - Regression |
| Ulcer Index (UI) | A measure of a specific volatility, but only for downside risk. The reasoning behind it is that an upward movement is not as stress-inducing for traders as a downward movement. | Investopedia - Ulcer Index |
| Negative and Positive Volume Index (NVI, PVI) | Cumulative indicators which incorporate price and volume in their calculation to show how price movement is influenced by the traded volume. | Investopedia - Negative Volume Index |

# 3
# Method

The creation of the different models was made using the TensorFlow - Keras library in Python [22]. Financial indicators were then incorporated into the machine learning model in an attempt to further increase accuracy. These indicators were first normalized and thereafter used as inputs to the LSTM model as additional features. A visualization of the data processing as well as how the data and features are structured can be seen in Appendix A. Most of these indicators were obtained using the Pandas TA and TALib libraries [23], [24]. A list of all indicators used in the project is also available in Appendix A. It is worth noting that that the indicators required a certain amount of historical data before returning a value. This means that no training can be made for the first periods of the data where some of the indicators are not yet calculated. As different indicators requiring more historical data for their calculation were introduced and added to the project, the length of data that could not be used for training increased, in total excluding three days worth of data.

The indicators were chosen with regards to minimizing the overlap of information they give. This is the reason for not including the same indicators with many different spans. With this in mind, a focus was set on trying to have a plentiful amount of various indicators. The increased complexity of the many different indicators therefore had to be balanced with the problem of over-fitting, see section 2.2.

Investopedia was used for technical indicator information as it is a well known trading website with the intention of attempting to replicate and exploit the patterns used by day traders. Investopedia has 44 million monthly readers. Because of this large amount the content provided by Investopedia is assumed to have impacted the stock market and its traders. The use of financial indicators as described and provided by Investopedia will therefore be ideal for this work [25].

## 3.1 Code Structure

The code was structure in the following way:

- Various imports such as TALib, Pandas, Tensorflow (Keras), etc.
- Creation of variables and loading of stock data
- Stock data gathering (only used once every month to append more data to the larger data-file)
- Data processing, Creation of model input data for one stock
- Data processing, Creation of model input data for all stocks
- Creation of LSTM model
- Training of model
- Evaluation

In addition to this, multiple other notebooks were made to facilitate a form of automated training, evaluation and plotting of various results and visualizations. The different models within these notebooks were thereafter used as workload tasks, being executed in parallel using a mininet simulation to enable the project to complete faster given access to multiple workers.

## 3.2 Data Collection

Only one stock was used to train the models. The predictions created from the trained models were later compared to the true values of the stock price. 100 stocks were obtained from the same market. These stocks were chosen with regards to similarity. With high and similar volumes of trade the impact of technical analysis is assumed to be similar for theses stocks. The availability of stocks will assist in further analysis of the algorithm as the project is not limited to a specific stock as well as enabling creation of more training data if necessary. The selected stock could then be chosen either arbitrarily or specifically. The stock chosen in this project was AMD. There was no specific reason for this particular choice, but a choice of stock had to be made in order to progress and evaluate the models.

Data collection was done periodically every month where new values were appended to the larger data-set. Over 8 months of data was obtained and used for all of the model training excluding the performance measures which were made using over 9 months of data as the project continued and more data was available. The data was collected using a Yahoo! API. It was then divided into two sets with a 70/30 split, where the first 70% of the data was trained on, and the last 30% was used to test and evaluate the different models. Additionally, a validation set from the training data of 10% was created to reduce the effect of over-fitting. The reason for the data split was to emulate a real world scenario where a prediction can only be based on previous data.

All of the data used in the project was obtained through the open source API using data from Yahoo! Finance [19]. 6 columns of data were given, these were Adjusted Close, Close, High, Low, Open and Volume. The earliest date which was obtained was 2021-12-16 as the API only allowed data to be collected from 60 days back. The storing, loading and overall processing of data was made using Pandas. After data gathering the new data was appended to the main csv-file. A total of almost 3 000 000 data points were obtained during the span of the project.

## 3.3 Normalization

Normalizing the data improves convergence and training times as well as eliminates the problem of having too large gradients [9], [26]. The data was therefore scaled to the range of 0 to 1 using the MinMaxScaler imported from sklearn. Having different scaling on the features causes the algorithm to prioritize the scaling with higher values. This is not desirable. Which is why normalization must be applied.

The data was normalized column to column. Meaning that all features were normalized independently. This was done as normalizing between features could make some features insignificant in comparison. An example is the indicator Volume, which can have values of over 5 000 000, compared to the derivative, which is no more than 8 for the AMD stock. Normalizing these two features to a range of 0 to 1 would mean derivative values consistently get close to 0 while volume stays close to 1. This needs to be avoided.

A reason for normalizing some features together is that the relative values for Open, High, Low and Close are important and may be lost during feature to feature normalization. This is the exception. Normalizing these independently would mean that the differences in price are potentially removed by the normalization. Rather than normalizing these in a feature to feature fashion, these values are normalized together, effectively keeping the relative differences. The effect this has on the normalized values can be seen in Appendix A. Note that the "Low" values are larger than the "High" and "Open" values with column to column normalization. This is not desirable.

For the model with all of the stocks, they were normalized in the same way as the AMD stock and thereafter used for training. There was no overlap in training between the stocks. Meaning the data that was appended to the training set never contained two different stocks in the same training set. If it did the normalized value could jump from 0 to 1, or 1 to 0 seemingly randomly which would not have been useful information to the algorithm.

## 3.4 Creation of a Larger Model

A larger model using all available stocks for training was attempted. This was done in order to make a more general model and solving the issue of having a low amount of data. The model is therefore referred to as the general model. The stocks were divided into training and testing data in the same way as the AMD stock. The overlap in data between the stocks was avoided when creating the training and testing data as that would have contributed to a training error.

The reason for not using all 100 stocks from the data collection is the existence of NAN-values in the data. This could be because of liquidation, company merging, etc. Instead of manually going through all 100 stocks, the ones which had NAN-values were simply excluded from the data. 87 stocks in total were used for training. This was considered more than enough data as it is in comparison to one stock effectively 87 times as much data. Or more specifically, over 40 years of data compared to a bit more than half a year.

There is a possibility that better accuracy values could be obtained using even fewer stocks than this, but ideally more data from a single stock would be used as combining stocks could potentially lead to a negative impact on prediction. The reason for this is that different stocks can have different patters. The assumption is however that multiple stocks also exhibit similar patterns which relate to each other. The result of the general model should nevertheless indicate whether we have a problem of too small of a data sample for one stock. The general model was later evaluated on the AMD stock in the same way as all previous models and finally compared to the other models as well.

## 3.5 Network Topology

This project uses a multi-layered approach which was optimized to 3 stacks of LSTM cells. This is a similar topology to other work in the same field of stock market prediction by Sunny et. al. and Wang et al. [4], [27].

A somewhat recently published paper from 2018 suggests using the formula for the specific application of stock price forecasting:

$$m = \sqrt{n + l} + \alpha;\ m = \log_2 n;\ m = \sqrt{nl},$$

$m$ being the number of hidden layer nodes, $n$ being the number of input layer nodes, $l$ being the number of output layer nodes and $\alpha$ being a constant between 1 and 10 [4]. The standard three-layered model used in the project was based on this equation and later optimized. The model was then made to a starting point for the other models.

The model which many of the project's calculations are based upon is the following:

- Input layer, 1500 LSTM cells
- Dropout layer with 10% dropout
- Hidden layer, 1500 LSTM cells
- Dropout layer with 10% dropout
- Hidden layer, 1500 LSTM cells
- Dropout layer with 10% dropout
- Output layer, Dense

The dropout layer was introduced after every LSTM layer. Dropout did not have to be symmetrical. It could be done on one layer, some of them, or all of them. A symmetrical approach was decided on for simplicity reasons. Tests were made with more and less dropout. Both of these tests showed some promise and having a minor increase in accuracy, however, the model had to be made general to ensure comparison between the created models. Increasing dropout too much could hurt performance and make the different models incomparable while no dropout could lead to over-fitting. A large model of this size would be expected to use some dropout or another regularizing method such as batch normalization [28]. For this reason, dropout was kept at 10%.

The chosen optimizer for this project was the Adam optimizer. The Adam optimizer was used as it is an efficient optimizer with a low memory requirement [29].

## 3.6 Model Parameters

The following section will explain the choices for the specific model parameters and how these have been obtained. The parameters and features explored in more detail will be presented in the result section. A Time Series Forecast (TSF) based on a linear regression from TA-lib will be included for comparison purposes with the created models [24].

### 3.6.1 Learning Rate and Batch Size

For the sake of comparison and ease of replication, the learning rate was held constant at $10^{-4}$ for all configurations with the sole exception being the training on all available stocks where the learning rate was set to $5 * 10^{-5}$. This was made as an attempt to combat the larger batch size which had to be increased to accelerate training speed. The reason for not using a variable learning rate or variable batch size was so that multiple models could be trained and later compared as well with minimal changes in training environment. A variable learning rate would contribute to another dimension in training. Therefore both a constant learning rate and a constant batch size were used for all the models.

A learning rate of $10^{-4}$ was deemed fitting as training times had to be kept reasonable as well as meaningful values of accuracy had to be obtained. A batch size of 32 was used for the models trained on one stock. The general model used a larger batch size of 512 to keep training times reasonable. With a batch size of 32, each epoch took about 4 minutes to train for the general model. This was not acceptable as the project used callbacks based on epochs which were introduced to limit over-fitting. A larger batch size of 512 was therefore used which reduced epoch times to under a minute.

The advantage of the general model was the large amount of training data. The problem of getting stuck in a local minima was not expected to be as large of an issue because of this. This meant that using an increased batch size compared to the single stock models was a preferable option. Generally, when using a larger batch size, a degradation in accuracy is expected. A lower learning rate was therefore introduced for the general model. These changes were made to keep both of these models reasonably comparable.

### 3.6.2 Layers

For this project, three layers was used. The exception being isolation tests of layers specifically testing fewer or more layers and their impact on accuracy in order to optimize the algorithm. The optimization clearly indicated three layers to perform the best in this application. Another aspect of this is the expectation of a more complex problem to be realized with a more complex model. The assumption being that a single stacked layer is incapable of realizing this complexity.

### 3.6.3 Dropout

A dropout of 10% was used in this project. Dropout values of up to 50% as well as no dropout at all was investigated as well. However 10% seemed to be a good trade-off for comparing the different models, as too high and too low dropout values were purposely avoided.

### 3.6.4 Window Size

One of the challenges of the project was deciding upon a time window for the calculation of financial indicators. The time window is the length of the input sequence of historical data introduced to the algorithm during training and evaluation. During testing and optimizations of the algorithm, larger time windows of up to 50 were tested in different model configurations. The results of these tests are presented in table 4.2. The specific lengths were 1, 5, 10 and 50, and they are referred to as "look-back" since that is the available information to the model during training.

In the same way as the method of number of LSTM cells was treated, the most promising look-back was used for comparing the other models and their accuracies to keep data values comparable and to keep the project consistent. This ended up being a look-back of 10 despite having other look-back values such as 1 or 50 showing a larger promise in terms of accuracy. 10 was chosen in order to keep a look-back of a larger length than 1 to enable the algorithm to see patterns for past data as well. However, A longer look-back of 50 would imply longer training times for the larger models as well as make the algorithm easier to over-fit, this was undesirable.

Since we are dealing with 15 minute intervals, a look-back of 50 translates to a bit over 8 hours of information for a prediction 15 minutes ahead. This seems to be a reasonable amount and a larger look-back was therefore not tested. An additional benefit to this is the increased training time which is conveniently avoided.

### 3.6.5 Number of Cells

The specific number of LSTM cells examined was 10, 50, 100, 500, 1000 and 1500. This was done both as a stacked neural network and as a single stack during model optimization to verify that the stacked neural network was the optimal approach. This is the reason for why the stacked network was used in all calculations as it generally performed the best. Results of these tests are presented in table 4.3. The results are created for comparative purposes to indicate how a large model would behave in this scenario. As discussed in the theory section, to achieve a more optimal model, training would have to be done using a decreasing variable learning rate and preferably an even larger data set.

### 3.6.6 Model Callbacks

Early stopping in combination with a model checkpoint was used to stop training once the best validation loss was obtained with these parameters. Early stopping is a common way of training a deep learning algorithm and was also used for this project. The patience was originally set to 50, although a higher patience of 100 was later used for all calculations as the model in some cases was observed to have found a better solution after 50 epochs of training without improvement. The model could then train for a longer time period so that the training and validation loss could be perceived for longer periods. A high number of epochs as patience was therefore allowed in order to obtain the lowest validation loss across all of the different models. The specific epoch with the lowest validation loss was saved for use for each model and later compared to the other models. The callbacks were created with the early stopping function and the model checkpoint function from the Keras library.

## 3.7 Exploring the Impact of the Stock Derivative and Time

The specific indicator introduced as the derivative is calculated by taking the difference in price between two data points, t[i] - t[i-1], and creating a new feature column. The assumption being that the LSTM algorithm will react better to derivated normalized data rather than only normalized data.

An analysis of time was made. This was investigated in three ways. The reason for including several methods specifically targeting this indicator was to better understand the impact of time on the stock market. There are different advantages and disadvantages to the different methods.

First, time of day as an indicator was implemented in the project in a similar way to the implementation of the derivative. An artificial creation of a column named "Time" was made using a counter from 0 with small increments of 1 which reset at the start of every day. This was done in order to make the algorithm aware of the time of trade. Isolation tests of these two features were conducted. These tests were made using only the stock closing price as well as the the complete OHLC data with the specific additions of either the derivative or time as parameters.

The second method divided the data into morning and evening by looking at the first and last two hours of each day. The model was then trained on as well as evaluated on these morning and evening times. The downside of this method was that the data sample got greatly reduced. By isolating the first and last two hours of the day, the training data only uses a bit over 30% of the total amount of available data. A way to overcome this data scarcity problem was the elimination of the other indicators in an attempt to reduce the problem complexity and isolate the specific look at time and its impact on the stock market.

Finally the last method was to train the model on all available training and testing data data for one stock, although only evaluating it on the evening or morning data. This would eliminate the problem of using less data for training although still examine the impact of prediction for different times of the day. All of the results obtained from these three methods are shown in the results section.

## 3.8 Hardware Specifications and Performance

The different models were trained on a Ryzen 7 5080H with 16 Gigabytes of RAM. An RTX 3070 laptop GPU was also used for the heaviest workloads. Training times were approximately 20 minutes per model with the exception of the general model which required approximately 16 hours of training. This limited the ability for optimizing the general model if required. For this reason the general model was made with optimizations based on previous models trained on only one stock.

### 3.8.1 Distribution

To improve the performance of the long sequential execution time of the different models which had to be run, a method of enabling more hardware resources to be used was also created. The next challenge was then making a parallelization of the project in a way which would be balanced and divide the workload effectively between the available resources. The distribution was created and simulated over mininet [30]. The workload was divided into 24 computationally demanding sections called tasks. The code regarding data collection and result evaluation was excluded from this distribution as the execution time of these was negligible in comparison to the model training time. The main tasks which could be executed in parallel are shown in the results section in figure 4.6.

The time of the workloads was measured using the time module in python. These times are rough estimates of the computational workload as it is heavily dependant on whether the task finds a better solution after 100 epochs. Furthermore, the tasks were executed using a low power energy saving configuration, optimized for cooler temperatures and higher energy efficiency. The relative differences for the computational demands of the tasks are kept as they were all executed with the same energy saving configuration. The same hardware is capable of faster training by increasing the amount of power to the hardware.

The available resources, from now on referred to as nodes, were all introduced to the different tasks in an executable script. A command was then sent to the nodes which executed specific tasks. When a task on a node was finished another task was issued, always keeping the node busy as long as there were tasks to be executed. The number of available nodes was introduced as a parameter. Meaning that the distribution was able to utilize any number of resources chosen.

The tasks were sorted and executed in their computational demand order, meaning the tasks taking the longest to execute were executed first, this reduced the time execution compared to a random order scheduling as the longest tasks would not continue executing at the end of the task schedule. At most, the execution time using longest processing time first scheduling will stay at 4/3 times the most optimal partitioning as proven by Xin Xiao [31]. This enabled an efficient task execution to be made with no need for a synchronous execution. Timing the tasks was one way for sorting them in computational requirement. Since other computers can have different hardware, there is a possibility that the tasks could be executed in different speeds and sorted in a slightly different order. Since the hardware used to execute and time the tasks was assumed to be the same for all of the nodes, the timing of the tasks on the specific hardware as specified in 3.8 was ideal as it correctly simulated the task execution for all of the nodes.

In contrast to this aspect, an improvement of the suggested task distribution with a distribution of nodes which are not uniform in hardware would be to start executing the heaviest tasks on the nodes with the fastest hardware. This would require a sorting of the nodes as well. The general model is the heaviest workload. Given a non-uniform distribution of resources, the general model is suited for the fastest hardware. If necessary, the largest time consuming general model could also be executed in parallel on multiple nodes instead. This could be done using a form of either synchronous or asynchronous distributed training.

The tasks were executed using python script files using the mininet python API. The commands were sent to the shell of the simulated worker, which executed the .py script file. The python script files included machine learning, creation of model, and an output of RMSE accuracy values as well as the saving of the model. The results of the nodes with the error analysis values were all stored in a separate .txt files, which were later easily accessible.

The code was structured in the following way:

- Various Imports
- Choosing the Number of Tasks and Nodes
- Creation of Network
- Connectivity Verification Test
- Initialization of Task Distribution Algorithm
- Waiting For Last Distributed Tasks to Finish
- Shutting Down of Network

# 4 Results

The following section will present the findings of the project and discuss them. Tables will be presented showing the impact of various models with different parameters and input data. The models used in all of these tables are the same standard stacked three layered model which performed well for this problem. During optimization of this model, different amount of layers were examined, however there was no indication of further accuracy improvements for a larger amount of hidden layers. Some models using fewer layers showed promise, however three layers were kept to ensure that more complex correlations between features are possible. All of the directional accuracy values are given in % and the regression error is given in RMSE. A visual graph is included comparing the predictions with the actual price values and a time series forecast of the stock price, see figure 4.1.

The models were all evaluated by comparing the RMSE values. These are calculated by comparing the stock values with their respective prediction. Directional accuracy was calculated by measuring whether the direction of the stock price is the same as the direction of the prediction. The Time Series Forecast prediction was evaluated in this way as well to enable a form of comparison.

All of the predictions and model evaluation are made on the AMD stock. This is also true for the general model as a fair comparison was aimed to be made between them, even though similar RMSE values are expected for the other stocks using the general model.

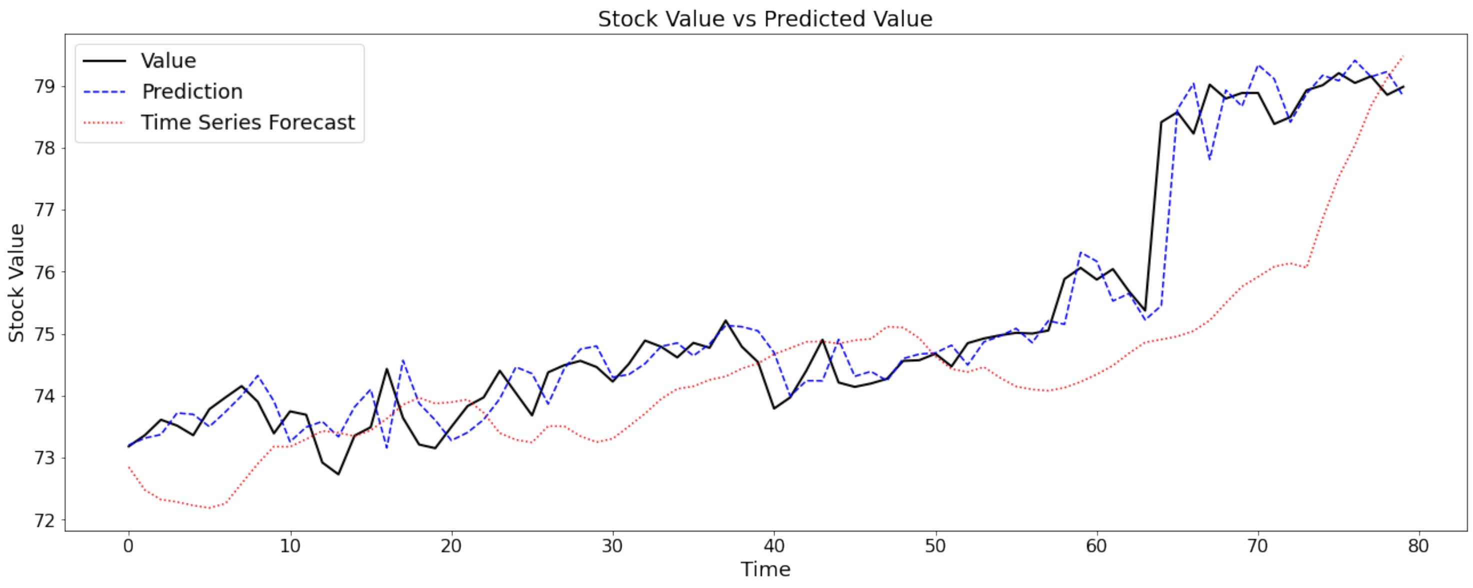


**Figure 4.1:** A comparison of the model predictions with a time series forecast and the actual values during a span of three days.

The first table, table 4.1, shows how the algorithm behaves using a different amount of LSTM cells. The number of LSTM cells examined is up to 1500. This is due to hardware limitations as larger models are exponentially harder to train because of the stacked neural network. Regarding the results there seems to be a trend which indicates that the use of a larger amount of LSTM cells impact predictions positively and lower the RMSE. An exception to this trend is the 100 LSTM cells model which seems to have a lower RMSE compared to the larger ones. This can be the effect of reduced over-fitting. In contrast to this trend, the directional accuracy seems to be declining for larger models, peaking in the 50 LSTM cells model.

**Table 4.1:** An investigation of the number of LSTM cells for a three-layered symmetrical model. All indicators are present in these calculations.

| Number of LSTM Cells | RMSE | Directional Accuracy (%) |
|---|---|---|
| 10 | 1.82 | 52.17 |
| 50 | 1.28 | 52.47 |
| 100 | 0.86 | 51.63 |
| 500 | 1.14 | 48.21 |
| 1000 | 0.96 | 48.29 |
| 1500 | 0.90 | 48.06 |

The second table, table 4.2, investigates how the model reacts to different look-back values of 1, 5, 10 and 50. These two tables decided the parameters for the models that follow. The best results were acquired using a look-back of 1, both in directional accuracy and RMSE, although the choice of using a larger look-back was made since introducing the algorithm and the different indicators to past data was deemed relevant.

**Table 4.2:** Look-back comparison. All indicators are present in these calculations.

| Lookback | RMSE | Directional Accuracy (%) |
|---|---|---|
| 1 | 0.57 | 52.87 |
| 5 | 0.81 | 48.94 |
| 10 | 0.90 | 48.06 |
| 50 | 0.77 | 50.67 |

The third table, table 4.3, shows how the different models react to an increasing amount of indicators up to a total of 97 features. The specific indicators "Derivative" and "Time" are tested separately. The expectation was to see improvements with the addition of features, however this improvement was not perceived in terms of RMSE accuracy, although it was arguably perceived in directional accuracy. The most accurate model in this table in terms of RMSE is the smallest model using only the closing price for training with an RMSE of 0.53.

This table furthermore includes the larger general model which is trained on a much larger training set and a comparison of the different models to a Time Series Forecast (TSF). TSF is a form of linear regression calculation using a least square fit method. Almost all of the different models in this table outperform the TSF in terms of RMSE, although fall behind in directional accuracy. Another observation is that the general model is significantly more accurate than the comparable equivalent model which only uses one stock for training.

**Table 4.3:** Different model inputs and their different RMSE and accuracy values.

| Model | RMSE | Directional Accuracy (%) |
|---|---|---|
| Close | 0.53 | 48.90 |
| Close with Derivative | 0.54 | 50.80 |
| Close with Time as an Indicator | 0.55 | 52.02 |
| OHLC | 0.56 | 52.32 |
| OHLC with Derivative | 0.56 | 51.25 |
| OHLC with Time as an Indicator | 0.54 | 51.63 |
| OHLCV with All Indicators (Standard Model) | 0.90 | 48.06 |
| OHLCV with All Indicators, Trained on All Available Stocks (General Model) | 0.58 | 52.32 |
| Time Series Forecast Benchmark | 0.71 | 55.44 |

The fourth and fifth table analyze how the model behaves when trained on specific times of the day. Table 4.5 shows the model trained on all available data for one stock, meaning the model is trained on data for all times of the day, although the evaluation is made only on morning and evening data. In contrast to this, table 4.4 is trained and evaluated on only the specific time of morning and evening. Since the available data is reduced, a degradation in RMSE accuracy is to be expected. Table 4.5 therefore allows more data for training, although this data may be less relevant as it is trained on times which are not being predicted on. Both of these methods are made in order to better understand whether there is a difference in trading during different times of the stock market, and the results seem to indicate such a difference.

The directional accuracy is improved to a large extent for evening prediction for both methods, and for morning predictions as well when the model is trained on all times of the day. The best directional accuracy of 55.42 % was obtained for an evening prediction using all indicators trained on only evening data. The RMSE values are on the other hand lower for both evening and morning predictions.

**Table 4.4:** A standard OHLC model trained and tested on different periods of time on the stock market and their different RMSE and accuracy values.

| Model | RMSE | Directional Accuracy (%) |
|---|---|---|
| OHLC | 0.56 | 52.32 |
| OHLC - Morning | 0.62 | 51.13 |
| OHLC - Evening | 0.70 | 54.68 |
| OHLC with Indicators | 0.90 | 48.06 |
| OHLC with Indicators - Morning | 1.54 | 49.62 |
| OHLC with Indicators - Evening | 1.43 | 55.42 |

**Table 4.5:** A standard OHLC model trained on all stock data and evaluated on different times of the market.

| Model | RMSE | Directional Accuracy (%) |
|---|---|---|
| OHLC | 0.56 | 52.32 |
| OHLC - Morning | 0.64 | 54.89 |
| OHLC - Evening | 0.73 | 53.94 |

The lowest RMSE values are obtained with only using the closing price as a feature. This can mean that there is a problem of having too few data points and that overfitting easily occurs for more features. Alternatively, that the features used do not correlate well enough with the data to aid prediction in any meaningful way.

Comparing the overall directional accuracy results we can see that the models in this report are relatively low on accuracy compared to other classification models that exist. As the directional accuracy was not the primary target of optimization this could be expected. We can however see that the directional accuracy is not necessarily correlated to the lower RMSE values. This is an interesting find. The best directional accuracy obtained in this project was 55.4% for an evening prediction using all indicators trained on only evening data. This is lower compared to the work using LSTM by Y. Wang et al. which managed to achieve 60% [4]. There are a few reasons for why this might be the case. Firstly the model has a fixed learning rate, which should ideally be lowered as the number of epochs increase during training. Furthermore the models are made to be comparable with one another and therefore built on the same basis. The ideal configuration for one model does not necessarily mean that it is ideal for another one, however this way of making the models and presenting the results enables us to see trends and how the larger models react to different parameters.

One interesting observation was where the stock market was in a down trend, the training data would be on higher values and the test data would be on lower values. This was noticed early in the project when data collection for a few months was made. The data was in a down trend and the predictions were made on low (close to 0) values although the training was on higher values (close to 1). This led to the algorithm predicting aggressively and many predictions spiked where there was no change in price of the stock. The same should be true for data which has an uptrend. A more general observation was discussed by R. Konstantinou where stock data trained on a different range other than the test data yielded low accuracies [32]. This could practically mean that if we are predicting data for a stock which is at its lowest or highest peak, we could get a lower accuracy than desired. However in a real life application there will not be as large of a gap between the training and testing of data. A 70/30 split was made in this project for presenting accuracies and evaluating the models in a meaningful way, however this is unnecessary for a practical application.

An expectation of the layered model to perform better than the one-layered model was made as the assumption was that there would be more complex patterns for the stock price to be realized in the model. Since the one-layered model showed promise during optimization, it could mean that the stock is either overall less dependant on previous data than expected to make accurate prediction, or that the method used in this report is insufficient in finding these patterns and that another approach could be explored. See section 4.8 on future work for a discussion on this.

## 4.1 Performance

Since the weights of the model are randomized at model creation the time it takes for the model to train and reach an optimal state can vary between runs. Although the workload numbers are taken from a single training, they are assumed to be similar enough to reflect the general workload. The workload numbers presented in table 4.6 are models trained with data from 2021-12-16 to 2022-08-19, meaning just over 9 months of data.

Given an ideal scenario, the total amount of training time when distributing the workload would be effectively reduced to the fraction of 1 divided by the number of nodes or workers. As the existence of overhead of workload division and unequal sections this could not be the case practically.

**Table 4.6:** A workload overview of the created project. All of these workloads can be executed in parallel.

| Type of Workload<br>Number of LSTM Cells | Time (Minutes) |
|---|---|
| 10 | 24.35 |
| 50 | 27.75 |
| 100 | 17.51 |
| 500 | 12.99 |
| 1000 | 12.45 |
| 1500 (Standard Model) | 16.88 |
| Look-back Length | |
| 1 | 6.42 |
| 5 | 13.35 |
| 10 (Standard Model) | 16.88 |
| 50 | 30.00 |
| Isolation tests | |
| Close | 28.80 |
| Close with Derivative | 20.58 |
| Close with Time | 27.40 |
| OHLC | 50.47 |
| OHLC with Derivative | 19.12 |
| OHLC with Time | 61.08 |
| OHLC with All Indicators (Standard Model) | 16.88 |
| General Model | 990.48 |
| Time Tests | |
| OHLC Evening Only | 40.63 |
| OHLC Morning Only | 96.79 |
| OHLC with All Indicators - Morning Only | 17.52 |
| OHLC with All Indicators - Evening Only | 32.08 |
| Dropout Exploration | |
| No Dropout | 14.04 |
| Dropout 10% (Standard Model) | 16.88 |
| Dropout 25% (Excluding Last Layer) | 17.76 |
| Dropout 25% | 48.13 |
| Dropout 50% | 74.80 |

The reasoning for why there in some cases is an increase in training time for a lower number of LSTM cells or lower number of indicators, which can be considered counter-intuitive, is because the algorithm keeps finding better solutions after a while of training for these models which in turn leads to the early stopping callback with a patience of 100 to keep waiting. Meaning that the algorithms continues searching for another solution an additional 100 epochs which increases the execution time considerably. This callback was however necessary to train all of the models in the same way and enable a form of comparison and discussion.

The lack of available resources forced a simulation of the distribution. An assumption was made of using the same hardware on all computational nodes as described in section 3.8. An uneven distribution of resources could enable even faster training where longer tasks have the possibility to be assigned to faster hardware. A distribution of nodes which is non-uniform would therefore benefit this project more than the assumed uniform distribution.

The total time for all of the tasks excluding the general model was 710.72 minutes. Using two nodes would therefore be sufficient and the project would be finished in a little over half of the sequential time of using one node. Meaning a speedup of 1.7 where speedup is defined as serial execution time divided by parallel execution time.

Excluding the general model from the task distribution would in contrast contribute to a much higher speedup depending on the number of resources used. The project could be finished within a little over 96 minutes if all of the tasks were given a worker. Many tasks could however be executed on the same node before the longest task finishes on the first worker. A speedup of 7.25 times is therefore achieved when using 8 or more workers. Using more than 8 workers does not affect speedup as multiple tasks are able to execute on one worker. It is noteworthy that the overhead of the task distribution is minor in comparison to the execution time in this specific application.

Another possibility would be to divide the general model to the workers as well. There are many ways to do this. One way would be to divide the larger data size to smaller sub-samples and have multiple models which are then reduced to one. This division of work could also be applied to the other models and not only the general one. Given this approach, and a plentiful amount of nodes, the project could finish all of its workload within a few minutes. Using these algorithms would however change the process of how the models are trained and also the accuracy depending on the methods chosen. The algorithm would therefore have to be applied to all of the models to ensure that they maintain their ability of comparison to each other. This would result in a much greater complexity.

## 4.2 Thesis Objectives

There were four objectives stated and explored in this thesis. The first being how an LSTM model with heavy data information using a large amount of indicators behaves with different model parameters. The findings indicate that using a lower window size is beneficial when the size of the training data is relatively low. This is expected as using a lower look-back reduces the model complexity and therefore reduces over-fitting. In contrast, using a larger amount of LSTM cells seems in general to be a beneficial factor in this type of application. A reason for this might be the increased memory that comes with an increased amount of cells.

The second objective was to identify whether the introduction of a large amount of indicators aid prediction or if it simply contributes to over-fitting. The results show that the use of a large amount of indicators does not necessarily improve the accuracy of the model. A possible explanation would be that the amount of training data was too low for such a complex model. The comparison of the standard model with the general model indicates such an explanation to be plausible as the largest difference between them is the amount of data used.

The third objective was to investigate how the influence of time and the use of the derivative impact prediction. Using the derivative as an indicator compared to only using the closing price showed an increase in directional accuracy. This implies that the direction of the stock is affected by the prior direction. The results regarding time show that time of day impacted predictions in different ways. The directional accuracy was increased for evening predictions and for some morning predictions as well, even though RMSE was lower in both of these cases.

The last objective was to explore a solution to the demanding computational requirement of the project. This was solved by introducing a distribution of tasks across multiple nodes. A simulation of this was made which enabled the time taken for the machine learning section of the project to be greatly reduced given multiple workers.

## 4.3 The Random Nature of the Stock Market

An argument can be made that some of the accuracy calculations are better or worse because of randomness and that a much better model is practically impossible since the stock price follows random patterns. This can be the reason for why we see some accuracy improvements in for example the largest tested look-back of 50 compared to the smaller one of 10, although the best accuracy values are seen in the lowest look-back of 1. There is also the possibility of patterns which can only be observed for larger look-backs over 10. The same argument of randomness could be applied for the different amounts of LSTM cells, where RMSE does not seem to follow a clear trend. Another possibility of this is that the a lower amount of LSTM cells or look-back mean a less complex model, which can lead to a reduced impact of over-fitting and aid prediction in turn. This could also be the case for the large amount of indicators.

It is very likely however, that a better model is possible to create and that the stock prices do not follow purely random patterns. The overall results from the different models indicate that some trends are present. These trends could be a good starting point in an attempt to create a better algorithm with many indicators incorporated. An example of this is the aforementioned decreased look-back and increased number of LSTM cells for a very large model which seems to perform better than using a larger amount of look-back.

## 4.4 The General Model

We can observe from table 4.3 that both the directional accuracy and RMSE of the general model which was trained on 87 stocks in total were substantially better than that of the standard OHLCV with all indicators. The reason for this could be that we have a problem of having too few data points for one stock where specific scenarios do not repeat often enough for the training to be be complete. Using more data would therefore improve prediction. The downside of this model being that the stocks are not the same and might not behave in the same way, resulting in an accuracy which might be lower compared to only training on the same stock with the same amount of data. The use of a heavy increase of indicators seems therefore to have led to over-fitting for the standard model.

## 4.5 Over-fitting and Feature Selection

The reason for not achieving even better accuracy could be the effect of over-fitting. Even though the data validation set reaches its peak accuracy, overall over-fitting could be happening where noise is inevitably trained upon. This could in combination with a relatively small data set be the main reasons for why we see a lower accuracy for the indicator results.

Over-fitting can occur in the addition of the many different kinds of indicators since they can exhibit the same properties as that of noise. Testing these indicators separately as an additional feature to the OHLC data could help indicate whether they are useful. However, the problem in testing them separately is that dependencies between different indicators are not taken into account, which can be deceiving information. A feature which does not correlate well with specifically OHLC data can in combination with another feature correlate well and therefore aid prediction and increase accuracy. A better method would therefore be to exclude one indicator at a time and compare the accuracy after training. This would take a lot of time and is outside of the scope of this project. The final model in table 4.3 investigates the possibility of feature to feature correlation as all indicators are used, however this means that the model has an increased risk of over-fitting and the results might not be as representative. Ideally, a perfect feature selection would both increase accuracy, and performance by reducing training times.

The difficulty of using a stacked neural network is knowing the exact importance of features as they are dependent on multiple features which go through the stacks with different weights in the multi-layered network. This limits the approaches we can take for excluding features which seem not to matter and solely contribute to the over-fitting of data. Lime was used to examine financial indicator importance more clearly, however the 97 different features with 10 look-back each were weighted so low that no important information could be gathered from it [33]. The relative impact every single feature had on predictions were according to Lime, either "0.00" or "0.01".

## 4.6 Features

The reason for having many indicators is simple. More information leads to a better prediction. However the information must also be relevant to the data. The findings indicate that this might not be the case. The indicators might not have a tight correlation to the prediction, or simply that the method used does not find such a correlation. The indicators usually have a few parameters of their own, and some of the indicators may not be useful altogether. Modifying all of the financial indicators so that they are optimized for accuracy is arguably possible, although the problem of creating such an optimal algorithm is extremely complex.

Shorter time-spans could be fitting for this project as they would indicate more recent differences in price as we have to generate a prediction every 15 minute. A day trader can sit and wait to see how the trend changes, which the neural network does not have the possibility of doing, at least traditionally and in the way it is made in this project. Trends do not necessarily follow the data from data point to data point. They can take many data points ahead to be realized. This means that using the most common spans for the data set does not necessarily provide an accurate description to the following prediction. This is a big difference in trading style between a technical analysis trader and this project. Future work can therefore include an attempt to predict either slightly further ahead, for example using 15 minute interval data but constantly predicting one hour ahead, or try using shorter time spans altogether. This could be the reason for why we see an increase in accuracy for the longest look-back of 50, where the indicators further back could be relevant.

Another aspect of predicting further ahead is the possibility of improving directional accuracy. By looking 2 or 3 steps further into the future, the difference between the future prices compared to the current price are in general larger. This effectively forces the model to predict in a more aggressive manner. This could potentially cause the directional accuracy of the model to be increased. Although this has not been tested in this thesis and is left as a suggestion for future work as well.

Since there are many other features with different data processing methods that have a potential to increase the performance of the algorithm, the algorithm can not be considered optimal in this regard. A suggestion would be the creation of a model with more tightly correlated indicator data and dismissing data which the algorithm does not see as useful. Given the possibility of including different spans, there are in theory an infinite amount of indicators possible. This problem is therefore out of scope for this project.

### 4.6.1 Time of Day

We can see from the results that the directional accuracy for an evening is generally higher than that of the morning. Furthermore that the RMSE of the evening when using all indicators is higher. There are a few reasons for why this is the case. One reason which is assumed to be the highest factor is the larger difference in price between the end of one day and beginning of the next. This also impacts the way that indicators are calculated and is assumed to impact predictions negatively. During the evening there is no such occurrence. There could also be a possibility of a different mentality between the traders of the stock for the different times, which is the assumed reason for seeing that the morning predictions generally have better RMSE using standard OHLC data.

Another aspect of this is the so called after- and pre-market trading which is a place where trades can be made outside of market hours. This contributes to the difference in price between the days. To combat this issue, after market trading data would have to be used as well.

The difference of the RMSE between time as a feature and time as a training parameter where the model is trained on specifically different times is an important difference. Presenting the data to the algorithm in different ways has different impacts on accuracy. The specific method of introducing time as a feature increased the directional accuracy by over 3 percentage points. Using specific times of training and testing increased the directional accuracy by 7 percentage points for evening predictions.

### 4.6.2 Derivative of Price and Indicators

The derivative could potentially be seen as a form of trend-indicator. The results show that using the derivative of a stock in combination with its price improves directional accuracy compared to only using the price. An interesting suggestion is the exploration of the derivatives of the indicators as well. Indicators such as the negative and positive volume index seem ideal for this kind of pre-processing.

Negative volume index is a cumulative indicator which generally decreases with time, where its counterpart, positive volume index, generally increases with time. See table 2.1 for further details. Normalizing the negative volume index would therefore mean that the algorithm is introduced to lower values towards the end and higher at the beginning. The reverse is true for positive volume index. This could mean that the algorithm has too low of a reference to what the value indicates. Using a derivative of these could instead introduce the relative differences of the indicators.

## 4.7 Predicting a More Fitting Asset

The stocks in this report are governed by, among other factors, the financial state of a company and the market it is which the algorithm does not have any access to. Trading a more fitting asset without the impact and complexity of a company would therefore be appropriate. This change would effectively eliminate some aspects of fundamental analysis. Since this thesis investigates the effect of technical analysis using a neural network, it would likely yield a higher accuracy. Crypto-currencies are an example of such an asset as they are solely priced by supply and demand.

## 4.8 Future Work

There are a lot of ways to improve LSTM prediction models for stock market applications and quite a lot of research is being made on this during the time of writing this thesis. A few of these papers are added as references, one of which specifically mentions technical analysis in their title [34]–[38]. Some thoughts and ideas will now follow.

The focus of this thesis was mainly on replicating technical analysis as a trading practice with a deep learning prediction model. However, incorporating another deep learning model based on fundamental analysis would likely result in better overall accuracy. Future work could therefore include for example incorporating news sentiment analysis as done by Y. Guo [13].

A closer look at indicators and their input spans could greatly affect performance. It is possible that lower spans would yield a more accurate model. This idea is considered as the success of day traders is not limited to an action or prediction at every time step, which is required from the LSTM model. Another suggestion in line with this would be using larger time steps in combination with a higher look-back.

Another indicator that was considered was the "pivot" indicator [39]. A notable aspect of using the pivor indicator is that normalization should be applied differently as the relative differences in price should be kept, much like the OHLC data. Additionaly, pattern-recognition and various pricing models such as the Black-Scholes or Monte-Carlo simulation models could also be included in future implementations of this project [40].

Another suggestion for creating a better model would be the incorporation of a classification model for price direction, which then could be used as a feature for the larger regression model or as ensemble learning. The reasoning behind this is the improved classification result which in turn could lead to even better RMSE accuracies. In contrast to the proposed LSTM regression model, classification results from a recent classification model using LSTM have seen as high values as 65 percent [4].

To make the model even more accurate, a suggestion of using a variable learning rate or batch size and investigating the use of a larger amount of LSTM cells is proposed. This would increase training time but could possibly improve accuracy. KerasTuner is suggested for obtaining optimal model hyperparameters [41].

### 4.8.1 Creation of a More Accuarate Model

The following changes discussed in the future work section are listed below to clearly define the measures that could be taken which would likely result in an overall increase of both RMSE and directional accuracy compared to the current model.

- Using more data with only one asset for training.
- Changing the prediction of stock market data to crypto-currency prices.
- Incorporating news sentiment analysis and a classification model with the algorithm.
- Removing noisy indicators and tuning current ones by examining all features in more detail as well as their spans.
- An investigation of pivot points, pattern recognition indicators and various pricing models for incorporation with the model.
- Using a variable learning rate or batch size and re-tuning all model parameters carefully, including an investigation of even further increasing the number of LSTM cells.

Although this model would be more computationally demanding as it is more complex and handles more data, it would likely result in more accurate predictions.

# 5
# Conclusion

Predicting the stock market is a beneficial aim as correctly doing so would result in financial profit. However, creating an accurate prediction algorithm for a stock market applications using deep learning is a difficult task. The noisy nature of the stock market makes most prediction methods unsatisfactory. This work has investigated the effects of using a large amount of financial indicators for stock market prediction in greater detail and developed a model for predicting intra-day time spans using a deep learning algorithm with Long Short-Term Memory.

An investigation in how this larger model behaves with different parameters, as well as in further detail discussing the effect of the derivative as well as the impact of time was made. The findings indicate favorable results in both using the derivative and the time as features for the application of stock market prediction. Further increased accuracies are observed in predicting the stock market at specific times of the day.

The model used in the thesis was optimized to a three layer stacked configuration. The parameters which showed the highest prediction accuracies was using lower window sizes as well as larger amounts of LSTM cells. The work has also introduced many new ideas and identified areas for future work where improving the Root Mean Squared Error of LSTM-based models within stock market prediction is the main goal.

In addition to stock market analysis, a task distribution system was introduced to manage the long training times. The time taken for the project to complete training was accelerated by enabling a distribution of the various models over multiple workers. This was made by assigning the heaviest workloads to the available workers first, essentially having a longest time first scheduling order. Simulations of the models made over mininet show a large increase in speedup, allowing the models to be trained up to 7.25 times faster given 8 or more workers. The distribution is furthermore scalable and general, allowing new tasks to be easily included as well as new workers.

# References

[1] M. Nabipour, P. Nayyeri, H. Jabani, Shahab, and A. Mosavi, "Predicting stock market trends using machine learning and deep learning algorithms via continuous and binary data; a comparative analysis," *IEEE Access*, vol. 8, pp. 150 199–150 212, 2020.

[2] N. Z. Hakim, J. J. Kaufmann, G. Cerf, and H. E. Meadows, "Nonlinear time series prediction with a discrete-time recurrent neural network model," in *IJCNN-91-Seattle International Joint Conference on Neural Networks*, Seattle, WA, USA: IEEE, 2002.

[3] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[4] Y. Wang, Y. Liu, M. Wang, and R. Liu, "LSTM model optimization on stock price forecasting," in *2018 17th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, Wuxi, China: IEEE, Oct. 2018.

[5] *Market capitalization of listed domestic companies (current us$)*, `https://data.worldbank.org/indicator/CM.MKT.LCAP.CD?locations=1W.`, Accessed: 2022-9-8.

[6] B. Qian and K. Rasheed, "Stock market prediction with multiple classifiers," en, *Appl. Intell.*, vol. 26, no. 1, pp. 25–33, Jan. 2007.

[7] *U.s. equities market volume summary*, Aug. 2022. [Online]. Available: `http://www.nasdaqtrader.com/trader.aspx?id=FullVolumeSummary`.

[8] A. Graves, *Generating sequences with recurrent neural networks*, 2013. DOI: `10.48550/ARXIV.1308.0850`. [Online]. Available: `https://arxiv.org/abs/1308.0850`.

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," en, *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[10] F. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," *Neural computation*, vol. 12, pp. 2451–71, Oct. 2000. DOI: `10.1162/089976600300015015`.

[11] G. Van Houdt, C. Mosquera, and G. Nápoles, "A review on the long short-term memory model," English, *Artificial intelligence review: An international survey and tutorial journal*, vol. 53, no. 8, pp. 5929–5955, Dec. 2020, Funding Information: We thank the reviewers for their very thoughtful and thorough reviews of our manuscript. Their input has been invaluable in increasing the

quality of our paper. Also, a special thanks to prof. J?rgen Schmidhuber for taking the time to share his thoughts on the manuscript with us and making suggestions for further improvements. Publisher Copyright: © 2020, Springer Nature B.V. Copyright: Copyright 2020 Elsevier B.V., All rights reserved., ISSN: 0269-2821. DOI: 10.1007/s10462-020-09838-1.

[12] L. Hou, J. Zhu, J. Kwok, *et al.*, "Normalization helps training of quantized lstm," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, *et al.*, Eds., vol. 32, Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/f8eb278a8bce873ef365b45e939da38a-Paper.pdf.

[13] Y. Guo, "Stock price prediction based on lstm neural network: The effectiveness of news sentiment analysis," in *2020 2nd International Conference on Economic Management and Model Engineering (ICEMME)*, 2020, pp. 1018–1024. DOI: 10.1109/ICEMME51517.2020.00206.

[14] J. Wilder, *New Concepts in Technical Trading Systems.* Trend Research, 1978, ISBN: 9780894590276. [Online]. Available: https://books.google.se/books?id=WesJAQAAMAAJ.

[15] S. B. Islam, M. M. Hasan, and M. M. Khan, "Prediction of stock market using recurrent neural network," in *2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2021, pp. 0479–0483. DOI: 10.1109/IEMCON53756.2021.9623206.

[16] Y. Liu, "Analysis and forecast of stock price based on lstm algorithm," in *2021 IEEE International Conference on Computer Science, Electronic Information Engineering and Intelligent Control Technology (CEI)*, 2021, pp. 76–79. DOI: 10.1109/CEI52496.2021.9574519.

[17] S. Kushwaha, N. S. Naruka, and S. Ramamoorthy, "Prospective stock analysis model to improve the investment chances using machine learning," in *2021 Asian Conference on Innovation in Technology (ASIANCON)*, 2021, pp. 1–5. DOI: 10.1109/ASIANCON51346.2021.9544846.

[18] A. de Myttenaere, B. Golden, B. Le Grand, and F. Rossi, "Mean absolute percentage error for regression models," *Neurocomputing*, vol. 192, pp. 38–48, 2016, Advances in artificial neural networks, machine learning and computational intelligence, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2015.12.114. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231216003325.

[19] *Yfinance*, en, https://pypi.org/project/yfinance/, Accessed: 2022-9-8.

[20] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, *On large-batch training for deep learning: Generalization gap and sharp minima*, 2016. DOI: 10.48550/ARXIV.1609.04836. [Online]. Available: https://arxiv.org/abs/1609.04836.

[21] J. Ehlers, *Signal analysis concepts*, 2022. [Online]. Available: http://www.technicalanalysis.org.uk/moving-averages/Ehle.pdf.

[22] F. Chollet, *Keras: Deep learning for humans*, en, https://keras.io, Accessed: 2022-9-8.

[23] K. Johnson, *Twopirllc/pandas-ta: Technical analysis indicators.* [Online]. Available: https://github.com/twopirllc/pandas-ta.

[24] M. Fortier, *Ta lib : Technical analysis library.* [Online]. Available: `https://www.ta-lib.org/about.html`.

[25] N. Vogel, *About us*, 2022. [Online]. Available: `https://www.investopedia.com/about-us-5093223`.

[26] A. Kumar, "Effects of different normalization techniques on the convolutional neural network," in *2021 8th International Conference on Computing for Sustainable Global Development*, pp. 201–204.

[27] M. A. Istiake Sunny, M. M. S. Maswood, and A. G. Alharbi, "Deep learning-based stock price prediction using lstm and bi-directional lstm model," in *2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, 2020, pp. 87–92. DOI: `10.1109/NILES50944.2020.9257950`.

[28] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, 2015. DOI: `10.48550/ARXIV.1502.03167`. [Online]. Available: `https://arxiv.org/abs/1502.03167`.

[29] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2014. DOI: `10.48550/ARXIV.1412.6980`. [Online]. Available: `https://arxiv.org/abs/1412.6980`.

[30] B. Lantz. [Online]. Available: `http://mininet.org/`.

[31] X. Xiao, "A direct proof of the 4/3 bound of lpt scheduling rule," in *Proceedings of the 2017 5th International Conference on Frontiers of Manufacturing Science and Measuring Technology (FMSMT 2017)*, Atlantis Press, 2017/04, pp. 486–489, ISBN: 978-94-6252-331-9. DOI: `https://doi.org/10.2991/fmsmt-17.2017.102`. [Online]. Available: `https://doi.org/10.2991/fmsmt-17.2017.102`.

[32] R. Konstantinou, *Stock market prediction using artificial neural networks*, 2022. [Online]. Available: `https://odr.chalmers.se/bitstream/20.500.12380/256121/1/256121.pdf`.

[33] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should I trust you?": Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, 2016, pp. 1135–1144.

[34] A. K. Singh, J. Patra, M. Chakraborty, and S. Gupta, "Prediction of indian government stakeholder oil stock prices using hyper parameterized LSTM models," in *2022 International Conference on Intelligent Controller and Computing for Smart Power (ICICCSP)*, Hyderabad, India: IEEE, Jul. 2022.

[35] C. Jethva, S. Dudani, E. Malik, M. Sonje, and G. Tanna, "Stock analyzer and bot using machine learning," in *2022 IEEE Region 10 Symposium (TENSYMP)*, Mumbai, India: IEEE, Jul. 2022.

[36] J. Sen and S. Mehtab, *Long-and-short-term memory (LSTM) NetworksArchitectures and applications in stock price prediction*, Jul. 2022.

[37] U. Bisarya, V. Parekh, and S. Bhattacharjee, "Stock price prediction using corporation network and LSTM," in *2022 2nd International Conference on Intelligent Technologies (CONIT)*, Hubli, India: IEEE, Jun. 2022.

[38] A. Singh, G. Bhardwaj, A. P. Srivastava, *et al.*, "Application of neural network to technical analysis of stock market prediction," in *2022 3rd International*

*Conference on Intelligent Engineering and Management (ICIEM)*, London, United Kingdom: IEEE, Apr. 2022.

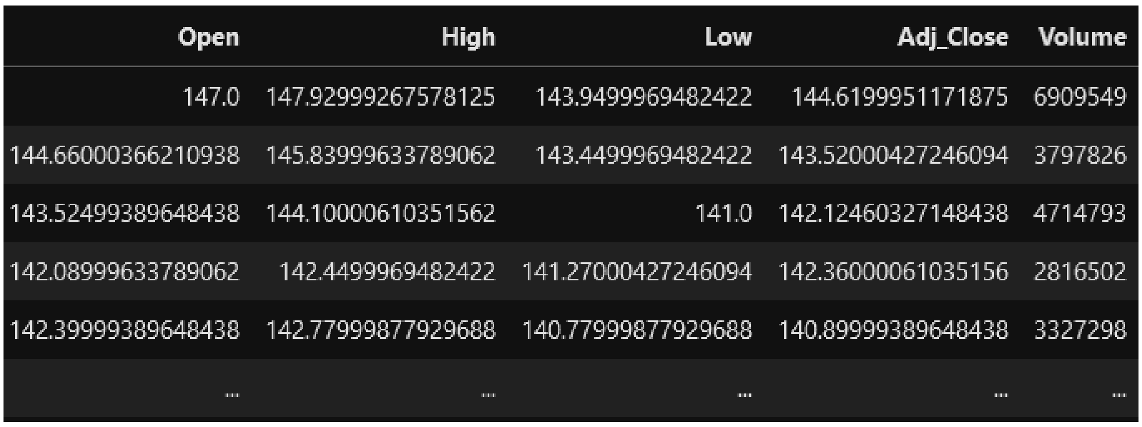
[39] C. Mitchell, *Pivot point: Definition, formulas, and how to calculate*, 2022. [Online]. Available: `https://www.investopedia.com/terms/p/pivotpoint.asp`.

[40] Q. Jiang, "Comparison of black–scholes model and monte-carlo simulation on stock price modeling," Jan. 2019. DOI: `10.2991/aebmr.k.191217.025`.

[41] *Keras-tuner: Hyperparameter tuning for humans*, en, `https://keras.io/keras_tuner/`, Accessed: 2022-9-8.

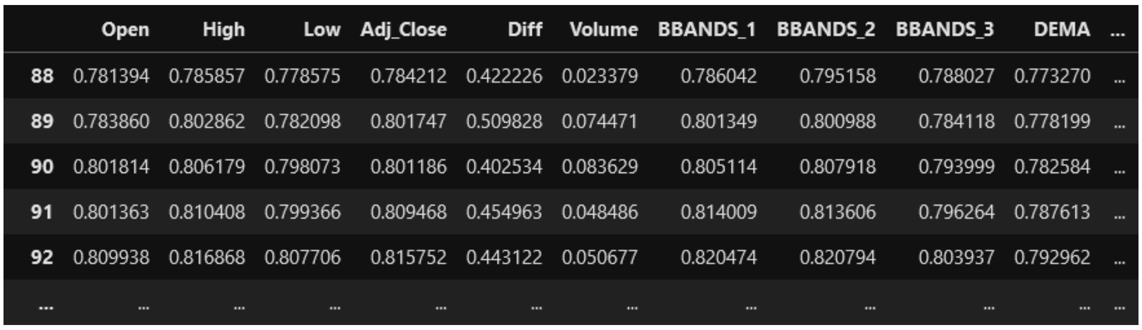
# A
# Stock Data and Indicators

The following figures are illustrations of the data collection both before and after data processing.



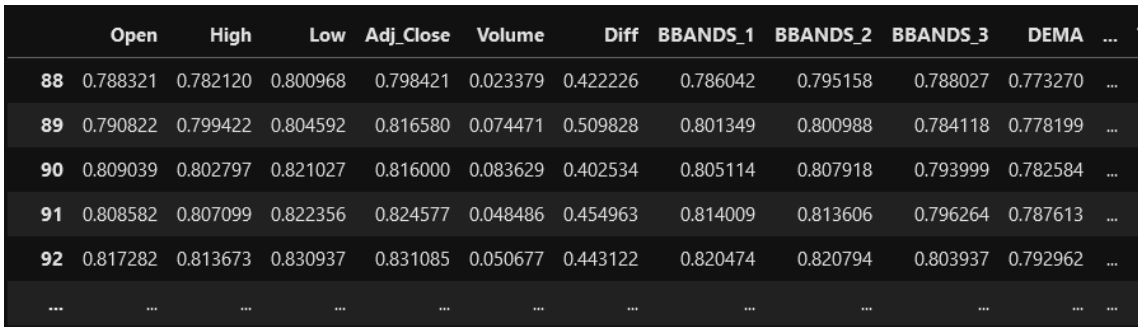
| Open | High | Low | Adj_Close | Volume |
|---|---|---|---|---|
| 147.0 | 147.92999267578125 | 143.9499969482422 | 144.6199951171875 | 6909549 |
| 144.66000366210938 | 145.83999633789062 | 143.4499969482422 | 143.52000427246094 | 3797826 |
| 143.52499389648438 | 144.10000610351562 | 141.0 | 142.12460327148438 | 4714793 |
| 142.08999633789062 | 142.4499969482422 | 141.27000427246094 | 142.36000061035156 | 2816502 |
| 142.39999389648438 | 142.77999877929688 | 140.77999877929688 | 140.89999389648438 | 3327298 |
| ... | ... | ... | ... | ... |

**Figure A.1:** Stock data before any processing



| | Open | High | Low | Adj_Close | Diff | Volume | BBANDS_1 | BBANDS_2 | BBANDS_3 | DEMA | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **88** | 0.781394 | 0.785857 | 0.778575 | 0.784212 | 0.422226 | 0.023379 | 0.786042 | 0.795158 | 0.788027 | 0.773270 | ... |
| **89** | 0.783860 | 0.802862 | 0.782098 | 0.801747 | 0.509828 | 0.074471 | 0.801349 | 0.800988 | 0.784118 | 0.778199 | ... |
| **90** | 0.801814 | 0.806179 | 0.798073 | 0.801186 | 0.402534 | 0.083629 | 0.805114 | 0.807918 | 0.793999 | 0.782584 | ... |
| **91** | 0.801363 | 0.810408 | 0.799366 | 0.809468 | 0.454963 | 0.048486 | 0.814009 | 0.813606 | 0.796264 | 0.787613 | ... |
| **92** | 0.809938 | 0.816868 | 0.807706 | 0.815752 | 0.443122 | 0.050677 | 0.820474 | 0.820794 | 0.803937 | 0.792962 | ... |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |

**Figure A.2:** Stock data after processing with indicators and normalization



| | Open | High | Low | Adj_Close | Volume | Diff | BBANDS_1 | BBANDS_2 | BBANDS_3 | DEMA | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **88** | 0.788321 | 0.782120 | 0.800968 | 0.798421 | 0.023379 | 0.422226 | 0.786042 | 0.795158 | 0.788027 | 0.773270 | ... |
| **89** | 0.790822 | 0.799422 | 0.804592 | 0.816580 | 0.074471 | 0.509828 | 0.801349 | 0.800988 | 0.784118 | 0.778199 | ... |
| **90** | 0.809039 | 0.802797 | 0.821027 | 0.816000 | 0.083629 | 0.402534 | 0.805114 | 0.807918 | 0.793999 | 0.782584 | ... |
| **91** | 0.808582 | 0.807099 | 0.822356 | 0.824577 | 0.048486 | 0.454963 | 0.814009 | 0.813606 | 0.796264 | 0.787613 | ... |
| **92** | 0.817282 | 0.813673 | 0.830937 | 0.831085 | 0.050677 | 0.443122 | 0.820474 | 0.820794 | 0.803937 | 0.792962 | ... |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |

**Figure A.3:** If we normalize column-wise we get this feature-to-feature disassociation

This additional figure illustrates the difference between column-wise normalization and normalization of the OHLC features with the same parameters to keep the relationship between the features. Note how the "Low" column is higher than the "High" column.

**Table A.1:** The complete list of all features used in the project:

| | |
|---|---|
| Open | High |
| Low | Close |
| Volume | Derivative |
| Bollinger Bands 1 | Bollinger Bands 2 |
| Bollinger Bands 3 | Double Exponential Moving Average |
| Exponential Moving Average | Hilbert Transform |
| Kaufman Adaptive Moving Average | Moving Average |
| MESA Adaptive Moving Average 1 | MESA Adaptive Moving Average 2 |
| Parabolic SAR | Simple Moving Average |
| Triple Exponential Moving Average | Triangular Moving Average |
| Weighted Moving Average | Average Directional Movement Index |
| Absolute Price Oscillator | Aroon Oscillator |
| Balance Of Power | Directional Movement Index |
| Moving Average Convergence/Divergence | Moving Average Convergence/Divergence Signal |
| Money Flow Index | Momentum |
| Percentage Price Oscillator | Rate of change |
| Relative Strength Index | Stochastic Relative Strength Index K |
| Stochastic Relative Strength Index D | Stochastic Relative Strength Index 1 |
| Stochastic Relative Strength Index 2 | ROC of a Triple Smooth EMA |
| Ultimate Oscillator | Williams' %R |
| Typical Price | Accumulation/Distribution Oscillator |
| On-Balance Volume | Average True Range |
| True Range | Linear Regression |
| Standard Deviation | Time Series Forecast |
| Variance | Even Better Sinewave |
| Awesome Oscillator | Absolute Price Oscillator |
| Bias | BRAR AR |
| BRAR BR | Commodity Channel Index |
| Center of Gravity | Correlation Trend Indicator |
| Chande Momentum Oscillator | Efficiency Ratio |
| Elder Ray Index Bull | Elder Ray Index Bear |
| Fisher Transform 1 | Fisher Transform 2 |
| Inertia | KST Oscillator |
| KST Signal | Pretty Good Oscillator |
| Psychological Line | Percentage Volume Oscillator |
| Percentage Volume Oscillator Histogram | Percentage Volume Oscillator Signal |
| Relative Vigor Index | Relative Vigor Index Signal |
| True Strength Index | True Strength Index Signal |
| Drawdown | Drawdown Percent |
| Drawdown Log | Percent Return |
| Aberration ZG | Aberration SG |
| Aberration XG | Aberration ATR |
| Mass Index | Relative Volatility Index |
| Elder's Thermometer | Elder's Thermometer MA |
| Elder's Thermometer LONG | Elder's Thermometer SHORT |
| Ulcer Index | Chaikin Money Flow |
| Elder's Force Index | Ease of Movement |
| Negative Volume Index | Positive Volume Index |
| Time | |