A comparative study of recurrent neural networks and statistical techniques for forecasting the stock prices of JSE-listed securities

by

Rushin Galant

Supervised by

Prof Patrick Marais

A Thesis submitted for the degree of
MSc in Information Technology

in the
Department of Computer Science
University of Cape Town

September 2022

Declaration of Authorship

I, Rushin Galant, declare that this work presented is my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

Abstract

As machine learning has developed, the attention of stock price forecasters has slowly shifted from traditional statistical forecasting techniques towards machine learning techniques. This study investigated whether machine learning techniques, in particular, recurrent neural networks, do indeed provide greater forecasting accuracy than traditional statistical techniques on the Johannesburg Securities' Exchanges' top forty stocks.

The Johannesburg Securities Exchange represents the largest and most developed stock exchange in Africa, though limited research has been performed on the application of machine learning in forecasting stock prices on this exchange. Simple recurrent neural networks, Gated Recurrent Units and Long-Short Term Memory Units were thoroughly evaluated with a Convolutional Neural Network and a random forest were used as machine learning benchmarks.

Historical data was collected for the period 2 January 2019 to 29 May 2020, with the 2019 calendar year being used as the training dataset. Both a train once and a Walkforward configuration were used. The number of input observations utilised were varied from four to fifteen observations whilst making forecasts from one up to ten timesteps into the future. The Mean Percentage Error was utilised to measure forecasting accuracy. Different configurations of the Neural Network models were assessed, including considering whether bidirectionality improved forecasting accuracy. The neural networks were run using two different datasets, the historical stock prices on its own and the historical stock prices with the market index (the JSE All Share Index) to determine whether including the market index improves forecasting accuracy.

The study found that bidirectional neural networks provided more accurate forecasts than neural networks that did not incorporate bidirectionality. In particular, the Bidirectional Long Short-Term Memory provided the greatest forecasting accuracy for one step forecast whilst the Bidirectional GRU was more accurate two to eight time steps into the future with the Bidirectional LSTM model being more accurate for nine and ten time steps into the future. However, the classical statistical model, the theta method, significantly outperformed all machine learning models. This is likely the result of the unforeseen impact of the covid-19 pandemic on financial markets that would not have been factored into the training sets of the machine learning algorithms. ...

Acknowledgements

I would like to thank my family for their support over the period of time it took to complete this dissertation as well as Professor Patrick Marais for his guidance throughout the research process.. . .

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Many share traders attempt to profit from share price fluctuations by attempting to accurately forecast movements in the prices of stock-exchange listed shares [4]. To do this, many forecasting techniques have been developed, having their roots in a range of different disciplines including mathematics, statistics and economic theory, with each technique situated at a different point on a spectrum between science and art. More recently, research applying machine learning to share price forecasting has been growing in popularity internationally, as researchers attempt to improve forecasting accuracy over a longer period of time to improve profitability of investments made. This has proven particularly challenging given the intrinsically noisy nature of stock prices, their complexity and their volatility. Machine learning techniques have been able to capture these nuances in stock price movements to a better extent when compared to traditional forecasting methods [3].

1.1 Research statement and objectives

Machine learning algorithms would be able to more accurately forecast future stock prices over extended forecast periods due to their ability to capture the underlying nuances present in stock prices and the broader economic market. The aim of this research was to compare the forecasting accuracy of machine learning architectures, with specific emphasis on recurrent neural networks, to that of classical statistical techniques in forecasting JSE-listed stock prices. The research also analysed the forecasting accuracy of machine learning algorithms for up to ten time steps in the future. By performing this analysis, insight was obtained on the efficiency of the Johannesburg Securities Exchange in terms of the Efficient Market Hypothesis.

1.2 Methodology

Survey papers evaluating the forecasting accuracy of both machine learning and traditional statistical methods for timeseries data was consulted as a starting point. From the research performed, statistical and machine learning models were selected to be included in further research. Recurrent Neural Networks were identified as an area within machine learning that displayed a great deal of potential to forecast stock prices. Further research was then done to evaluate existing research on Recurrent Neural Networks in forecasting share prices across time periods.

Based on the research performed, machine learning models were built to evaluate the forecasting accuracy of the Recurrent Neural Network architectures whilst ultimately attempting to build a configuration with the greatest degree of forecasting accuracy. The number of inputs used as historical observations into the models were varied to evaluate how this impacted upon forecasting accuracy over a forecasting window covering ten days. Both the historical share prices on its own and coupled with the market index were added to the models to evaluate whether adding the market index improved the forecasting accuracy. The data utilised was obtained from the Refinitiv Eikon platform, exported into CSV files and mounted to the Google Collaboratory virtual machine in which the models were built. The forecasts produced by the models were compared to the actual observations at the forecast time period, with the Mean Percentage Error between the two calculated. The accuracy of the built models were evaluated based on these Mean Percentage Errors. This was selected given the range of share prices (from below one rand up to over one thousand rand) and would thus allow for comparability between forecasts of prices for different shares. The forecasting accuracy of the machine learning models were also compared to the forecasting accuracy of traditional statistical forecasting methods.

1.3 Thesis Structure

Foundational background information on stock markets and machine learning is presented in Chapter 2, a review of existing literature is presented in Chapter 3, information on data pre-processing and the methodology utilised in presented in Chapter 4, experiments using statistical models are presented in Chapter 5, experiments using machine learning models are presented in Chapter 6. Finally, overall observations and conclusions are presented in Chapter 7.

Chapter 2

Background

This chapter contains a summary of background literature relevant to the research study performed. It begins with a description of the stock market and stocks, the different investment strategies employed by investors. It then presents a short discussion on the concept of market efficiency. Time series forecasting using both classical statistical methods and machine learning is then introduced and discussed. It then focuses on recurrent neural networks, its components and its variants.

2.1 The Stock Market

A stock market is a public market where company stocks and derivatives are traded at agreed upon prices. Investors are brought together at stock markets with the intention of trading these financial instruments at prices that are influenced by supply and demand. Stocks that are in great demand relative to its supply will see its price increase, whereas stocks that have a supply that exceeds its demand would see its price decrease [5]. The stock market is known for its complexity and volatility. People are always searching for accurate and effective ways to guide stock trading [6].

A share, or a stock, is a document issued by a company entitling the stock's holder to a defined proportion of an ownership interest in that company. Stocks are generally divided into two broad categories. Common stocks, representing the majority of stocks issued, represents an ownership interest in a company entitling its holders to an unguaranteed dividend, i.e., a fluctuating portion of the entity's profits, when the board of the entity declares that such a distribution be made. Common stocks also entitle their holders to voting rights that can be exercised to influence the entity's long-term decision making, most notably evident in their ability to elect members of the entity's board

of directors [5]. As the returns earned by common stockholders are directly impacted by the performance of the underlying entity as well as other market indicators, their trading prices are highly volatile.

In contrast, preferred stocks represent a degree of ownership in a company, but without any voting rights. Instead, preferred stockholders are entitled to a guaranteed fixed rate of dividend, before a dividend is declared for common stockholders [5]. Given that returns earned by preferred stockholders are generally fixed, calculating a price for these stocks are relatively easier and simpler exercise.

TABLE 2.1: Share price terminology

Opening Price	The price a stock trades at, at the start of the business day
Closing Price	The price that a stock trades at, at the end of a business day
High Price	The highest price at which a stock trades at during the business day
Trading Volume	The quantity of individual stocks traded for a specific financial instrument during the business day

2.1.1 Investment Strategies

Investors generate a return on the stock market by investing in stocks with the intention of earning dividends, capital appreciation, or a combination of the two. Investors trading with the intention of capital appreciation normally take one of two widely used approaches to investing, fundamental analysis or technical analysis. Fundamental analysis refers to an approach whereby the prospective investment's earnings and dividends potential, interest rate expectations and risk factors are considered to determine whether the investment's share price is expected to increase in the future. In contrast, technical analysis refers to the search for recurrent and predictable patterns in stock prices. Traders adopting a technical trading strategy do not believe information other than the share price is necessary for forecasting share prices because if share prices respond slowly enough, the analyst will be able to identify a trend that can be exploited profitably before the price fully reflects that external information [7].

2.1.2 Market Efficiency

One of the main areas of research conducted on financial markets relates to market efficiency. The Efficient Market Hypothesis states that prices of securities fully reflect all available information and that any new information would rapidly be assimilated by the market and reflected in the share prices. Closely linked to this is the concept that stock prices follow a random walk, i.e., stock prices are independent of each other and

that past stock price changes should not affect future stock price changes [8]. There are three versions of the efficient market hypothesis. The weak-form hypothesis holds that stock prices already reflect all available information that can be obtained by analysing market trading data such as historical prices and trading volumes. In this situation, technical analysis would be fruitless, as the share price would immediately reflect the latest available information [9]. As a result, stock prices follow a random walk, making it impossible to predict a future price based on a series of past prices [10]. However, no financial market appears to be fully weak form efficient as investors have still been able to adapt investment strategies, like technical analysis, which should not be profitable if this theory holds, to continue to earn returns from share trading. Testing to determine whether weak-form efficiency is present on a stock exchange include serial correlation tests, which determine whether prices follow a trend, and mechanical investment strategies, which determine whether historically identified price patterns are evident in share price movements [11]. Hansson [12] investigated weak form efficiency on the American, Brazilian and Swedish stock markets using both traditional time series modelling as well as recurrent neural networks. They found that the American and Brazilian stock markets showed greater evidence for the presence of weak form efficiency whilst the Swedish market did not.

The semi-strong form hypothesis holds that stock prices would have already incorporated all publicly available information related to the stock. This includes price and trading data as well as information unique to the company represented by the stock, such as management competency and performance forecasts [8]. If this form of efficiency is present on a stock exchange, it would not be possible to profitability conduct an investment strategy dependent on fundamental analysis as all information needed for such a strategy would already have been incorporated into the prevailing stock price. Many investors do not accept that stock markets are efficient. As a result, they engage in fundamental analysis with the intent of finding further information that they could profitably exploit. Ironically, doing so increases market efficiency. The presence of semi-strong form efficiency in a stock market is determined through examining historical price changes as a result of stock-relevant information becoming available [11].

Finally, strong-form efficiency holds that stock prices would reflect all information already incorporated by the other two efficiency forms, as well as insider information only known by those close to the entity [8]. As a result, it would be impossible to profitably conduct share trading activities as all private and publicly available information has already been priced into the stock. Tests have been conducted to determine whether individuals with more access to information than the general market have been able to earn returns higher than the general market. However, company insiders are legally prohibited from trading based on non-public information. Further, professional fund

managers who likely have 'inside sources' at companies that could provide them with information not widely known tend to under-perform compared to returns earned by simply tracking a market index [11].

2.1.3 The Johannesburg Securities Exchange

The Johannesburg Securities Exchange is the largest stock exchange in South Africa. The Exchange's platform allows companies to issue shares and raise primary capital to fund business operations. This is referred to as the primary market. Additionally, the Exchange ensures that there is a secondary market for share trading so that investors may buy and sell shares. The Exchange also monitors the performance of companies as measured by movements in its share prices. The Exchange runs the Share Transactions Totally Electronic platform, which allows shares to be traded and settled electronically. Common stocks are the most popularly traded financial instruments on the Johannesburg Securities Exchange [11].

Extensive research has been conducted on market efficiency on the Johannesburg Securities Exchange. Noakes and Rajaratnam ([8]) have found and analysed multiple studies providing evidence for an against the Efficient Market Hypothesis on the Johannesburg Securities Exchange over an extended period of time. The results of their own studies were mixed, though they did find that stocks with mid to larger market capitalisations appear to be more efficient (i.e., their share prices appeared to follow a more random walk).

2.2 Classical Statistical Forecasting Techniques

The main driver of the analysis performed was the daily closing prices of JSE-listed stocks. This is an example of time series data as it changes over time. Time series data is expected to display certain characteristics. This includes, firstly, a trend, which refers to the general direction in which the data is moving, secondly, seasonality, which refers to patterns repeating themselves at regular and predictable intervals, thirdly, white noise, which is random values with seemingly little predictive value and, fourthly, autocorrelation, which refers to a very deterministic type of movement [13]. A stationary time series is one whose statistical properties such as the mean, variance and autocorrelation are all constant over time [13]. Given the nature of the external factors influencing stock price movements, stationarity cannot be assumed for share prices for an extended period of time.

2.2.1 Moving Averages

A popular technique used by technical analysts to forecast share prices is the use of moving averages. The Simple Moving Average Technique analyses a time series by averaging different subsets of the full data set. These subsets are equally weighted. The Exponential Moving Average Technique modifies this by giving the subsets an exponential weighting, where the most recent data points of the time series have a greater weight [14].

Bolton [14] found that the Simple Moving Average outperformed the Exponential Moving Average at their starting point of fourteen observable days. As the number of observable days increased, Bolton [14] found that the Exponential Moving Averages started outperforming the accuracy of the Simple Moving Averages during the period 1 March 2009 to 8 April 2014, for JSE-listed stocks. A disadvantage of using a moving average model is that it does not anticipate trends or seasonality in the data. This can be addressed by using differencing, whereby the movement from one time period to the next is analysed instead of the actual value of the data points [13].

2.2.2 ARIMA

The Autoregressive Integrated Moving Average Model (commonly referred to as 'ARIMA') is a statistical model used for analysing and forecasting time series data in a manner that models the next step in a sequence as a linear function of the observations and residual errors at prior time steps [15]. Brownlee [15] further notes that the ARIMA model has the following parameters:

- The lag order (denoted as 'p'): This denotes the number of lag observations (i.e. observations prior to the current observation) included in the ARIMA model.
- Degree of differencing (denoted by 'd'): This denotes the number of times raw observations are differenced.
- Order of moving average (denoted by 'q'): This denotes the size of the moving average window.

2.2.3 Holt-Winter's Exponential Smoothing

The Holt Winter's Exponential Smoothing method models the next time step as an exponentially weighted linear function of observations at prior time steps, using exponentially decreasing weights for older input observations, taking trends and seasonality

into account [16]. This is particularly valuable for share data where trends are visible. The model has the following hyper-parameters. If these hyper-parameters are not specified, the model would automatically tune them to their optimal values:

- Smoothing Level (Denoted as 'alpha'): Smoothing coefficient for the level.
- Smoothing slope (Denoted as 'beta'): Smoothing coefficient for the trend.
- Smoothing Seasonal (Denoted as 'gamma'): Smoothing coefficient for the seasonal component.
- Damping slope (phi): Coefficient for the damped trend. Dampening refers to reducing the size of the observed trend over future timesteps.

2.2.4 Theta Method

The Theta Method is a univariate forecasting method that builds upon simple exponential smoothing with drift. It is based on modifying the local curvature of time series data through a coefficient, 'theta' [17]. It works by identifying the individual components of the data, i.e., the trend cycle, seasonality and irregular component. These are then projected separately into the future, and then recombined to form a forecast of the underlying time series [18]. The theta method is of interest to time series forecasters due to its simplicity and high accuracy in forecasting [17].

The theta method works by, firstly, deseasonalising the data, if a statistically significant seasonal component is detected. The data is then decomposed into two Theta lines, Z(0) and Z(2). Z(0) is then extrapolated as a normal linear regression line whilst Z(2) is extrapolated using simple exponential smoothing. A forecast is then generated from the extrapolated Z(0) and Z(2) linesby the combination of their weights. Finally, the forecast is reasonalised if the original data was identified as having a seasonal component [19].

2.2.5 Baseline Models

A baseline model is a model used as a reference point for comparing how well another model (typically, a more complex one) is performing. For example, a logistic regression model might serve as a good baseline for a deep model. For a particular problem, the baseline helps model developers quantify the minimal expected performance that a new model must achieve for the new model to be useful. For the purposes of this study, classical statistical forecasting models were used as the baseline models.

2.3 Machine Learning

Mohri [20] defined machine learning broadly as computational methods using experience to improve performance or to make accurate predictions. It consists of designing efficient and accurate prediction algorithms. Machine learning has been studied and applied to solve various problems and perform various tasks. These include classification, whereby it is used to assign categories to items, regression, whereby a real value is predicted for each item, ranking, whereby items are ordered according to some criterion, clustering, whereby items are partitioned into homogeneous subsets, and dimensionality reduction, whereby an initial representation of items is transformed into a lower-dimension representation while preserving some properties of the initial representation [20].

These tasks are performed using different machine learning scenarios. Supervised learning, the most common scenario, refers to when the learner receives a set of labelled examples as training data and makes predictions for all unseen points. This is used for classification, regression and ranking problems. Unsupervised learning refers to when the learner only receives unlabelled training data and makes predictions for all unseen points. This is used for clustering and dimensionality reduction [21]. Semi-supervised learning refers to when a learner receives a sample of both labelled and unlabelled data and makes predictions for all unseen points. Reinforcement learning refers to when the training and testing phases are intermixed. The learner would actively interact with its environment, in some cases affecting its environment and immediately receiving a reward for each action [20].

2.3.1 Machine Learning in Finance

Mathur [4] has found that Machine Learning has been used to enhance a vast range of functions in both the consumer finance and capital market sub-sectors of the broader financial sector. Regarding its application specifically to stock market investments, Mathur noted that machine learning algorithms are used to complement the activities of quantitative analysts as market trades would have to be made at increasing speeds to maximise their profitability. However, the machine learning methods used tend to be basic. Other examples of the application of machine learning in finance include forecasting of financial results of individual companies and economic indicators of the overall financial markets.

2.3.2 Neural Networks

Neural networks consist of basic units comparable in certain respects to neurons. These units are linked to each other by connections whose strength is modifiable as a result of a learning process or algorithm. The units independently integrate the information received from its synapses in order to evaluate its state of activation [22]. Neural networks are made up of different layers. These layers are independent of one another and each layer can have an arbitrary amount of nodes. Deep learning, which is aspect of artificial intelligence, in the context of neural networks, refers to neural networks with complex multi-layers. Deep learning neural networks have more complex ways of connecting layers than standard neural networks and also have more neurons to express complex models [23].

Qiu, Wang and Zhou [6] note that the neural network in deep learning has become a popular predictor due to its good nonlinear approximation ability and adaptive self-learning. Utilising Neural networks for time series forecasting instead of traditional statistical methods provides various advantages. Neural Networks are robust to noise in their input data. This is particularly valuable for stocks listed on the Johannesburg Securities Exchange, which is known to be a market with a large degree of volatility in share prices. Additionally, neural networks do not make strong assumptions about the mapping functions, easily learning both linear and nonlinear relationships present in the input data [16].

2.3.3 Components of Neural Networks

There are various parts that are used to construct a machine learning model. These are defined and discussed below. This discussion was used to define how these parts would be defined within the machine learning models utilised within this study

Hyper-parameters

Hyper-parameters specifically refer to the parameters regulating the design of the model [1]. The main hyper-parameters used in neural networks are noted below:

TABLE 2.2: Hyper-parameters

Learning Rate	Controls how much we are adjusting the weights of our neural network with respect to the loss gradient.
Epoch	The number of times the entire data set is passed forward and backward through a neural network.
Batch Size	The total number of training examples present in a single batch

Optimisation

Optimisation refers to the process followed by a neural network algorithm to determine the best configuration of the network. The preferred manner to optimise neural networks is gradient descent [24]. Gradient Descent is an iterative optimisation algorithm used to find the best results of the model, i.e., the minima of a curve [25]. Variables within a model are updated iteratively and with each update the gradient descent method gradually converges upon the optimal value of the objective function.

The Adaptive Moment Estimation (referred to simply as 'Adam') gradient descent algorithm computes the adaptive learning rate for each parameter. It stores an exponentially decaying average of past squared gradients and an exponentially decaying average of past gradients [24]. Benefits of using the Adam optimisation algorithm include that it is straightforward to implement, computationally efficient, has low memory requirements, it is suitable for large volumes of data and is appropriate for problems with noisy or sparse gradients [26].

Activation functions

Activation functions are used to determine the output of a node within a neural network [25].

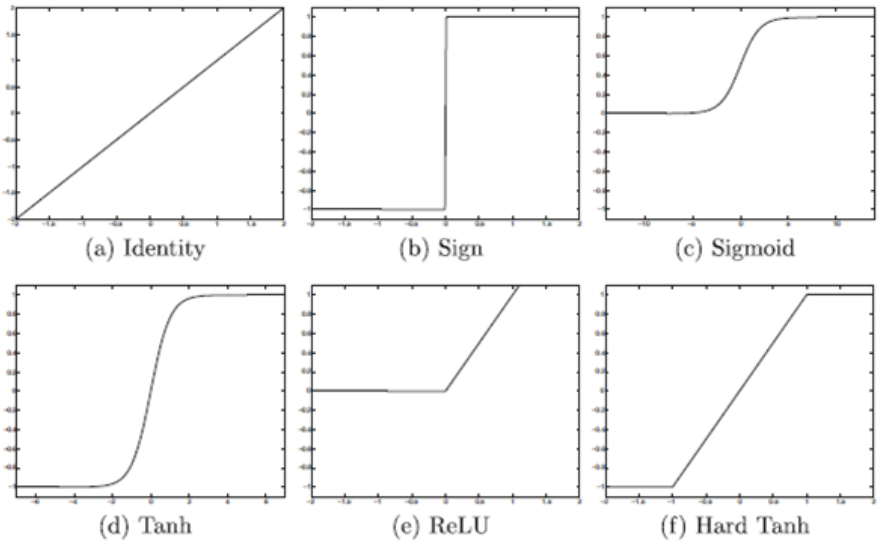

FIGURE 2.1: A visualisation of the most common activation functions [1]

Aggarwal [1] notes that the choice of activation function to use is dependent on the target variable to be predicted. If the target variable is real, then an identity activation function should be used. And the resulting algorithm would be the same as a least-squares regression. If the target variable is a probability of a binary class, then a sigmoidal function would be appropriate so that the output would then be a prediction. Some of the more widely used activation functions are noted in Table 2.3.

Table 2.3: Activation Functions

Sign function	$\phi(v) = \text{sign}(\text{v})$	The sign activation function is used to map predictions of binary outputs.
Sigmoid function	$\phi(v) = \frac{1}{(1+e^{-v})}$	The sigmoid function outputs a value in the range $(0, 1)$, which is useful for probabilities.
tanh	$\phi(v) = \frac{e^{2v}-1}{e^{2v}+1}$	The tanh function maps output over the range (-1,1). This is helpful when outputs are desired to be both positive and negative. It is also easier to train.
Rectified Linear Unit (ReLU)	$\phi(v) = \max\{\text{v}, 0\}$	Quite popular for its ease in training multi layered neural networks. All negative values become zero. It is less likely to create a vanishing gradient problem.

One of the problems that may arise when a neural network is determining its weights to use is that of the vanishing and exploding gradients. Vanishing gradient refers to a situation where updates in earlier layers are negligibly small whereas exploding gradient refers to a situation whereby updates to earlier layers are increasingly large [1]. Vanishing gradients are often experienced with sigmoid activation functions whereas a ReLU activation function is less likely to create a vanishing gradient problem.

As the input values and the output values of the machine learning models to be built for this study are positive values noted in South African Rands and due to its ability to address possible issues related to vanishing gradients, the ReLU activation function was used.

Loss function utilised for model training

A loss function is used to evaluate candidate solutions generated by the machine learning model developed. The loss function used is directly related to the activation function used in the output layer of the neural network and has to be selected in a manner that is mindful of the application it is being used for. Some of the more widely used loss functions are noted in Table 2.4.

As forecasts are to be evaluated within this research, the Mean Squared Regression Loss Function was utilised for training the models in the the experiments performed.

Backpropagation is used to compute the gradient of the loss function selected for model training. The backpropagation process consists of two phases. Firstly, the forward phase is required to compute output values and local derivatives at different nodes. This is

Table 2.4: Loss Functions

Mean Square Error	$L = (y - \hat{y})^2$	Calculated as the average of the squared differences between the predicted and actual values. This is widely used in regression scenarios.
Hinge loss	$L = \max\{0, 1 - \mathrm{y} \cdot \hat{\mathrm{y}}\}$ for $y \in \{-1; +1\}$	Can be used to implement a learning method (a support vector machine)
Binary targets (probabilistic predictions)	$L = \log(1 + \exp(-\mathrm{y} \cdot \hat{\mathrm{y}}))$	Assumes that observed value y is drawn from $\{-1; +1\}$ and prediction $\hat{y}$ is an arbitrary numeric value on using the identity activation function. Can also be used on Sigmoid function. Then loss is provided as the negative logarithm of $\lvert\frac{y}{2} - 0.5 + \hat{y}\rvert$
Categorical targets	$L = -\log(\hat{\mathrm{y}}_\mathrm{r})$	$\hat{y}_1 ... \hat{y}_k$ are the probabilities of the k classes

done by taking inputs for a training instance and feeding this forward into the neural network. This then results in a series of computations across the layers based on the weights that are current in the system at that time. The predicted value that is output can then be compared to the output of the training instance. Next, the backward phase is required to accumulate the products of local values over all paths from the node to the output. The main purpose of the backward phase is to learn the gradient of the loss function in model training with respect to the different weights by using the chain rule of differential calculus. These gradients are then used to update the weights. These gradients are learned in the backwards direction from the output node [1]. Backpropagation is managed internally by TensorFlow.

Overfitting refers to a situation where a model has been trained well with one dataset but does not guarantee it will provide good prediction performance on an unseen dataset, even if the model perfectly predicts the outputs in the training dataset. When the model has been under fit, i.e., it has been trained with very few data points, it would not generalise well with unseen data points and would not be able to forecast accurately. Generalisation is improved through adding more data points to the test set. [1]. Therefore, the perfect balance between overfitting and under fitting should be found. Aggarwal [1] notes that a good rule of thumb to use when deciding the size of the training set is that the total number of training data points should be at least two to three times larger than the number of parameters in the neural network. One of the main causes of overfitting is the use of a large number of parameters. Regularisation refers to constraining the model to use fewer nonzero parameters [1]. Consideration related

to this has been incorporated into the development of the neural network models in Chapter 4.

Based on the above discussion, the following was used within the machine learning models developed. Firstly, the Adaptive Moment Estimation optimiser was utilised. The Rectified Linear Unit activation function and the Mean Squared Regression loss function was used for model training.

Loss function utilised for performance evaluation

There are different measurements that can be applied in measuring the accuracy of machine learning in forecasting time series data. Measuring performance consists of comparing the predicted models from the forecasting tool used to the actual value being forecast. Botchkarev [27] performed a study in which they set out to develop a typology to be used for facilitating the use of performance metrics in machine learning. The framework developed grouped performance metrics into three groups: primary metrics, extended metrics and composite metrics. Primary metrics, specifically, are noted as being structured into three steps. These steps are calculating point distance, performing normalization and aggregating point results over a data set. Primary metrics are then used as the basis upon which metrics in the other two categories are constructed. The most common primary metrics are the Mean Absolute Error (denoted 'MAE'), Mean Square Error (denoted 'MSE') and the symmetrical Mean Absolute Percentage Error (denoted 'sMAPE').

The Mean Absolute Error is defined as the sum of the absolute value of the differences between all the expected values and predicted values, divided by the total number of predictions. The Mean Squared Error represents the average of the squared errors. The main difference between the Mean Absolute Error and the Mean Squared Error is the contribution of individual error values to the final result. In the Mean Absolute Error model, errors' contributions to the final error are linear, whereas for in the Mean Squared Error model the contributions are squared, meaning larger errors are penalised to a greater degree than smaller errors [28].

The Mean Absolute Percentage Error measures the absolute error as a percentage of the expected value. The symmetrical Mean Absolute Percentage Error is a modification of the Mean Absolute Percentage Error model that accounts for the asymmetrical nature of the Mean Absolute Percentage Error whereby the Mean Absolute Percentage Error model places a heavier penalty on negative errors as compared to positive errors [29]. This is represented as follows:

$$\text{sMAPE} = \text{mean}\left(\frac{2|y_t - \hat{y}|}{|y_t| + |\hat{y}|}\right) \tag{2.1}$$

For the purposes of this study, the Mean Absolute Percentage Error was used to analyse performance in individual share price forecasts over different time horizons. This would enable comparison between shares with large prices and shares with small prices to be done as shares with larger prices would be expected to have much larger Mean Squared Errors than shares with small prices.

2.3.4 Types of Neural Networks

Multi-layer Perceptron

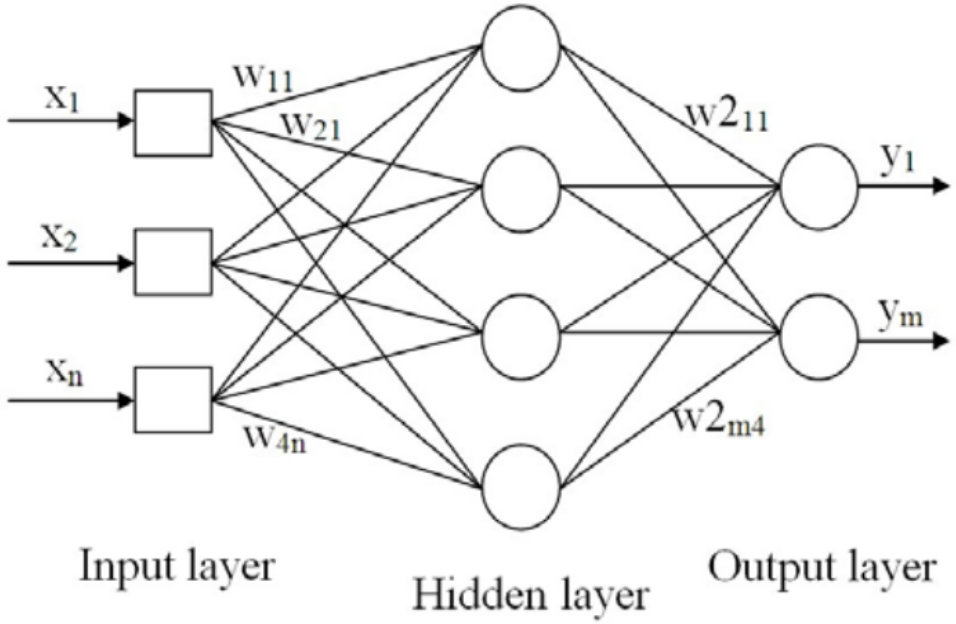

FIGURE 2.2: A schematic diagram of a Multi-Layer Perceptron (MLP) neural network [2]

The multi-layer perceptron is a simpler neural network that approximates a mapping function from input variables to output variables. It is valuable for sequence prediction problems as it is robust to noise and is capable of learning both linear and nonlinear relationships. It also addresses the limitations of classical linear models, like the ARIMA model, through its ability to have multivariate inputs and multi-step outputs [30]. Figure 2.2 exhibits the architecture of a simple multi-layer perceptron. This neural network consists of three layers: an input layer, a hidden layer and an output layer. Each layer consists of a number of neurons, which are successively connected by vectors X, W, W2 and Y. The input data is received into the network in the input layer. These are then passed through to the hidden layer's neurons via hidden weight connections. The hidden layer then performs computations, the results of which are passed to the output layer.

Recurrent Neural Networks

Recurrent Neural Networks are neural networks which operate over sequential data. They take sequences of vectors as their inputs and extract another sequence as output, which represents forecasts of information at potentially multiple time steps [31]. They allow input x_t to interact directly with the hidden state created from the inputs at previous time stamps. This hidden state changes at each time step as new data arrives. There is an implicit assumption that the time series exhibits a certain level of stationarity, i.e., the underlying properties do not change with time. What differentiates a recurrent neural network from other neural networks is the presence of a self-loop which will cause the hidden state of the neural network to change after the input of each x_t [1].

These networks add the explicit handling of order between time series observation. One of the main recurrent layers within the neural network architecture is the Long Short-Term Memory (LSTM) layer.

Brownlee [26] noted that LSTMs offer native support for sequences. A CNN reads across the entire input vector at once whilst in contrast the LSTM model reads one time step at a time and builds up an internal state that would be used as a learned context for forecasting.

Long Short-Term Memory (LSTM)

The Long Short-Term Memory unit was first proposed by Hochreiter and Schmidhuber [32]. It was originally developed to extend the Recurrent Neural Network's memory state to enable it to deal with longer input sequences [33]. One of the main drawbacks of standard Recurrent Neural Networks is that they tend to be difficult to train over a longer term because gradients either vanish or explode. This is addressed through the utilisation of the Long Short-Term Memory unit [31]. Long Short-Term Memory avoids these issues due to its unique storage unit structure and it is suitable for processing and predicting financial time series data [6]. A Long Short-Term Memory layer consists of a set of recurrently connected blocks, each of which consists of one or more recurrently connected memory cells. Each memory cell contains an input gate, which decides which values from the input are used to update the memory state, an output gate, which decides what to output based on input and the memory of the cell, and a forget gate, which decides which information to discard from the cell [15]. Advantages of a Long Short-Term Memory unit includes that it generalises well, can handle noise in the dataset and that there appears to be no need for parameter fine tuning as it works well over a broad range of parameters such as the learning rate input gate bias and output gate bias [32].

Gated Recurrent Unit

The Gated Recurrent Unit was first proposed by Cho et al. [34]. It can be seen as a simplification of the Long Short-Term Memory Unit in that it does not use explicit cell states. Additionally, in contrast to the Long Short-Term Memory which directly controls the number of information changed in the hidden state using separate forget and output gates, the Gated Recurrent Unit uses a single reset gate to achieve this (Aggarwal, 2018:313). A Gated Recurrent Unit contains only an update and reset gate. Gated Recurrent Units provide results comparable to that of Long Short-Term Memory units, but use fewer parameters, making it faster to train [35].

Bidirectional Recurrent Neural Networks

A modification to standard recurrent neural networks is bidirectional recurrent neural networks. A bidirectional recurrent neural network learns the input sequence both forwards and backwards [16]. Bidirectional Recurrent Neural Networks were initially proposed by Schuster and Paliwal [36] who displayed that training regression and classification data in both a positive and negative direction provided greater accuracy than just in the positive direction. It works by splitting the state neurons of a regular Recurrent Neural Network in a part that is responsible for positive time direction, forward states, and a part for negative time direction, negative states. The outputs from each state are not connected to each other. As such, Bidirectional Recurrent Neural Networks can be trained using the same algorithms as regular, unidirectional Recurrent Neural Networks.

Convolutional Neural Networks

Convolutional neural networks were designed to efficiently handle image data. They work by extracting features from raw data (in the case of image data, pixel values) that are directly useful in their application cases. This ability has since been applied to time series forecasting as well.

Convolutional Neural Network with Long Short-Term Memory (CNN-LSTM)

The CNN-LSTM architecture is the first of two hybrid architectures that incorporates aspects of both the CNN and RNN architectures. The Convolutional Neural Network layers are used for feature extraction whilst the Long Short-Term Memory layers are used for forecasting. The CNN is used to interpret sequences of input, which in turn is provided as a sequence to the LSTM to interpret. This is appropriate for forecasting share data as the CNN layer would thus detect underlying patterns that may exist in the data whilst the Long Short-Term Memory layer would then take the output from the CNN layer to make a forecast [37].

ConvLSTM

In contrast to the standard CNN-LSTM, the ConvLSTM neural network performs the convolution operation directly in the LSTM network, i.e. the LSTM unit reads input data using the convolutional process of a CNN. Not only does the ConvLSTM take the merits of a CNN network to mine the spatial features, but it captures the temporal features simultaneously [38]. The CNN-LSTM neural network, on the other hand, first completes feature extraction on the CNN layer prior to the LSTM interpreting the output of the CNN layer.

2.4 Summary

This chapter has evaluated background literature on forecasting stock prices. It did so by firstly considering the financial aspect of stocks, presenting what stocks are, how financial markets work, and the investment strategies utilised by investment managers. It then presented different traditional statistical forecasting methods and how they worked. Thereafter, it introduced theory related to machine learning, its components and how it works with specific emphasis on neural networks. Finally, it focused on specific neural network architectures. The application of these architectures to share price forecasting is further evaluated in the following chapter.

Chapter 3

Literature Review

This chapter contains an evaluation of related work pertaining to the use of machine learning in finance, with specific application in forecasting share prices. An evaluation of research on architecture analysis is performed. An evaluation of whether machine learning models have historically outperformed classical statistical forecasting techniques is presented, along with research done on the forecasting accuracy of machine learning overall. Thereafter, an evaluation of existing research performed focussing on recurrent neural networks to forecast share prices is presented, with specific emphasis on work performed over different time horizons. Research on the performance of neural networks incorporating bidirectionality is then assessed to determine whether incorporating bidirectionality improves forecasting accuracy. Finally, a brief analysis of the research performed on the application of machine learning on the Johannesburg Securities Exchange is presented.

3.1 Architecture Analysis

The purpose of this section is to analyse research that has been completed on the architecture and structure of machine learning implementations utilised in forecasting stock prices. Its intention is to obtain an understanding of how to best configure the overall architecture to obtain the most accurate forecasts.

Gandhmal and Kumar [39] conducted a study of stock market prediction techniques which consisted of reviewing 50 research papers to analyse which methodologies, datasets, software tools and performance evaluation measures are used most widely in stock market forecasting. They found that the Bayesian model, Fuzzy classifier, Artificial Neural Networks and Support Vector Machine classifiers are most commonly used. The most

commonly used software tools were MATLAB and TensorFlow with work performed in Python and JavaScript. They found that the performance metrics most commonly used by the research papers they evaluated were the Mean Absolute Percentage Error, the Root Mean Square Error, accuracy, the Mean Squared Error, the Mean Absolute Error and precision.

Ullah et al. [40] set out to identify the most accurate configuration of a Recurrent Neural Network for forecasting stock prices on the Pakistan Stock Exchange. They found that the Recurrent Neural Network architecture that produced the most accurate forecasts utilised the Rectified Learning Unit activation function, the Mean Squared Error loss function and a learning rate of 0.0002. Further, with regard to feature selection, they found that adding technical indicators which are a fundamental part of technical analysis, like the simple moving average, exponential averaging and the average directional index, did not improve model performance and that using the historical closing prices on their own provided the greatest forecasting accuracy.

Shynkevich et al. [41] performed a study on the impact of varying the input window length on the forecasting accuracy of stock prices, setting out to identify the optimal combination of input window length and forecast horizon. Their research was performed on three machine learning architectures, the Support Vector Machines (SVM), Artificial Neural Networks (ANN) and k-Nearest Neighbours (kNN), which were utilised to forecast the directions of future price movements. The dataset utilised in the study was the daily stock prices of fifty stocks listed on the New York Stock Exchange for a ten year period. They found that for each forecasting horizon, the highest prediction performance was reached when the input window length was approximately equal to the forecast horizon. For the minimum forecast horizon (one day), the input window that provided the greatest forecasting accuracy was the minimum input window used (three days). This relationship held throughout the experiments performed, with the longest forecast horizon, thirty days, being most correctly forecasted with an input window size of thirty.

Based on the research performed in this section, it is noted that the most recent stock prices on their own appear to be the best indicator of future stock prices when machine learning algorithms are used.

3.2 Research on the utilisation of machine learning for forecasting timeseries data

Extensive research has been conducted on the value provided by machine learning for forecasting time series data. This section analyses the forecasting accuracy of an array of both machine learning and classical statistical methods for forecasting time series data to determine which techniques have historically provided the greatest degree of forecasting accuracy.

Makridakis and Spilliotis [3] evaluated the performance of popular machine learning algorithms in comparison to eight traditional statistical methods to forecast time series data across multiple horizons. As part of their research, the authors, evaluated pre-existing research comparing the accuracy of machine learning models to that of traditional statistical models. They found that these studies have generally led to mixed results. However, a characteristic common to all these studies was that a limited number of datasets were used in their comparisons.

To measure the accuracy of their models, they used the symmetric Mean Absolute Percentage Error and the Mean Absolute Scaled Error. Computational complexity, being the time needed to train a given model and use it for extrapolation, was measured simply as the mean computational time required by the model to predict a time series divided by the corresponding time needed by the Naïve method to achieve the same task.

As pre-existing research provided conflicting conclusions on whether pre-processing data increased the accuracy of machine learning models, Makridakis and Spilliotis [3] applied various pre-processing methods on a one-step Multi-Layer Perceptron Model and then applied the most accurate pre-processing method on all machine learning models. They found that both the Box-Cox and logarithmic methods do not improve accuracy by much. They also found that seasonal adjustments provided significantly better results and that removing the trend from data using a linear function provides more accurate results than using a first difference. However, they concluded that more work needs to be performed on the best approach to pre-processing, particularly to determine whether the same conclusions found with a multi-layered perception would be found using other machine learning models.

They used the models to forecast the next 18 time steps for a collection of different time series data. They grouped these forecasts into short term (1 to 6 time steps ahead), medium term (7 to 12 time steps ahead) and long term (13 to 18 time steps ahead). They found that traditional statistical forecasting models were more accurate than machine

learning models across all forecast horizons as well as more computationally efficient. Their results are visualised in the below graph:

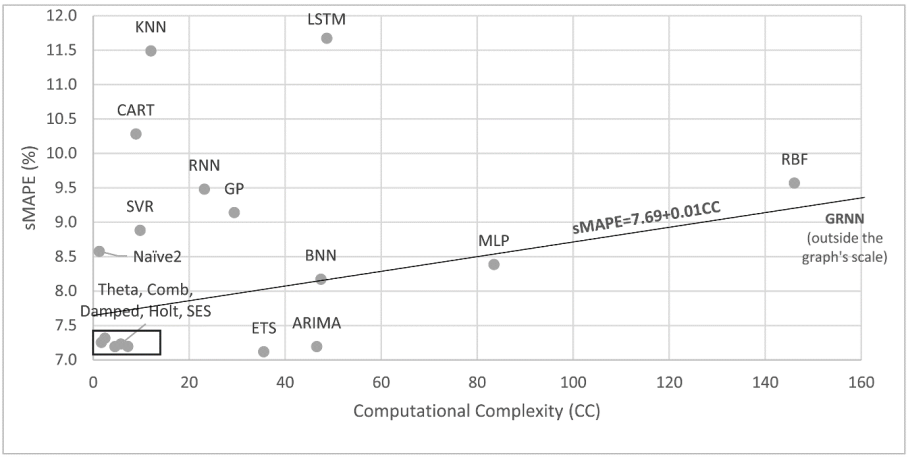

FIGURE 3.1: Results of the study performed by Makridakis and Spilliotis [3] indicating the symmetric Mean Absolute Percentage Error and Computational Complexity of the models investigated.

In their results, which are visualised in Figure 3.1, it was clearly observed that not only was classical statistical forecasting techniques much more computationally efficient when compared to machine learning algorithms, but they exhibited superior forecasting accuracy as well. The machine learning technique that provided the greatest degree of forecasting accuracy was the Bayesian Neural Network, followed by the Multi-Layer Perceptron and the Support Vector Regression.

Soni [5] conducted a research survey to analyse the applications of neural networks to stock price forecasting. They found that neural networks are significantly more accurate than other competitive models and algorithms, including multiple linear regression analysis, in forecasting stock prices. Further, they found that a vast range of different stock market parameters were used in stock price forecasting using neural networks, including market index movements, fundamental analysis, moving averages, stock price values and daily returns, among others.

Balaji, Ram and Nair [42] designed fourteen different deep learning models to forecast the closing prices of shares included in the S&P BSE-BANKEX index over the period 12 July 2005 to 3 November 2017. The results of their study were measured using the Root Mean Squared Error, the Median Absolute Percentage Error as well as the Directional Accuracy, which measures only the accuracy of the movement direction of the stock price forecast. They found that an Extreme Learning Machine, a Long-Short Term machine with two hidden layers, a Convolutional Neural Network with three hidden layers and a Gated Recurrent Unit with three hidden layers performed best among the fourteen models they constructed. Additionally, they found that Gated Recurrent Units provided

the lowest Median Absolute Percentage Errors for shorter forecast horizons whilst the Extreme Learning Machines provided the lowest Median Absolute Percentage Errors for longer forecast horizons.

Samarawickrama & Fernando [43] evaluated different recurrent neural network models' forecasting accuracy in predicting daily stock prices of selected companies on the Colombo Stock Exchange. They used Feedforward Unidirectional Recurrent Neural Network, Gated Recurrent Unit and Long Short-Term Memory architectures. Closing, High and Low prices for the past two days were utilised as input variables. They found that the LSTM model provided the best forecasting accuracy whilst the GRU performed worst.

Hiransha et al. [44] conducted a study on the accuracy of four deep learning architectures (Multilayer perceptron, Recurrent Neural Network, Long Short-Term Memory and Convolutional Neural Network) in forecasting stock prices for stocks listed on the National Stock Exchange of India and the New York Stock Exchange. They found that the Convolutional Neural Network model outperformed the other machine learning architectures used. They further found that the Convolutional Neural Network was capable of capturing abrupt changes in the underlying data. This is consistent with the findings of Selvin et al. [45] who performed a similar study on stocks listed on the National Securities Exchange of India using Recurrent Neural Network, Long Short-Term Memory and Convolutional Neural network architectures and found that the Convolutional Neural Network architectures outperformed the other machine learning architectures. They concluded that this was due to the sudden changes that were present in the stock market which may not follow a regular pattern or the same cycle as both the Recurrent Neural Network and the Long Short-Term Neural Network are time sensitive.

Guan and Lu [46] evaluated the forecasting accuracy of a CNN-LSTM model compared to a simple LSTM, k Nearest Neighbours and linear Regression. Their dataset consisted of the S&P500 Index as well as the stock prices for Intel, Tesla, Macy's and Celgene for the period January 2000 to April 2020, with 90% used as the training set and 10% as the testing set. They also incorporated other economic indicators, including market indicators and exchange rates, into their model in order to evaluate whether these improve forecasting accuracy. They found that their CNN-LSTM model's results were not significantly better than the other models used for next day forecasting, but their model did perform significantly better for the ten day average movement forecast. They noted that this was due to the CNN-LSTM model being able to capture the underlying trends much better than the simple LSTM model. This was attributed to the CNN layers being activated by a variety of input patterns and then passing processed information to the downstream LSTM layers. This is consistent with the observations of Hiransha

et al. [44] and Selvin et al. [45] who both highlighted the ability of CNN's to identify underlying patterns in the data. Guan and Lu [46] further found that adding additional technical and economic indicators to their dataset, in addition to the historical share prices, did not impact on the performance of their models for next day prices, though an improvement in performance was observed for longer time horizons.

The research contained in this section covers a broad range of data sets and forecasting techniques. Makridakis and Spilliotis [3] found that classical statistical forecasting methods outperformed machine learning algorithms in forecasting time series data. Of the machine learning techniques analysed, they noted that the Bayesian Neural Networks, followed by the Multi-Layer Perceptron and the Support Vector Regression, followed by the Recurrent Neural Network. In their study, they found that the Long Short-Term Memory Network provided the worst forecasting accuracy. This is in contrast to Balaji, Ram and Nair [42] who observed that the LSTM model was one of the most accurate models analysed in their study. However, what differentiates their LSTM model from Makridakis and Spilliotis's is that they included two hidden layers as opposed to one, indicative that increasing the amount of hidden layers included may increase forecasting accuracy as the additional hidden layer allows the model to better understand the nuances present in the underlying data. However, simply adding complexity to the machine learning model does not necessarily improve forecasting accuracy. This was found to be the case by Guan and Lu [46] who observed that the forecasting accuracy of a CNN-LSTM model was not significantly better than the simpler neural network models. Overall, Soni [5] found that neural networks outperformed alternative machine learning techniques, which is in line with the rest of the research papers analysed in this section.

3.3 Research on Neural Networks for Share Price Forecasting

Multiple approaches have been taken in assessing the use of recurrent neural networks to forecast share prices for different time horizons.

3.3.1 Trend forecasting

Parray, Khurana and Kumar [47] performed an evaluation of the forecasting accuracy of a support vector machine, a perceptron and logistic regression for the shares included in the Indian Stock Exchange's NIFTY50 index for the period 1 January 2013 to 31 December 2018. Their research focused only on forecasting the trend of share price movements for the following day, using three historical observations as the input. Trend

forecasting refers to only forecasting the direction (up or down) of the share price. The support vector machine was able to correctly forecast the trend 87.35% of the time, whilst the perceptron's accuracy rate was 75.88% and the logistic regression's 86.98%.

3.3.2 Forecasting next day closing prices

Gao and Chai [48] built a model using a Long Short Term Memory architecture to forecast next day closing prices, i.e. the price the shares would trade at, at the close of business on the following day. Their input data consisted of the daily basic trading data set (opening price, closing price, high price, low price, trading volume and the adjusted price, which is the closing price adjusted for distributions and corporate actions) for companies and 15 stock market technical indicators. Their first dataset contained S&P 500 companies' data for the period 3 January 2000 to 10 November 2016, and their second dataset contained NASDAQ-listed companies' data for the period 2 January 2004 to 15 August 2016. The data was normalised using min-max normalisation. Glorot uniform initialisation was used to generate random weights and biases. The Adam optimisation algorithm was used as Gao and Chai felt that this algorithm was relatively computationally efficient with few memory requirements. To demonstrate the utility of their system, they tested seven models: Moving Average, Estimated Moving Average, Autoregressive Moving Average (ARMA), Generalised Autoregressive Conditional Heteroscedasticity (GARCH), Support Vector Machine (SVM), Feed Forward Neural Network and Long Short-Term Memory. They found that when compared to other machine learning models, the Long Short Term Memory model combined with basic trading data, technical indicators and Principal Component Analysis were more accurate at forecasting next day closing prices.

3.3.3 Next day opening prices

Qiu, Wang and Zhou [6] built a Long Short-Term Memory model with an attention mechanism to forecast next day opening prices, i.e. the price the share would trade at, at the start of the following day. The data used for their experiment consisted of three stock indices. The S&P 500 and Dow Jones Industrial Averages were used for the period 3 January 2000 to 1 July 2019 and the Hang Seng Index was used for the period 2 January 2002 to 1 July 2019. The input data into their model consisted of the opening price, closing price, high price, low price, adjusted closing price and the trade volume. The data was normalised using zero mean normalisation. The data was denoised using the coif3 wavelet transformation method. This consisted of three decomposition layers. This was evaluated using the signal to noise ratio and the root mean square error:

A higher signal to noise ratio and a lower root mean square error indicated a better denoising effect of the wavelet transformation. They found that their model was more accurate at forecasting the next day opening prices as compared to a standard Long Short Term Memory model (i.e., without an attention mechanism), a Long Short-Term Memory Model with wavelet transformation only, and a Gated Recurrent Unit. They also indicated that they would like their future work to incorporate data predictions based on stock-related news.

3.3.4 Next 12 days closing prices

Jahan and Sajal [49] used the closing price of one stock, Advanced Micro Devices, for 168 days to forecast the losing price for the stock for the subsequent 12 days using a recurrent neural network with Long Short-Term Memory architecture, i.e. the price the stocks would trade at, at the end of the subsequent twelve business days. They found that their model had a less than 5% error percentage. They noted that they would like to incorporate investor sentiment into their future work.

3.3.5 Bidirectional Recurrent Neural Networks

Althelaya et al. [35] performed an evaluation of whether bidirectional Long Short-Term Memory provide greater forecasting accuracy for stock market prices over both a long term and short term horizon when compared to standard Long Short-Term Memory units. Their research was done on the S&P500 Index for the period 1 January 2010 to 30 November 2017. They found that Bidirectional Long Short-Term Memory Recurrent Neural Networks provided greater forecasting accuracy than the single directional Long Short-Term Units. Further, they found that the gap between the forecasting accuracy of Bidirectional and singular directional networks grew over longer time periods, i.e., the forecasting accuracy of singular directional neural networks degraded at a faster rate than bidirectional neural networks the further into the future the forecast was made.

Althelaya et al. [35] performed further analysis of the forecasting accuracy of multiple configurations of Single Direction and Bidirectional Long Short-Term Memory and Gated Recurrent Units in forecasting stock prices. The same dataset utilised in their previous research, the S&P500 Index over the period 1 January 2010 to 30 November 2017, was utilised. However, instead of only utilising closing prices in the input set, they utilised the closing price, opening price, high price, low price and volume of shares as part of their input set. The time series data was also differenced and normalised using min max normalisation. Their research used a window time of 10 historical observations

to forecast one day ahead (short term) and thirty days ahead (long term). A Multi-Layer Perceptron was utilised as a benchmark. All Gated Recurrent Units and Long Short-Term Memory Units performed better than the Multi-Layer Perceptron. Further, the bidirectional models outperformed their singular directional counterparts, whilst the Bidirectional Long Short Term Memory model provided greater forecasting accuracy as compared to the Gated Recurrent Unit's.

Siami-Namini et al. [50] performed a study on whether a Bidirectional Long Short-Term Memory Unit architecture outperformed a single directional architecture. Their study was performed over various market indexes as well as the closing share price of IBM for the period July 2009 to July 2019. Their data set was split 70% for training and 30% for testing. They found that the Bidirectional LSTM was, on average 37.78% more accurate in forecasts than the single directional LSTM model. It was noted that training based on the Bidirectional LSTM model was slower than the single directional LSTM model.

Huynh, Dang and Duong [31] developed a new model for predicting stock price movement directions using a Bidirectional Gated Recurrent Unit (BGRU). Their research was performed over both the S&P500 Index as well as individual stocks over the period October 2006 to November 2013. The forecasting accuracy of the Bidirectional Gated Recurrent Unit was compared to the forecasting accuracy of a Long Short-Term Memory Unit and a standard Gated Recurrent Unit. The proposed Bidirectional Gated Recurrent Unit provided a greater forecasting accuracy as compared to the standard Gated Recurrent Unit and the Long Short Term Memory unit.

3.3.6 The Application of Machine Learning algorithms to JSE-listed securities

In a survey review performed, Henrique et al. [51] found that the vast majority of research performed on the application of machine learning for stock price forecasting was performed within a North American context. Limited research has been conducted on the forecasting accuracy of machine learning for stocks listed on the Johannesburg Securities Exchange. Drue [52] found that an investment portfolio of JSE-listed instruments built using a Support Vector Machine outperformed portfolios built using Factor Based Investment portfolios used in traditional finance. The Support Vector Machine was set up to perform a classification task, categorising shares as over or under performers, being shares that provided positive and negative excess returns, respectively. Support vector machines are more appropriate for classification problems as opposed to regression problems.

Balusik, Magalhaes and Mbuvha [53] evaluated whether a Long Short-Term Memory Network outperformed the Seasonal Autoregressive Integrated Moving Average in forecasting the intraday directional movements and closing price of the JSE Top 40 Index. They found that their LSTM model was able to forecast the movement of the JSE Top40 Index with 50% accuracy as compared to an accuracy of 44.3% for the Seasonal ARIMA model.

3.4 Summary

This chapter has evaluated extensive existing research in using neural networks for time series forecasting, with specific focus on share price forecasting. It has found that different types and configurations of neural networks do indeed provide predictive value in forecasting share prices, and that bidirectionality improves the accuracy further. It has also found that very limited research has been performed thus far in analysing whether using machine learning can be used to forecast share price movements of JSE-listed shares specifically.

Based on the review performed, it was concluded that for this study, a standard recurrent neural network, a Long Short-Term Memory neural network and a Gated Recurrent Unit neural network would be built for JSE-listed securities, using both single and bidirectional models. These ensured that the full spectrum of recurrent neural networks have been evaluated within a South African context. The size of the input window was varied as well to evaluate whether a linear relationship does exist between the input window and the forecasting horizon as identified by Shynkevich et al. [41]. Finally, experiments were performed based on using both the historical price on its own as well as the historical price along with the market index. This was to assess whether, in the context of the Johannesburg Securities Exchange, using the historical price on its own provides greater forecasting accuracy than when it is coupled with the market index, as found by Ullah et al. [40] with research on the Pakistan Stock Exchange.

Chapter 4

Experimental Framework and Datasets

This chapter contains a summary of the data used to conduct the experiments and the computing environment in which the experiments were performed.

4.1 Data Sets Used

The data used in this study is limited to the shares that are included in the Johannesburg Securities Exchange's Top 40 Index (referred to as the 'JSE Top40'). The share data was obtained from the Reuters' Refinitiv Eikon financial data platform. Data was extracted for each share included in the Johannesburg Securities Exchange's Top 40 Index as at 21 September 2020, according to the Reuters Refinitiv Eikon platform. The data extracted for each company is as follows: The exchange date, the closing price, the opening price, the high price and the volume. The data extracted covered the period 4 January 2010 to 30 May 2020. However, in instances where the date on which the stock listed on the Johannesburg Securities Exchange is after 4 January 2010, the data was extracted from the first available date.

Initially, all neural network models were run across all 40 shares included in the JSE's Top 40 index, using the period 2 January 2019 to 31 December 2019 as the training set whilst the period 2 January 2020 to 29 May 2020 was used as the testing set. The performance of the JSE's All Share Index, which captures the share price performance of all shares listed on the JSE, is included above. As evidenced above, it is noted that the overall performance of the market remained relatively stable throughout the 2019 year

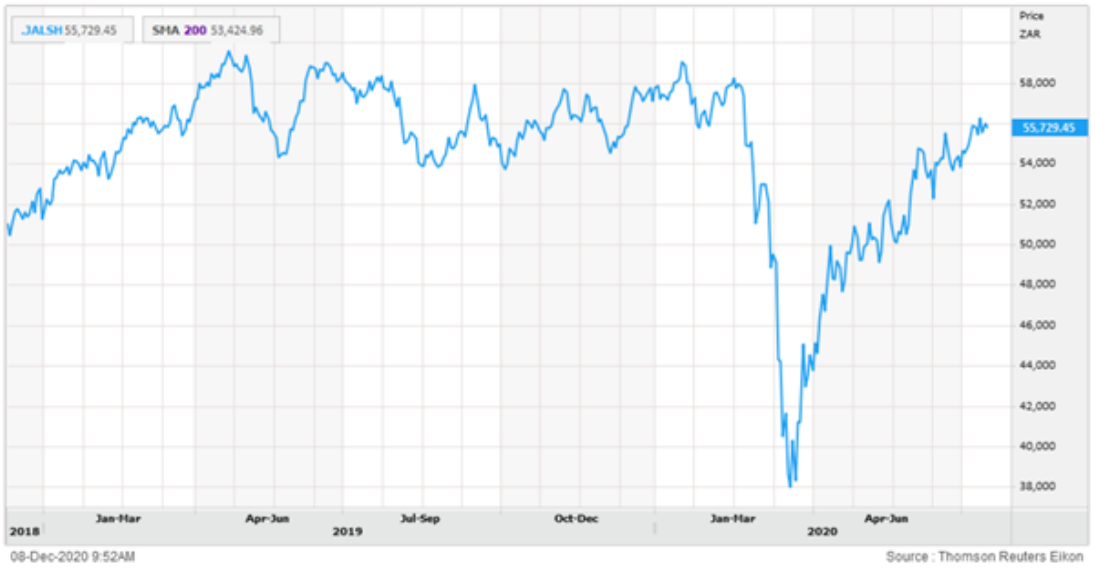

FIGURE 4.1: JSE All Share Index

before falling sharply with the advent of the Covid-19 pandemic in 2020 before slowly recovering.

As a result of the above noted unforeseen external factors' impact upon the market, using only the 2019 year as the basis for which the models were trained to forecast 2020 share prices produced models that were not highly accurate. Consequently, the same models were then restructured as walk forward validation models. This means that the entire model is retrained at each individual time step with the model being retrained with all historical data from the start date, 2 January 2019, up to the day before the forecast date. Using walk forward validation uses significantly more computational time than simply building a model once and making forecasts using the same model. This computational time was measured using Python's built in timeit.timeit function.

4.2 Experimental Framework

The financial data noted above were exported from a Reuters Refinitiv Eikon terminal into a series of Comma Separated Values (.csv) files which were then uploaded to an individual Google Drive. This Google Drive containing the financial data was then mounted to a Google Collaboratory Python 3 notebook in which the experiments were performed. This notebook utilises a Tensor Processor Unit v2 with 8 cores. The relevant share data for individual stocks were read into a Pandas Data Frame at the start of the execution of individual experiments. Neural networks were built using both the Tensor Flow and Keras libraries within Python, whilst Statsmodels for traditional statistical models, and Matplotlib and Sci-Kit Learn were used for model evaluation.

Ten time steps were forecast and assessed against the actual share prices. Additionally, the number of historical observation inputs used varied as 4, 5, 6, 10 and 15 inputs were used for each model to assess whether adding additional historical inputs provided greater forecasting accuracy. The recurrent and hybrid neural networks were run using both the share price alone and the share price in conjunction with the market index on the same dates as the input datasets. The accuracy of the machine learning models were measured by calculating the Mean Absolute Percentage Errors at each forecast time step between the forecast share price and the actual share price. Additionally, the gradient of the change in the Mean Absolute Percentage Errors were calculated as well. The purpose of this was to aid the assessment of the degradation in the Mean Absolute Percentage Error as additional future time steps were forecast.

4.3 Statistical Python Implementations

4.3.1 Naïve Method

A simple Python for loop was run on the closing prices of each stock individually, performing a simple differencing exercise starting at one time step and ending at a time step difference of fourteen. That is, the forecasted price at time step t would be exactly equal to the historical actual price at time step (t-n) where n increases from one to fourteen. A range of fourteen was utilised to evaluate the change over time and to inform the number of historical inputs to utilise when building machine learning models.

4.3.2 Autoregressive Integrated Moving Averages

The Arima model from Python's StatsModels library was used. A single degree of differencing was applied to the model to make the model stationery. This model was used to make one day forecasts based on 5 historical observations.

4.3.3 Holt Winter's Exponential Smoothing

The simplified version of this exponential smoothing model was utilised from Python's StatsModels library.

4.3.4 Theta Method

The model was built in Python using the Seasonal Decomposition and Simple Exponential Smoothing models from the StatsModels library as well as the Linear Regression Model from the Sci Kit Learn Library. Different configurations of the Theta Model were run, varying the number of historical observations included used as the model's inputs (five to fourteen observations were used), and varying the interval being forecast from forecast one to forecast five.

4.4 Machine Learning Python Implementations

All neural network models noted below utilises the rectified linear unit (Relu) activation function, the Adam stochastic gradient descent optimiser and Mean Squared Error loss function. These models were all trained using 100 epochs.

Model 1: Random Forest The RandomForestRegressor function from SciKit Learn was utilised as a Walkforward validation model. Given the nature of the data to be trained, the n_estimator parameter was set to 1 000. This means that 1 000 trees will be used in the forest.

Model 2: Feedforward Neural Network A feedforward neural network model was built with a single hidden layer with 100 nodes and a single dense output layer with a single node.

Model 3: Convolutional Neural Network As a starting point, a one-dimensional Convolutional Neural Network model was built. The first hidden layer, which was the standard Conv1D layer, consisted of 64 filters and a kernel size of 2. This was then followed by a pooling layer, a flatten layer to reduce the feature maps to a single one-dimensional vector, which in turn was followed by a dense layer that would interpret the features extracted by the model and distil this into a single forecasted stock price. This model was initially built on a single univariate stream of input data, being the historical stock prices for the stock in question.

Model 4: Simple Recurrent Neural Network Models TensorFlow's SimpleRNN, LSTM and GRU layers were utilised. These models each consisted of a single hidden layer, with 100 nodes. These were then assessed on their own before being wrapped in a Bidirectional layer to assess whether adding bidirectionality would improve forecasting accuracy.

Model 5: CNN-LSTM The CNN part of the model starts with a time-distributed convolutional layer for initially reading the input data. This layer consisted of 64 filters and a kernel size (convolutional window length) of 2. This is then followed by a time-distributed max pooling layer, with a pool size of 2, to distil the data down to its most important features. This is then flattened down to a single one-dimensional vector which was to be used as the input into the LSTM layer. The LSTM then reads this data and makes a forecast.

Model 6: ConvLSTM The ConvLSTM2D layer was utilised to read and process the input data. This layer utilised 64 filters and had a kernel size of (1, 2). This was then flattened into one dimension before a dense layer used to forecast.

4.5 Summary

This chapter described the experimental framework for the study. The computational environment in which the experiments were conducted was presented first. This was followed by a description of the inputs and outputs that was to be used in each model, including an analysis of the market conditions that existed and drove the patterns present in the input dataset. Finally, the manner in which each model would be built in the Python programming language was introduced.

Chapter 5

Statistical Forecasting Evaluation

Prior to applying machine learning models to the data, exploratory analysis of the data was performed to obtain an understanding of the data as well as to identify any trends and nuances found in the data. The experiments performed related to statistical forecasting techniques and their results are presented in the following chapter. Four statistical techniques were applied to the stock price data, i.e., a naïve method, Autoregressive Integrated Moving Averages (ARIMA), simple exponential smoothing and the theta method. The numerical results of these experiments are presented in Appendix ATable 1 (Naïve method, ARIMA, Simple Exponential Smoothing, Theta Method), Table 2 (Detailed Naïve method), and Table 3 (Detailed Theta method) of the Appendix. This Appendix Table contains the individual Mean Absolute Percentage Errors for the stock prices utilised, as well as overall averages of these Mean Absolute Percentage Errors for each traditional statistical method. The outcomes of these analyses were analysed as a baseline when evaluating the outcomes of the machine learning models applied later on.

5.1 Naïve method

A naive method was applied as a simple benchmark against which the predicted values of other methods were compared. This method consists solely of using the value at time step n-x to forecast the value at time step n where x represents the forecast horizon. This was used to analyse the degree of momentum in day to day stock price changes. The method was applied over a 1 business day to 14 business day time horizon in order to determine how quickly the predictive value of this method deteriorates over time.

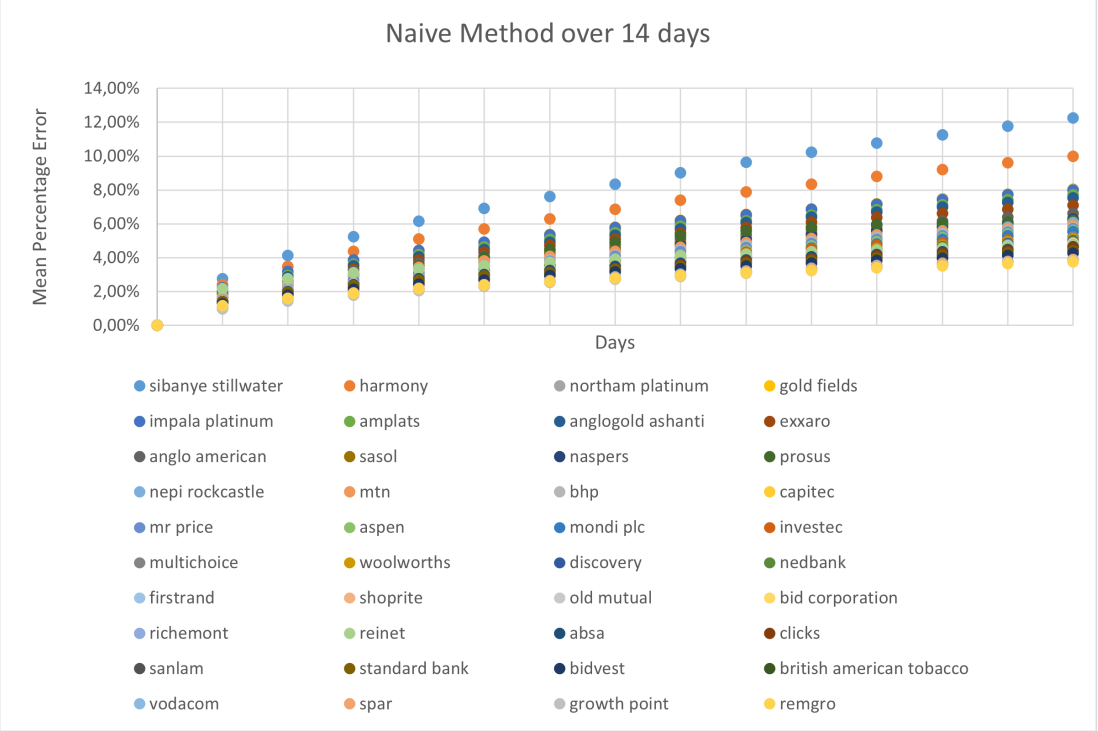

FIGURE 5.1: Naïve method results

5.1.1 Findings

The Mean Absolute Percentage Errors observed for all stocks grew steadily over the fourteen-day time horizon with no stock showing any reduced error at a later time step than a prior time step. When forecasting 1 timestep, the model produced an average Mean absolute Percentage Error of 1.58% whilst the model produced an average Mean absolute Percentage Error of 5.77% when forecasting 14 timesteps into the future. Additionally, the largest increase in the observed Mean Absolute Percentage Error overall was from day one to day two. Thereafter, the day on day increase in the Mean Absolute Percentage Error slowly declines. Interestingly, it is observed that the stocks with the nine highest Mean Absolute Percentage Errors were all from the mining sector. Refer to Table A.2 in the Appendix for the average Mean absolute Percentage Errors produced by this model at all timesteps.

5.2 ARIMA

The ARIMA model implemented was the Rolling Forecast ARIMA Model. This model works by going through the dataset from the starting point, consistently forecasting the next time step, using a predefined number of prior time steps to, comparing the forecast

to the actual and adding the mean squared error to the model's total mean squared error.

The model has been applied to only to closing price data of the JSE's Top 40 Stocks. It evaluated how accurately the ARIMA model was able to forecast the next closing price based on the prior five closing prices.

5.2.1 Findings

The Arima Model was unable to produce forecasts for a number of stocks due to the perceived presence of seasonality in these share prices. This is because a standard Arima model does not support time series data with a seasonal component [26]. For the stock prices that were successfully forecast by the model, an average Mean absolute Percentage Error of 1.598% was observed when forecasting the next day closing price. The errors observed were in line with the errors observed for the Theta Method noted below.

5.3 Theta Method and Holt Winter's Exponential Smoothing

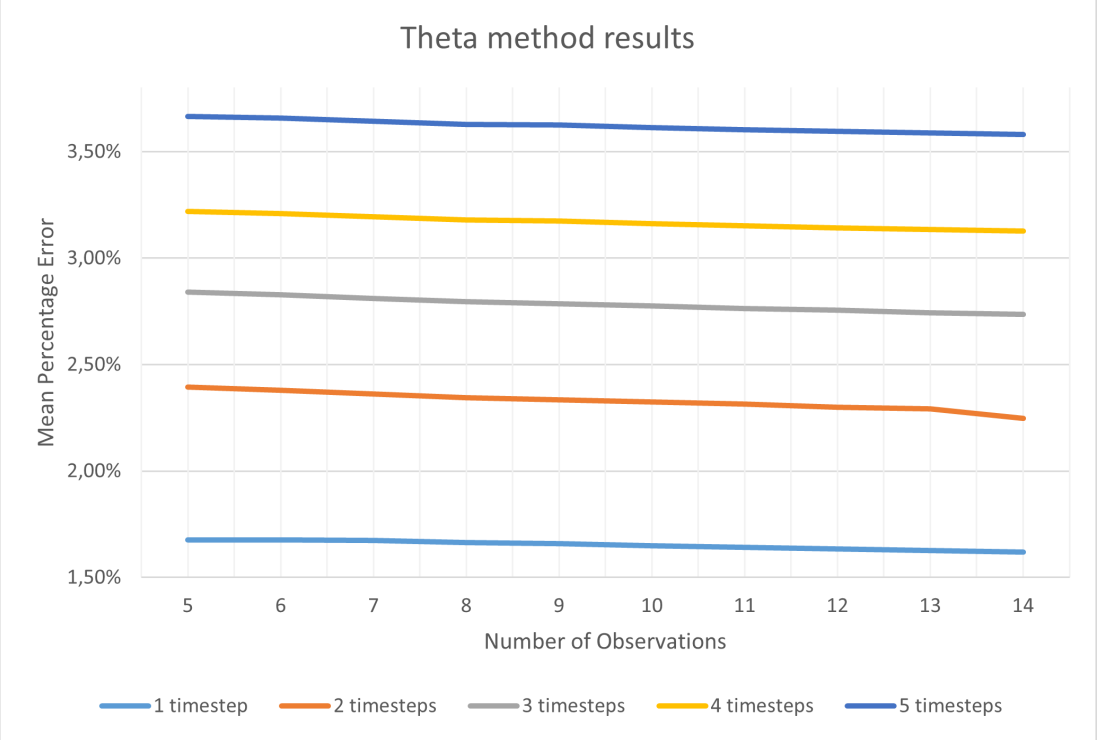

FIGURE 5.2: Theta Method Results

Across all forecast intervals, the observed errors very slowly declined when more historical observations were used in the forecast. The 1 time-step forecast produced an average Mean absolute Percentage Error of 1.676%, which declined very slightly to 1.62% when 14 historical observations were included. When forecasting five steps into the future, the model produced an average Mean absolute Percentage Error of 3.665% when 5 historical inputs were used, which declined to 3.581% when 14 historical inputs were used. As with prior statistical models utilised, it was observed the biggest change in the day to day movement in the forecast errors observed was in the movement from forecasting the share price the following day to forecasting the stock price in two days' time. Thereafter, the day to day change in the error observed tapers off. Refer to Table A.3 in the Appendix for detailed results. As with prior statistical models utilised, it was observed that forecasting stock prices in the mining industry still produces the greatest Mean Absolute Percentage Errors.

The Holt Winter's Exponential Smoothing model produced results with a Mean Absolute Percentage Error slightly higher than that of the Arima model and the Theta Method discussed above.

5.4 Summary

This chapter has documented the results of the experiments performed using classical statistical techniques. The following observations were made based on the analysis above. Firstly, considering the results of the models utilised, the Theta Model performed best. Therefore, this model would be used as a benchmark when analysing the results of the neural networks utilised. Additionally, it is observed that increasing the number of historical observations as inputs into the model increased the forecasting accuracy of the model. The forecast accuracy changes the most when moving from forecasting the stock price in one day's time to forecasting the share price in two days' time. The forecasting errors then taper off thereafter. The results obtained in this chapter were utilised as a benchmark against which the forecasting accuracy of the machine learning methods was measured.

Chapter 6

Neural Network

The neural network models developed are described and their results presented and analysed in the following chapter. A simple random forest model was also developed as a machine learning benchmark alternative to the neural network models. The first neural network model is a feedforward neural network. The results of a Convolutional Neural Network are then presented and evaluated, followed by multiple configurations of recurrent neural networks. Finally, two neural network models that were built using both convolutional and recurrent layers are presented and evaluated. All neural network models were initially trained only once using only 2019's data as the training set. Thereafter, they were trained as walk forward validation models. The recurrent neural networks were forecast for 10 time steps to further evaluate their accuracy over a longer forecast period. The numerical results of these models are contained within Table 4 (Single train univariate), Table 5 (Walkforward univariate), Table 6 (Single train multivariate) and Table 7 (Walkforward multivariate) of the Appendix where the overall Mean Absolute Percentage Errors are contained as well as the gradient of the change in the Mean Absolute Percentage Error for forecast time step 1 to forecast time step 10.

6.1 Univariate Inputs

6.1.1 Model 1: Random Forest

This model provided a Mean Absolute Percentage Error of 3.9157% for one time step forecast. This was used as a benchmark against which to measure the performance of neural networks.

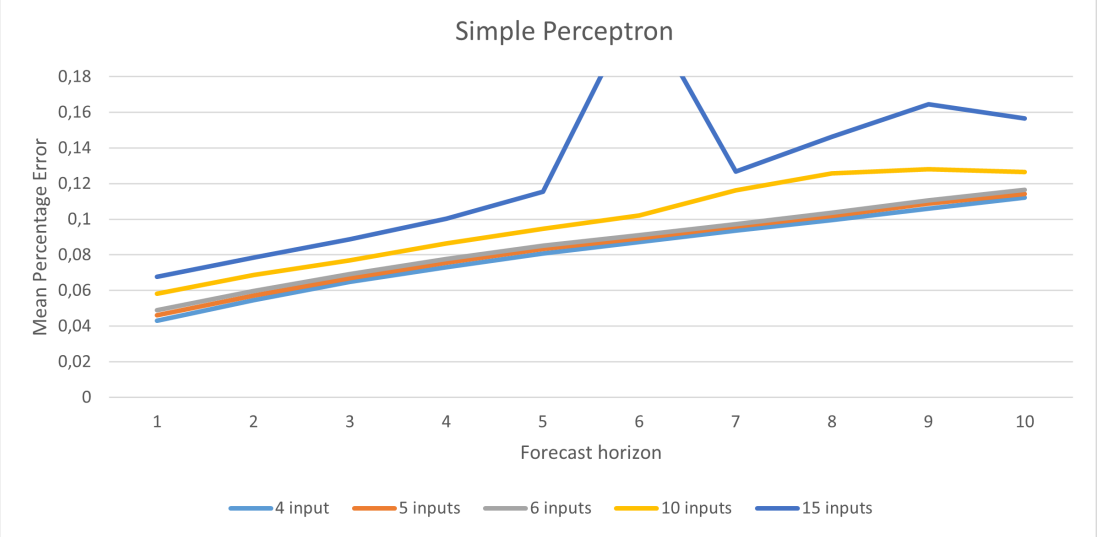

FIGURE 6.1: Feedforward Neural Network Results

6.1.2 Model 2: Feedforward Neural Network

The feedforward neural network model (walk forward) produced its smallest Mean Absolute Percentage Error for the configuration that used four historical observations as its input (4.29% for its first time step forecast). This configuration also produced the smallest gradient along its forecast window (0.007407), with the Mean Absolute Percentage Error of its tenth time step forecast error of 11.2%. As the number of historical inputs used were increased, the Mean Absolute Percentage Error observed consistently increased as well as the gradient measuring the increase in the Mean Absolute Percentage Error over the forecast window.

6.1.3 Model 3: Convolutional Neural Network

The Convolutional Neural Network model displayed the same characteristic as the feedforward neural network as the Mean Absolute Percentage Error observed increased as the number of inputs increased, though the feedforward neural network did outperform the Convolutional Neural Network across most forecast time steps and input sizes, the gradient, over the ten step forecasts is however smaller across all input sizes. Additionally, it was observed that in contrast to the feedforward neural network that had a growing gradient as the number of inputs increased, the gradient observed with the Convolutional Model declined as more input observation were added. These two effects have had the combined effect of the Convolutional Neural Network having a lower Mean Absolute Percentage Error than the feedforward neural network for models with more input observations at later forecast windows.

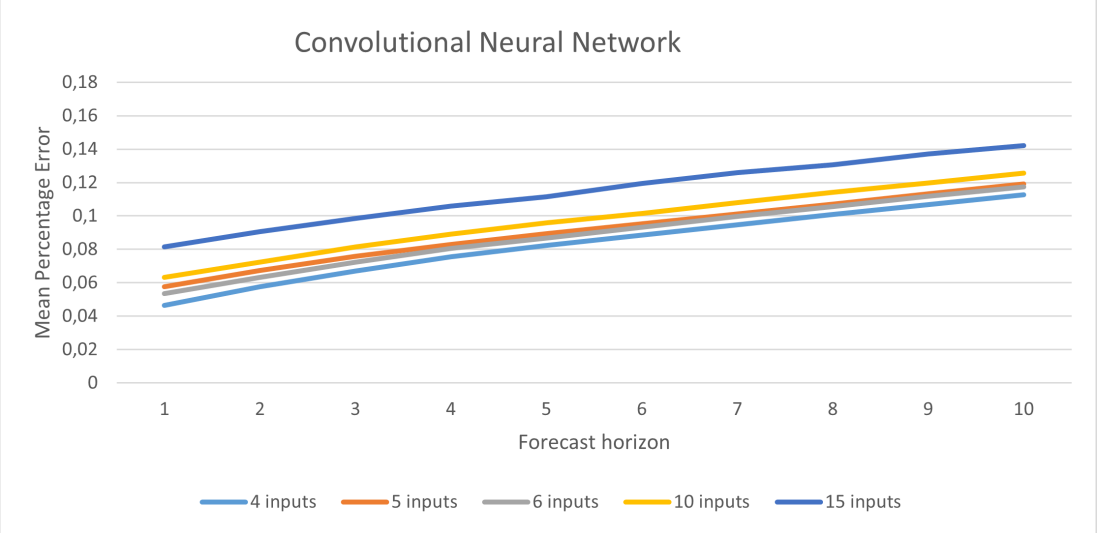

FIGURE 6.2: Convolutional Neural Network Results

6.1.4 Model 4: Recurrent Neural Networks

TABLE 6.1: Recurrent Neural Network Experimental Framework

Input configurations	2	Stock price alone and stock price with market index
Number of inputs	5	4, 5, 6, 10 and 15 historical observations
Forecast horizon	10	10 time steps were forecasted and compared to actuals for each configuration
Training styles	2	Walkforward and train once

Unidirectional Recurrent Neural Network

The Unidirectional Recurrent Neural Network produced Mean Absolute Percentage Errors that were only marginally better than the Mean Absolute Percentage Errors produced by the Convolutional Neural Networks. However, whilst the gradients of the Convolutional models reduced as more historical observations were added, the gradients of the Unidirectional Recurrent Neural Network increased.

Interestingly, the experiment of the Recurrent Neural Network that was trained once did not perform noticeably worse than the walk-forward version of the experiment.

It was observed that adding bidirectionality to the Recurrent Neural Network reduced the observed Mean Absolute Percentage Errors across the forecast window for the Recurrent model with few inputs (4, 5 and 6 historical observations). Thereafter, it was outperformed by the unidirectional recurrent model as the number of inputs were increased.

FIGURE 6.3: Unidirectional Recurrent Neural Network Results

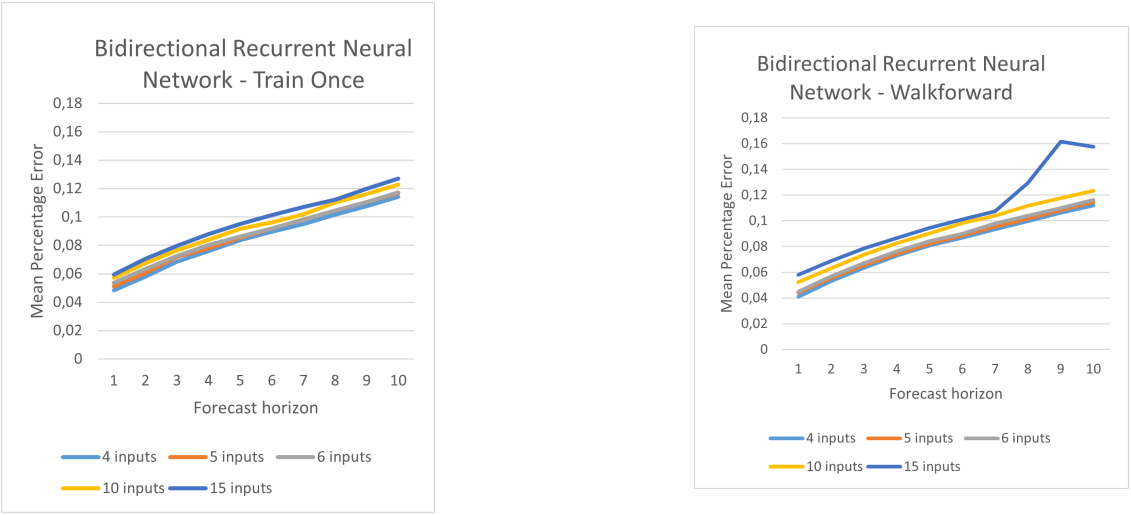

FIGURE 6.4: Bidirectional Recurrent Neural Network Results

Long Short-Term Memory

In contrast to the Unidirectional Recurrent Neural Network where the version of the model that was trained once produced comparable results to the walk forward version of the model, the Long Short-Term Memory model's train once model produced Mean Absolute Percentage Errors that were significantly poorer than the Mean Absolute Percentage Errors produced by the walk forward model. The standard LSTM model produced results that were in line with the results produced by both Recurrent Neural Network models. However, the Bidirectional LSTM model outperformed both the Recurrent Neural Network models as well as the standard LSTM Model across the forecast horizon. However, it is observed that the size of this advantage shrinks as more input observations were added to the model.

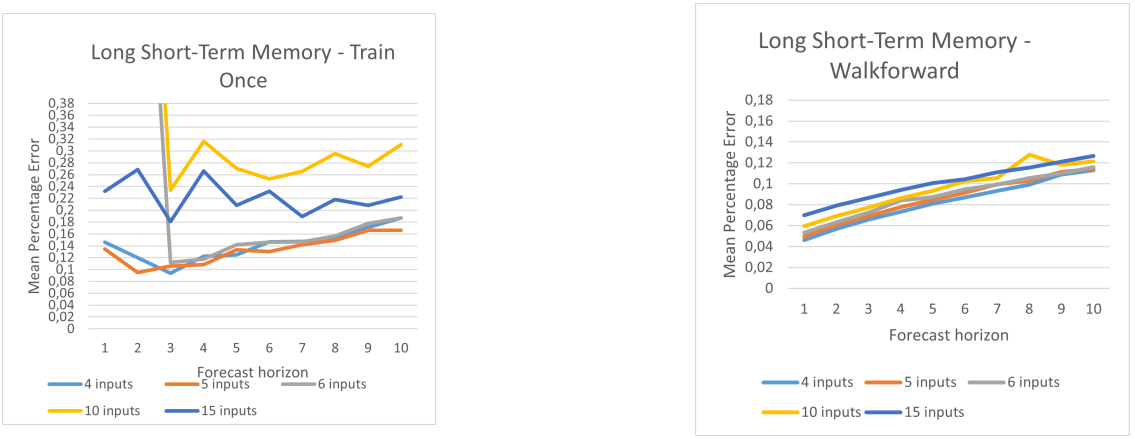

FIGURE 6.5: Unidirectional Long Short-Term Memory results

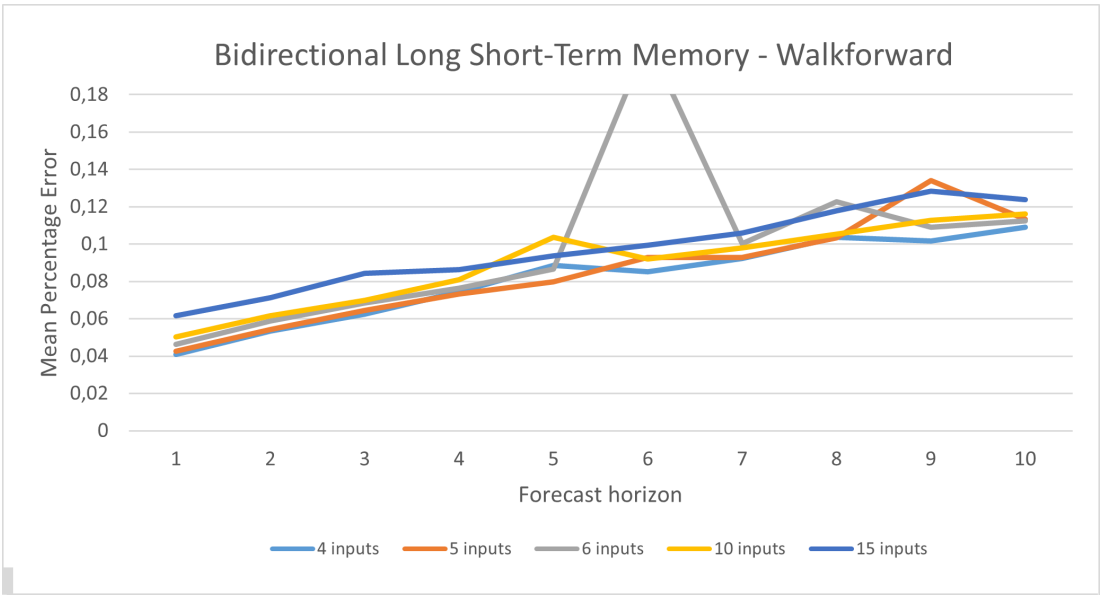

FIGURE 6.6: Bidirectional Long Short-Term Memory - Walkforward

Gated Recurrent Unit

The Gated Recurrent Unit displayed the same characteristic that was prevalent in other models in that the walk forward model outperforms the models that were trained once. However, in contrast to the other models, it was observed that the Bidirectional Gated Recurrent Unit improved upon the Mean Absolute Percentage Errors observed on the Simple Gated Recurrent Unit model, whilst its Mean Absolute Percentage Errors actually performed worse when compared to the Simple model once additional input units were added.

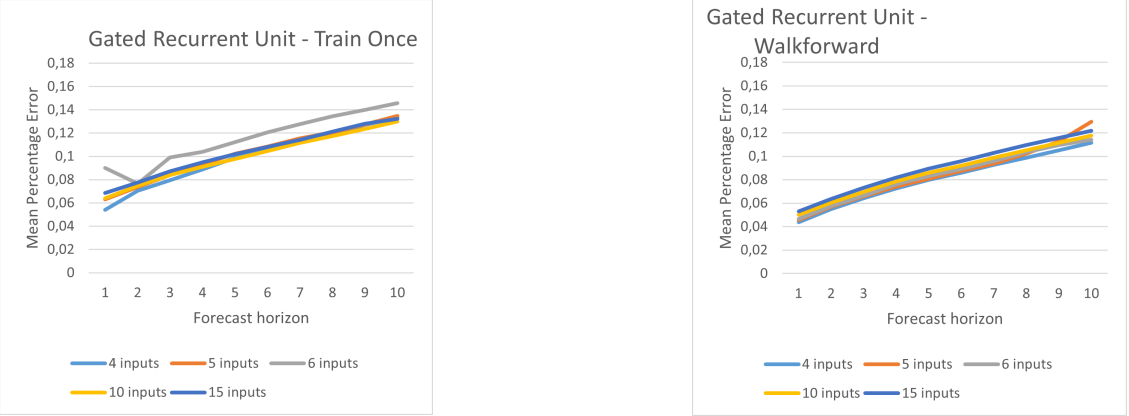

FIGURE 6.7: Unidirectional Gated Recurrent Unit Results

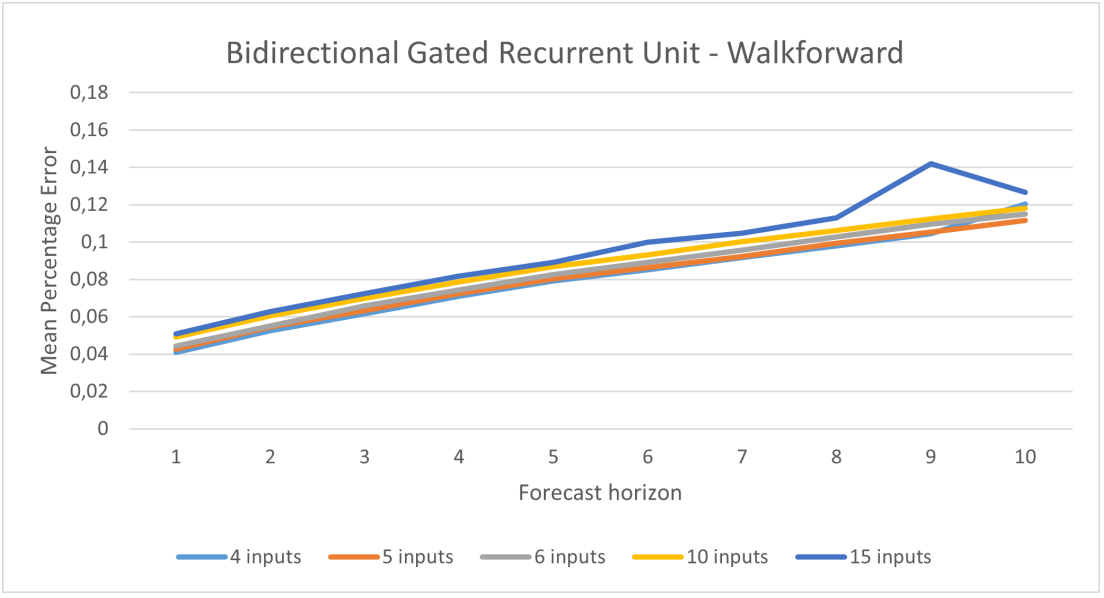

FIGURE 6.8: Bidirectional Gated Recurrent Unit - Walkforward

6.1.5 Model 5: CNN-LSTM

The CNN-LSTM Models underperformed when compared to all simpler recurrent neural network models. However, it was noted that the walk forward version of the CNN-LSTM model produced the smallest gradients over the time forecast horizon out of all models that were tested. This was a result of the model producing significantly poorer forecasts in the earlier forecast values than other models whilst the later forecast values were relatively more comparable to other models. Additionally, the CNN-LSTM train once model was the only model where the version with 10 inputs produced a higher Mean Absolute Percentage Error than the version with 15 inputs.

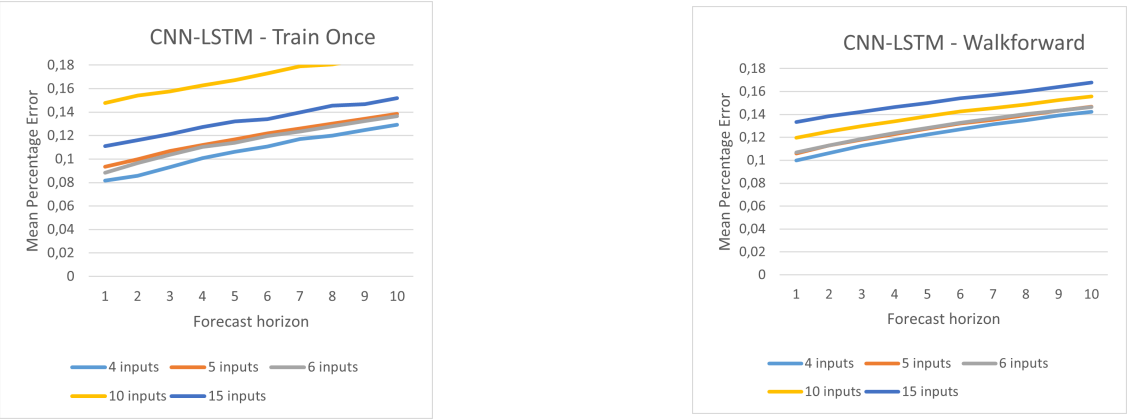

FIGURE 6.9: CNN-LSTM Results

6.1.6 Model 6: ConvLSTM

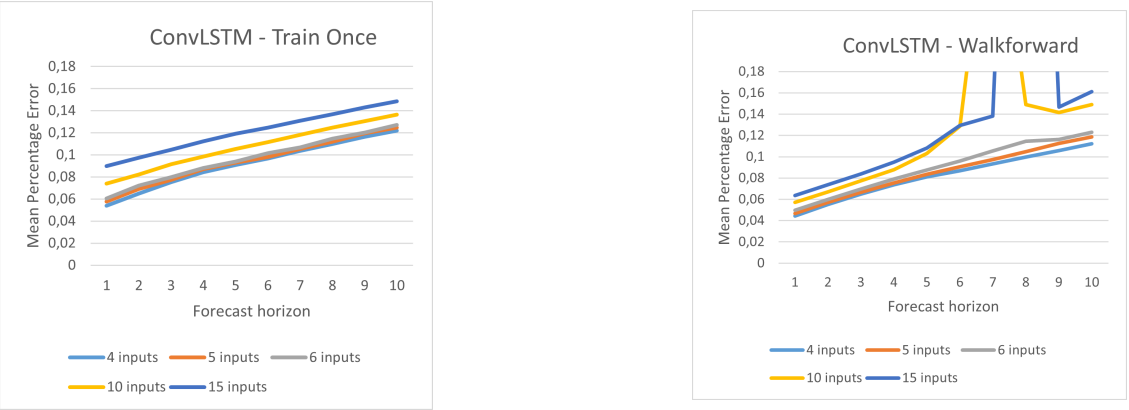

FIGURE 6.10: ConvLSTM Results

The ConvLSTM models underperformed against the simplified recurrent models across all forecast horizons and input sizes.

6.2 Multivariate Inputs

The multivariate tests utilised the Johannesburg Securities Exchanges' All Share Index as the second input stream of data into the model, along with the closing stock prices. Initially, these values were all inserted at their original index values and rand values, respectively. However, the resulting models provided forecasts that had extremely large Mean Absolute Percentage Errors. Therefore, all input values were normalised to be between 0 and 1. This has resulted in the Mean Absolute Percentage Errors decreasing significantly. However, the resulting Mean Absolute Percentage Errors still greatly

exceeded the Mean Absolute Percentage Errors observed when only the stock price was used as the input. Therefore, it can be concluded that taking the stock price on its own provides a greater degree of forecasting accuracy than adding the market index, in this case the JSE All Share Index, as well. This is in line with the research performed by Ullah et. al. [40] who noted that the models that provided the greatest degree of forecasting accuracy were the models that used only historical stock prices as their inputs. Refer to Table 6 and Table 7 of the Appendix where these results are contained.

6.3 Overall Performance

TABLE 6.2: Neural Network Models producing the lowest Mean Absolute Percentage Error

Time step forecasted	Lowest Mean Absolute Percentage Error	Model Type	Training Type	Number of Input Units
1	4.097%	LSTM Bidirectional	Walkforward	4
2	5.240%	GRU Bidirectional	Walkforward	4
3	6.173%	GRU Bidirectional	Walkforward	4
4	7.095%	GRU Bidirectional	Walkforward	4
5	7.914%	GRU Bidirectional	Walkforward	4
6	8.511%	GRU Bidirectional	Walkforward	4
7	9.177%	GRU Bidirectional	Walkforward	4
8	9.807%	GRU Bidirectional	Walkforward	4
9	10.160%	LSTM Bidirectional	Walkforward	4
10	10.899%	LSTM Bidirectional	Walkforward	4

The above table was compiled based on the overall performance of all models that were assessed. From the above table, it is evident that adding bidirectionality into Neural Network models improved accuracy among all time steps and that training models in a Walkforward fashion improved accuracy as well. Finally, it was observed that the fewer the number of input observations used, the more accurate the forecasts are likely to be, as well as that using only the stock price (and not coupled with a market index) provided greater accuracy.

6.4 Computational Performance

The purpose of this section is to assess the computational performance of each model, i.e., the time it took the full model to complete all forecasts in seconds. The experiments

performed were performed in a Google Collaboratory Python 3 notebook which utilises a Tensor Processor Unit v2 with 8 cores.

The computational performance of models that were only trained on the stock price are contained in Table 8 of the Appendix, whilst the computational forecasts of models that were trained on both the stock price and the market index are contained in Table 9 of the Appendix. As the Walkforward models were expected to have a much greater training time than the train once models, the "Change" column in these Appendix Tables were calculated by dividing the Walkforward time by the train once time, therefore producing a multiple of how many times longer it took the Walkforward model to complete its computations, when compared to the relevant train once model. As expected, as the number of input observations increased, the model took a longer time to complete its computations. The unidirectional Recurrent Neural network model performed the most efficiently with the Walkforward model using four inputs taking 9 934 seconds to complete its full set of computations whilst the version using fifteen historical observations as its inputs took 13 987 seconds. In contrast, the Bidirectional Gated Recurrent Unit models performed least efficiently with the Walkforward version with four historical observations as its input taking 27 419 seconds whilst the version with fifteen observations performed took 50 185 seconds. Interestingly, the Bidirectional Gated Recurrent Unit produced the lowest Mean Absolute Percentage Errors out of all models (Refer to 'Overall Performance' section above).

From the resulting performance noted, it was observed that the version of the models that used both the stock prices and the market index as its inputs only performed marginally poorer than the versions that used on the stock price. For the aforementioned unidirectional Recurrent Neural Network model, the Walkforward model with four inputs took 39 times longer than when it was only trained once whereas the Walkforward version with fifteen inputs took 51 times longer than its train once version. For the Bidirectional Gated Recurrent Unit model, the Walkforward model was 69 times slower than the train once model for the version with four input observations whilst it was 87 times slower for the version with fifteen observations as inputs.

The performance of the models using both the historical stock prices and the market index as its input data took a slightly longer time to complete its computations. The Unidirectional Recurrent Neural Network model took 10 207 seconds to complete all computations for four historical inputs and took 13 645 seconds for 15 historical inputs whilst the Bidirectional Gated Recurrent Unit took 26 413 seconds for four historical inputs and 44 903 seconds for fifteen historical inputs.

6.5 Discussion of Results

This section evaluates the overall results and findings of the experiments performed. It does this, firstly, by considering which machine learning model performed best, then comparing the results of the machine learning models to the results of the statistical models. Finally, it considers the overall results within the context of the Efficient Market Hypothesis and whether the results are indicative of at least weak form market efficiency.

The machine learning experiments performed found that bidirectional neural networks provided more accurate forecasts than neural networks that did not incorporate bidirectionality. By analysing the input data in both directions, the neural networks were able to identify nuances in the stock price movements that were not present in the analyses performed in one direction only. This is in line with the results of the experiments performed by Althelaya et al. [35] on the S&P500. The Bidirectional Long Short-Term Memory provided the greatest forecasting accuracy for the one step forecast. This is in line with the findings of Gao and Chai (2018) who found that the Long Short-Term Memory provided greater forecasting accuracy for next day stock prices, when compared to a variety of machine learning models, including a Support Vector Machine and a feed forward neural network.

Adding the applicable market index, the Johannesburg Securities Exchanges' All Share Index, to the neural network models greatly reduced the predictive value of the neural models. A possible reason for this is due to momentum effects on stock prices. Momentum effects refer to the tendency of poorly performing stocks and well-performing stocks in one period to continue that abnormal performance in following periods [7]. Therefore, should stock prices trend towards a certain direction, only historical prices of the same stock would provide insight as to what the stock price at the next time step would be. In such cases, adding the stock market index, which considers the movements in all stocks, to the model would only add noise to the model.

However, based on the traditional statistical experiments performed, it is evident that the traditional statistical models, in particular the theta model (See Table 3 of the Appendix) significantly outperform neural networks in forecasting next day closing prices of stocks listed on the Johannesburg Securities Exchange. This is in line with the results of the survey conducted by Makridakis and Spilliotis [3] which found that traditional statistical models outperformed machine learning models, and that the theta model, in particular, performed best.

The experiments performed on the neural networks which were run as Walkforward models greatly outperformed the models that were trained once with the 2019 calendar year's data being used as the input set. A possible explanation for this is the volatility that

was present in the financial markets during 2020 due to the coronavirus pandemic, which included one of the biggest market crashes and quickest market recoveries in history [54]. This market volatility was unforeseen by investors and thus stock price movements in 2019, the year upon which the machine learning models were trained, could not have factored these extreme fluctuations in. It was also found that the machine learning models with fewer inputs proved most accurate and that as inputs were increased, forecasting accuracy deteriorated. This could be attributed to the abovementioned market volatility which would have seen greater deviations in stock prices over longer historical input horizons, which were a result of factors external to the data. The deterioration in forecasting accuracy as more historical inputs are added contrasts with the statistical Theta method which saw its Mean Absolute Percentage Errors decline as more historical stock price observations were added to the model, albeit marginally. This is likely a result of the theta method providing greater weighting for the most recent forecasts.

As there were both machine learning and statistical methods that provided Mean Absolute Percentage Errors below 10%, it appears that future stock prices over multiple forecasting horizons can indeed be forecast using primarily historical stock prices as the inputs. Therefore, sufficient evidence has been obtained from the experiments performed to conclude that stocks listed on the Johannesburg Securities Exchange are not fully weak-form efficient. This is because prices could be forecasted within a reasonable error range using trends present in historical prices.

6.6 Summary

This chapter evaluated the forecasting accuracy computational performance of the neural network models built.

When evaluating the forecasting accuracy, it was found that fewer historical inputs and bidirectionality improves the forecasting accuracy of neural networks. Further, it was noted that the Gated Recurrent Unit provided the greatest degree of forecasting accuracy across multiple time steps for JSE-listed stocks. With regard to computational performance, the results produced were in line with expectations as more complex models which incorporated bidirectionality and/or additional inputs took longer to complete their computations.

Following this, the overall outcomes of the experiments were analysed. It was noted that the theta method outperformed all machine learning models utilised. Furthermore, the experimental outcomes of the research performed on the JSE were shown to be in line with the existing research evaluated as part of the literature review. Finally,

as the models built provided relatively accurate forecasts, it was concluded that the Johannesburg Securities Exchange was not fully weak-form efficient.

Chapter 7

Conclusion

This study evaluated the forecasting accuracy of multiple neural networks and multiple configurations of these neural networks in forecasting next day closing stock prices of the top forty stocks listed on the Johannesburg Securities Exchange. The neural networks were built utilising recurrent neural network, Long Short-Term Memory network and Gated Recurrent Unit architectures. The experiments were run with the neural networks on their own and incorporating bidirectionality. Additionally, the experiments were run as a train once implementation whereby the period 2 January 2019 to 31 December 2019 was used as the training set and 2 January 2020 to 29 May 2020 used as the testing set and then as a Walkforward implementation whereby all historical observations from 2 January 2019 up to the forecast date were used as the training set. A convolutional neural network and a random forest were used as machine learning baseline models.

The results of these neural network models were compared to the forecasting accuracy of classical statistical models forecasting the same stock prices. The classical statistical techniques utilised included a naïve method, Arima, exponential smoothing and the theta method. The experiments focussed on forecasting closing stock prices over a forecasting window of 10 observations and considered whether incorporating a secondary input source, being the market index, would improve model accuracy.

The results show that neural networks incorporating bidirectionality outperform neural networks that do not incorporate bidirectionality and that Walkforward forward models greatly outperform models that were trained once. Walkforward and bidirectional models were also shown to take a significantly larger number of computational time compared to models that were only trained once and models that did not incorporate bidirectionality, respectively.

In addition to the above, the study found that forecasts using only the historical stock price as the input outperformed forecasts, where both the stock price and market index were included. Additionally, it was found that for neural network models, fewer historical inputs provided greater forecasting accuracy.

In general, it was shown that the Bidirectional LSTM model produces the best forecasting accuracy one time step into the future. Furthermore, the Bidirectional Gated Recurrent Unit is more accurate two to eight time steps into the future with the Bidirectional LSTM model being more accurate for nine and ten time steps into the future. However, the classical statistical models significantly outperformed these models as well, with the statistical Theta Method performing best.

Overall, even though certain configurations of machine learning architectures provided relatively accurate forecasts, all of these were outperformed by statistical techniques that tend to place much greater emphasis on only the most recent historical observations and not an extended training window.

7.1 Contributions

The primary contribution of this work is an evaluation of the forecasting strength of statistical and machine learning models on stocks listed on the Johannesburg Securities Exchange. The Johannesburg Securities Exchange represents the largest and most developed stock exchange in Africa. However, research on the application of machine learning algorithms to stock price forecasting has not been extensively performed on the Johannesburg Securities Exchange [53]. This research also sought to provide insight into the degree of market efficiency, in terms of the Efficient Market Hypothesis, of the Johannesburg Securities Exchange.

7.2 Future Work

Further studies may consider whether including alternative secondary input data into the machine learning models may improve forecasting accuracy. These could consist of other numerical data like exchange rates or non-numerical data like performing sentiment analysis on news articles relevant to individual stocks.

Additionally, it may consider whether using even more historical observations could be utilised to forecast even further into the future in an accurate manner, and thus potentially improve profitability over a longer term.

Bibliography

[1] Charu C Aggarwal. Neural networks and deep learning. *Springer*, 10:978–3, 2018.

[2] Khursiah Zainal-Mokhtar and Junita Mohamad-Saleh. An oil fraction neural sensor developed using electrical capacitance tomography sensor data. *Sensors*, 13(9): 11385–11406, 2013.

[3] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PloS one*, 13(3):e0194889, 2018.

[4] Puneet Mathur. *Machine Learning Applications Using Python: Cases Studies from Healthcare, Retail, and Finance*. Apress, 2018.

[5] Sneha Soni. Applications of ANNs in stock market prediction: a survey. *International Journal of Computer Science & Engineering Technology*, 2(3):71–83, 2011.

[6] Jiayu Qiu, Bin Wang, and Changjun Zhou. Forecasting stock prices with long-short term memory neural network based on attention mechanism. *PloS one*, 15 (1):e0227222, 2020.

[7] Zvi Bodie, Alex Kane, and Alan Marcus. *Investments-Global edition*. McGraw Hill, 2014.

[8] Michael A Noakes and Kanshukan Rajaratnam. Testing market efficiency on the Johannesburg Stock Exchange using the overlapping serial test. *Annals of Operations Research*, 243(1):273–300, 2016.

[9] Eugene F Fama. Efficient capital markets a review of theory and empirical work. *The Fama Portfolio*, pages 76–121, 1970.

[10] Lumengo Bonga-Bonga. The evolving efficiency of the South African stock exchange. *International Business & Economics Research Journal (IBER)*, 11(9):997–1002, 2012.

[11] Carlos Correia, David Flynn, Enrico Uliana, and Michael Wormald. *Financial management. 8th Edition*. G. D'Ambrosio Angelillo, 2015.

[12] Magnus Hansson. On stock return prediction with LSTM networks. *Lund University Publications*, 2017.

[13] Ashu Prasad. Prediction and analysis of time series data using Tensorflow, Apr 2020. URL https://towardsdatascience.com/prediction-and-analysis-of-time-series-data-using-tensorflow-2136ef633018?gi=722f56752ee9.

[14] Jordy Bolton and Sven T von Boetticher. Momentum trading on the Johannesburg Stock Exchange after the global financial crisis. *Procedia Economics and Finance*, 24:83–92, 2015.

[15] Jason Brownlee. How to create an ARIMA model for time series forecasting in Python. *Machine Learning Mastery. Saatavissa: https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/. Hakupäivä*, 2:2019, 2017.

[16] Jason Brownlee. *Deep learning for time series forecasting: predict the future with MLPs, CNNs and LSTMs in Python*. Machine Learning Mastery, 2018.

[17] Grzegorz Dudek. Short-term load forecasting using Theta method. In *E3S Web of Conferences*, volume 84, page 01004. EDP Sciences, 2019.

[18] Vassilis Assimakopoulos and Konstantinos Nikolopoulos. The theta model: a decomposition approach to forecasting. *International journal of forecasting*, 16(4): 521–530, 2000.

[19] Jose A Fiorucci, Tiago R Pellegrini, Francisco Louzada, Fotios Petropoulos, and Anne B Koehler. Models for optimising the theta method and their relationship to state space models. *International Journal of Forecasting*, 32(4):1151–1161, 2016.

[20] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[21] Ciro Donalek. Supervised and unsupervised learning. In *Astronomy Colloquia. USA*, volume 27, 2011.

[22] Herve Abdi. A neural network primer. *Journal of Biological Systems*, 2(03):247–281, 1994.

[23] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11):e00938, 2018.

[24] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[25] Sagar Sharma. Epoch vs batch size vs iterations. *Towards Data Science*, 23, 2017.

[26] Jason Brownlee. Gentle introduction to the adam optimization algorithm for deep learning. *Machine Learning Mastery*, 3, 2017.

[27] Alexei Botchkarev. Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology. *arXiv preprint arXiv:1809.03006*, 2018.

[28] Juan Orozco Villalobos. Mean absolute error vs root-mean square error, Jan 2020. URL https://www.brainstobytes.com/mean-absolute-error-vs-root-mean-square-error/.

[29] Rob J Hyndman. Errors on percentage errors. *Hyndsight*, 2014.

[30] Jason Brownlee. *Long short-term memory networks with python: develop sequence prediction models with deep learning*. Machine Learning Mastery, 2017.

[31] Huy D Huynh, L Minh Dang, and Duc Duong. A new model for stock price movements prediction using deep neural network. In *Proceedings of the Eighth International Symposium on Information and Communication Technology*, pages 57–62, 2017.

[32] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[33] Khaled A Althelaya, El-Sayed M El-Alfy, and Salahadin Mohammed. Stock market forecast using multivariate analysis with bidirectional and stacked (LSTM, GRU). In *2018 21st Saudi Computer Society National Computer Conference (NCC)*, pages 1–7. IEEE, 2018.

[34] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[35] Khaled A Althelaya, El-Sayed M El-Alfy, and Salahadin Mohammed. Evaluation of bidirectional LSTM for short-and long-term stock market prediction. In *2018 9th international conference on information and communication systems (ICICS)*, pages 151–156. IEEE, 2018.

[36] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[37] Ali Agga, Ahmed Abbou, Moussa Labbadi, and Yassine El Houm. Short-term self consumption PV plant power production forecasts based on hybrid CNN-LSTM, ConvLSTM models. *Renewable Energy*, 177:101–112, 2021.

[38] Zexian Sun and Mingyu Zhao. Short-term wind power forecasting based on VMD decomposition, ConvLSTM networks and error analysis. *IEEE Access*, 8:134422–134434, 2020.

[39] Dattatray P Gandhmal and K Kumar. Systematic analysis and review of stock market prediction techniques. *Computer Science Review*, 34:100190, 2019.

[40] Shakir Ullah, Noman Javed, Ambreen Hanif, and Ali Abdullah. Stock price forecast using recurrent neural network. *Urdu News Headline, Text Classification by Using Different Machine Learning Algorithms*, page 47, 2019.

[41] Yauheniya Shynkevich, T Martin McGinnity, Sonya A Coleman, Ammar Belatreche, and Yuhua Li. Forecasting price movements using technical indicators: Investigating the impact of varying input window length. *Neurocomputing*, 264:71–88, 2017.

[42] A Jayanth Balaji, DS Harish Ram, and Binoy B Nair. Applicability of deep learning models for stock price forecasting an empirical study on BANKEX data. *Procedia computer science*, 143:947–953, 2018.

[43] AJP Samarawickrama and TGI Fernando. A recurrent neural network approach in predicting daily stock prices an application to the Sri Lankan stock market. In *2017 IEEE International Conference on Industrial and Information Systems (ICIIS)*, pages 1–6. IEEE, 2017.

[44] Ma Hiransha, E Ab Gopalakrishnan, Vijay Krishna Menon, and KP Soman. Nse stock market prediction using deep-learning models. *Procedia computer science*, 132:1351–1362, 2018.

[45] Sreelekshmy Selvin, R Vinayakumar, EA Gopalakrishnan, Vijay Krishna Menon, and KP Soman. Stock price prediction using LSTM, RNN and CNN-sliding window model. In *2017 international conference on advances in computing, communications and informatics (icacci)*, pages 1643–1647. IEEE, 2017.

[46] Yushi Guan, Peiyao Li, and Cheng Lu. Stock Price Prediction with CNN-LSTM Network. *University of Toronto Publications*, 2020.

[47] Irfan Ramzan Parray, Surinder Singh Khurana, Munish Kumar, and Ali A Altalbe. Time series data analysis of stock price movement using machine learning techniques. *Soft Computing*, 24(21):16509–16517, 2020.

[48] Tingwei Gao and Yueting Chai. Improving stock closing price prediction using recurrent neural network and technical indicators. *Neural computation*, 30(10): 2833–2854, 2018.

[49] Israt Jahan and Sayeed Sajal. Stock Price Prediction using Recurrent Neural Network (RNN) Algorithm on Time-Series Data. In *The Midwest Instruction and Computing Symposium (MICS 2018)*, pages 6–7, 2018.

[50] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. The performance of LSTM and BiLSTM in forecasting time series. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 3285–3292. IEEE, 2019.

[51] Bruno Miranda Henrique, Vinicius Amorim Sobreiro, and Herbert Kimura. Literature review: Machine learning techniques applied to financial market prediction. *Expert Systems with Applications*, 124:226–251, 2019.

[52] Stefan Drue. A comparison of a factor-based investment strategy and machine learning for predicting excess returns on the JSE. Master's thesis, University of Cape Town, 2018.

[53] Adam Balusik, Jared de Magalhaes, and Rendani Mbuvha. Forecasting The JSE Top 40 Using Long Short-Term Memory Networks. *arXiv preprint arXiv:2104.09855*, 2021.

[54] Ruan Jooste. Business maverick: Amazing: The JSE is now up for the year 2020, Aug 2020. URL https://www.dailymaverick.co.za/article/2020-08-12-amazing-the-jse-is-now-up-for-the-year-2020/.

Appendix A

Appendix

All results are presented in Mean Percentage Error, in decimal format.

A.1 Traditional Statistical Results

Company	Naïve method (5th day differencing)	Limited ARIMA Reults	Simple Exponential Smoothing (5 input observations)	Theta Method (5 input observations)
prosus	0.040392	0.016959	0.018388	0.017755
british american tobacco	0.023363	0.010725	0.011839	0.011273
naspers	0.037461	*	0.017948	0.016628
bhp	0.035486	0.016093	0.017827	0.015361
richemont	0.029761	0.013586	0.014932	0.013683
anglo american	0.040881	0.018373	0.020598	0.019045
amplats	0.046493	0.020449	0.022465	0.020969
vodacom	0.024216	0.011318	0.012278	0.012072
firstrand	0.031220	0.014866	0.016443	0.015244
gold fields	0.048941	0.022153	0.024145	0.023688
anglogold ashanti	0.044986	0.020491	0.022336	0.022521
standard bank	0.029213	0.014147	0.015627	0.014543
mondi plc	0.032197	0.014323	0.015766	0.013684
impala platinum	0.049226	0.028323	0.024654	0.023922
sibanye stillwater	0.069111	0.013697	0.030947	0.030333
sanlam	0.027665	0.021263	0.014949	0.014614
capitec	0.032954	0.017428	0.015938	0.014080
mtn	0.038005	0.014679	0.019139	0.016644
bid corporation	0.030595	0.014679	0.015811	0.015191
northam platinum	0.047587	*	0.021454	0.020412
sasol	0.039038	0.016971	0.018833	0.017540
discovery	0.029515	0.013228	0.014474	0.013978
shoprite	0.029665	0.013956	0.015231	0.015056
absa	0.030088	0.014254	0.015845	0.015244
reinet	0.035279	0.021398	0.022723	0.023536
clicks	0.028521	*	0.014578	0.014269
harmony	0.057047	0.024367	0.026979	0.027122
aspen	0.032555	0.014600	0.016178	0.015354
nedbank	0.031512	0.014511	0.015983	0.014965
bidvest	0.026921	0.012962	0.014288	0.013947
old mutual	0.030566	0.014260	0.015598	0.013449
remgro	0.023866	0.011824	0.012833	0.012630
exxaro	0.043167	0.018464	0.020440	0.019570
nepi rockcastle	0.035932	0.015749	0.016200	0.015967
multichoice	0.032420	0.015284	0.016495	0.015640
growth point	0.023319	0.010023	0.011047	0.011157
woolworths	0.031098	0.014388	0.015839	0.015768
spar	0.024223	0.011508	0.012763	0.012260
mr price	0.034664	0.015034	0.016687	0.017097
investec	0.032517	0.014947	0.016461	0.014121
AVERAGE	0.035292	0.015980	0.017574	0.016758

TABLE A.1: Traditional Statistical Results

*Data noted as not being stationary by the Arima model in Python's statsmodels library.

A.2 Naive Method Results

Naïve results	
Timesteps forecasted	Average Mean absolute Percentage Error
1	0.015818682
2	0.022622264
3	0.027595234
4	0.031741278
5	0.035291644
6	0.038513995
7	0.041482710
8	0.044282498
9	0.046890813
10	0.049316519
11	0.051566065
12	0.053670573
13	0.055671619
14	0.057651954

TABLE A.2: Naive Method Results

A.3 Theta Method Results

Theta method	Timesteps forecasted				
Number of input observations	1 timestep	2 timesteps	3 timesteps	4 timesteps	5 timesteps
5	0.01676	0.02394	0.02842	0.03220	0.03665
6	0.01677	0.02380	0.02828	0.03209	0.03657
7	0.01673	0.02362	0.02809	0.03194	0.03644
8	0.01664	0.02345	0.02794	0.03180	0.03629
9	0.01658	0.02335	0.02786	0.03174	0.03625
10	0.01650	0.02325	0.02776	0.03162	0.03614
11	0.01643	0.02314	0.02764	0.03152	0.03604
12	0.01634	0.02300	0.02754	0.03143	0.03595
13	0.01625	0.02291	0.02744	0.03135	0.03588
14	0.01620	0.02246	0.02736	0.03128	0.03581

TABLE A.3: Theta Method Results

A.4 Single train models using the share price only as its input

Model	Number of Inputs	1	2	3	4	5	6	7	8	9	10	Gradient
RNN Simple	4	0.048378	0.058133	0.068439	0.075846	0.083552	0.089701	0.095159	0.101565	0.107738	0.114103	0.007082
	5	0.051279	0.06066	0.072106	0.078086	0.085396	0.091871	0.098143	0.104527	0.110392	0.116672	0.007063
	6	0.053688	0.063358	0.072506	0.080205	0.086151	0.092005	0.09832	0.104459	0.110524	0.117346	0.006806
	10	0.057238	0.067363	0.07643	0.083999	0.091582	0.096343	0.101958	0.110179	0.116247	0.122661	0.00702
	15	0.059579	0.070559	0.079561	0.08798	0.094938	0.101458	0.107071	0.112128	0.119751	0.127036	0.00714
RNN Bidirectional	4	0.042279	0.055187	0.06509	0.073712	0.080648	0.087214	0.092667	0.098786	0.105009	0.111144	0.007275
	5	0.045719	0.057442	0.067345	0.075437	0.083516	0.089844	0.095408	0.101936	0.108077	0.114319	0.00734
	6	0.0486	0.060526	0.069085	0.076921	0.08372	0.090414	0.096254	0.10324	0.109096	0.11549	0.007136
	10	0.058406	0.070471	0.079106	0.086615	0.09339	0.09898	0.106498	0.11155	0.117214	0.12445	0.006964
	15	0.069993	0.076778	0.086738	0.095989	0.102503	0.106585	0.112057	0.117991	0.122098	0.129549	0.006435
LSTM Simple	4	0.146266	0.120023	0.094002	0.122458	0.124491	0.146473	0.147485	0.151946	0.171002	0.187071	0.006733
	5	0.134795	0.094917	0.105819	0.108124	0.133452	0.130298	0.141706	0.148856	0.166061	0.166412	0.006638
	6	1.641562	0.965415	0.11191	0.1177	0.141654	0.145913	0.146597	0.156546	0.177673	0.187077	-0.11085
	10	0.477774	1.111722	0.23358	0.31605	0.270704	0.25266	0.265472	0.295181	0.274242	0.310777	-0.0438
	15	0.231968	0.268823	0.181015	0.266157	0.208188	0.231799	0.189145	0.218191	0.20812	0.222092	-0.00324
LSTM Bidirectional	4	0.057558	0.068425	0.084136	0.098132	0.112249	0.126385	0.133099	0.147629	0.156546	0.163147	0.012143
	5	0.193827	0.084647	0.090466	0.114535	0.125578	0.140076	0.156093	0.159242	0.173291	0.18634	0.00628
	6	0.189942	0.27924	0.097852	0.114146	0.123015	0.141314	0.157766	0.162018	0.165073	0.168428	-0.00317
	10	0.344573	0.240124	1.562456	0.181135	0.172245	0.187754	0.193186	0.19853	0.42244	0.21652	-0.04027
	15	0.182112	0.174531	0.556088	0.22092	0.203285	0.177016	0.214504	0.225305	0.253014	0.195413	-0.00624
GRU Simple	4	0.05412	0.070423	0.079455	0.088905	0.098814	0.106297	0.113092	0.118713	0.126955	0.133074	0.00838
	5	0.06308	0.073643	0.0838	0.093264	0.102195	0.108371	0.115426	0.120454	0.127278	0.134615	0.007728
	6	0.090194	0.076665	0.09891	0.103882	0.112119	0.12054	0.127564	0.134305	0.14006	0.145863	0.00728
	10	0.064088	0.074577	0.083826	0.091153	0.097913	0.104609	0.111509	0.117554	0.123356	0.130084	0.007102
	15	0.068602	0.077545	0.087028	0.094721	0.101839	0.108167	0.114308	0.121254	0.128125	0.13232	0.007053
GRU Bidirectional	4	0.050436	0.069476	0.081407	0.092918	0.103038	0.109811	0.118526	0.125928	0.133076	0.139462	0.00941
	5	0.060662	0.070899	0.08271	0.095968	0.099333	0.111121	0.116454	0.12368	0.131465	0.138194	0.008484
	6	0.061554	0.070442	0.079556	0.09109	0.099619	0.105822	0.115793	0.122914	0.129909	0.135431	0.008353
	10	0.066357	0.075472	0.085364	0.093833	0.104229	0.107484	0.116038	0.119647	0.127165	0.130503	0.007154
	15	0.071645	0.080963	0.09001	0.101953	0.108488	0.112137	0.122271	0.124768	0.131812	0.13749	0.007194
CNN	4	0.047433	0.058327	0.068237	0.076348	0.083108	0.089089	0.095491	0.101404	0.107439	0.113434	0.007073
	5	0.058886	0.068688	0.076971	0.083872	0.089984	0.096076	0.101715	0.10775	0.113867	0.119561	0.00652
	6	0.057235	0.066343	0.074763	0.082071	0.088704	0.094577	0.101126	0.107235	0.11259	0.117916	0.006638
	10	0.068716	0.07701	0.084487	0.091107	0.098003	0.103566	0.110827	0.11677	0.123024	0.128112	0.006562
	15	0.087827	0.095965	0.10128	0.108651	0.113848	0.120596	0.125452	0.130536	0.136215	0.142654	0.005931
CNNLSTM	4	0.081496	0.085617	0.093121	0.100894	0.106112	0.110542	0.117097	0.120043	0.124835	0.129149	0.0054
	5	0.093298	0.099846	0.106716	0.111876	0.11676	0.121745	0.126074	0.130168	0.134158	0.138606	0.004926
	6	0.088337	0.096697	0.10366	0.110326	0.113891	0.119505	0.123413	0.127884	0.132525	0.136504	0.005153
	10	0.147767	0.154181	0.157623	0.162766	0.167249	0.173093	0.178935	0.180604	0.185319	0.190552	0.004681
	15	0.111106	0.116197	0.121128	0.127286	0.132018	0.134083	0.139794	0.145516	0.146648	0.151931	0.004498
ConvLSTM	4	0.053788	0.064914	0.075404	0.084322	0.090809	0.096535	0.103733	0.109917	0.116325	0.122013	0.007336
	5	0.05775	0.068939	0.077436	0.086886	0.092998	0.098188	0.105379	0.111851	0.119314	0.124768	0.007203
	6	0.060463	0.072062	0.079746	0.088022	0.094007	0.101434	0.106663	0.114821	0.120013	0.126976	0.007109
	10	0.07382	0.082162	0.091558	0.098568	0.105498	0.111492	0.118064	0.124691	0.130597	0.136562	0.006872
	15	0.089654	0.097497	0.104636	0.112167	0.119304	0.124574	0.130909	0.136721	0.143034	0.148592	0.006492

TABLE A.4: Single train models using the share price only as its input

A.5 Walkforward models using the share price only as its input

Model	Number of Inputs	1	2	3	4	5	6	7	8	9	10	Gradient
Simple Perceptron	4	0.042928518	0.054570903	0.064878894	0.073059146	0.080646997	0.087088914	0.093529005	0.099572047	0.105880562	0.112005299	0.007407
	5	0.046168585	0.057220369	0.066826255	0.075560432	0.083219953	0.089216557	0.095845751	0.101953889	0.108733909	0.114145453	0.007363
	6	0.04880726	0.05972727	0.069205237	0.077690492	0.08523215	0.091080041	0.097279893	0.103681655	0.110590708	0.116537011	0.007289
	10	0.058097327	0.068690257	0.07703105	0.086326796	0.094666478	0.102210024	0.11626708	0.125661156	0.128009263	0.126616462	0.008318
	15	0.067768861	0.078369695	0.088745854	0.100403099	0.115360863	0.223150989	0.126765391	0.14615074	0.164439756	0.15659979	0.011369
Convolutional Neural Network	4	0.046361558	0.05745881	0.06707708	0.075582358	0.082363067	0.088513407	0.094738986	0.100818156	0.106811903	0.112735366	0.007122
	5	0.057606289	0.067200054	0.07579998	0.0829015	0.08930423	0.095343025	0.101282652	0.107181734	0.113265197	0.119041966	0.006627
	6	0.053398766	0.063115283	0.072298868	0.080459026	0.086663874	0.093309428	0.099540894	0.105586603	0.111734615	0.117300325	0.006944
	10	0.063324469	0.072216381	0.081419018	0.089011781	0.09589971	0.101469066	0.107866665	0.11405238	0.119703607	0.125592009	0.006776
	15	0.081556767	0.090473544	0.098561544	0.105990129	0.111427949	0.11947139	0.125808809	0.13072992	0.136978766	0.142239127	0.006667
Simple Recurrent Neural Network	4	0.046033697	0.056807653	0.066872989	0.07495156	0.082432673	0.088636587	0.094901616	0.101101101	0.107368397	0.113542935	0.007265
	5	0.048505093	0.059366395	0.068254058	0.076865105	0.083788382	0.090187954	0.096399048	0.103162629	0.109490797	0.115397889	0.007227
	6	0.04888691	0.059934369	0.069631069	0.078047347	0.085786848	0.091694388	0.098514743	0.105044341	0.112836553	0.118172959	0.007505
	10	0.051974887	0.063380729	0.073671262	0.082535236	0.090662252	0.097156803	0.12777459	0.110779905	0.117805733	0.124389515	0.008245
	15	0.053906316	0.065194131	0.075310955	0.085107432	0.093646483	0.099111039	0.106328698	0.113737123	0.120512535	0.126955477	0.007917
Bidirectional Recurrent Neural Network	4	0.040984506	0.053164497	0.063305044	0.072695135	0.0807945	0.086864636	0.093229623	0.099710323	0.106082671	0.111959555	0.00763
	5	0.044015842	0.05536873	0.065284161	0.074387053	0.082190677	0.088418885	0.095320586	0.101565257	0.108145644	0.114386593	0.007595
	6	0.045044224	0.056726457	0.067121714	0.076147758	0.084230557	0.089963166	0.098043425	0.103848294	0.109958539	0.116318352	0.007692
	10	0.052301619	0.063116933	0.073615099	0.082627718	0.090091109	0.098270993	0.103943168	0.111573883	0.117557463	0.123308014	0.00777
	15	0.058146316	0.068807057	0.07849668	0.086601126	0.094476203	0.101150416	0.107337669	0.129350063	0.161389806	0.157425582	0.011301
LSTM Simple	4	0.046143133	0.056894785	0.06565319	0.073096779	0.080931904	0.087053988	0.093355929	0.099130514	0.108731205	0.112808322	0.007255
	5	0.049656222	0.060491309	0.069274247	0.077736322	0.084780661	0.091322211	0.099399344	0.102932911	0.111560942	0.114194477	0.00714
	6	0.053132491	0.063055045	0.072592009	0.084031991	0.087285873	0.094675746	0.099361291	0.105442506	0.109994251	0.116079265	0.006744
	10	0.059650463	0.06902111	0.077260042	0.086186338	0.093430702	0.102545653	0.105327078	0.127709839	0.118192143	0.121337294	0.007383
	15	0.069798255	0.079222076	0.086459425	0.094009869	0.10070755	0.104307569	0.111022124	0.115532122	0.121122685	0.126828895	0.0061
LSTM Bidirectional	4	0.040965142	0.053428112	0.062512177	0.074257087	0.088712069	0.085308869	0.092338354	0.10362725	0.101603946	0.108991536	0.007308
	5	0.04260744	0.054253283	0.064339505	0.07316771	0.079822907	0.092959879	0.092860282	0.103307906	0.133908278	0.113365067	0.008857
	6	0.046409131	0.058763057	0.068331948	0.076499592	0.086549705	0.208844121	0.10009303	0.122647679	0.109156736	0.112560153	0.008562
	10	0.050249792	0.061661672	0.069937223	0.080920171	0.103540667	0.091987547	0.097905473	0.105378146	0.112713004	0.116210482	0.007076
	15	0.061656649	0.071389384	0.084239936	0.086355716	0.09379465	0.09929189	0.10578897	0.117884566	0.128210764	0.123900559	0.007212
GRU Simple	4	0.043805812	0.05479637	0.06429022	0.07267918	0.079964433	0.085832049	0.092757435	0.098789226	0.105131673	0.11157889	0.007278
	5	0.045706155	0.056978412	0.065735402	0.074223403	0.080836419	0.087602071	0.093776061	0.10229102	0.11265703	0.129410362	0.008432
	6	0.046574179	0.057486382	0.066592171	0.076166165	0.083218447	0.089173226	0.095771646	0.103380086	0.109516022	0.114205168	0.007404
	10	0.050031437	0.060269055	0.069598213	0.07857464	0.085933929	0.09207718	0.098754236	0.105179474	0.111377604	0.117638001	0.007338
	15	0.053198508	0.06358516	0.073107895	0.081815628	0.089334369	0.095764487	0.102826086	0.109463775	0.115778368	0.121769476	0.007477
GRU Bidirectional	4	0.040970621	0.052401606	0.061731293	0.070945447	0.079143951	0.085108731	0.091768611	0.098073326	0.104432378	0.12033579	0.008052
	5	0.042538826	0.054396278	0.063397719	0.072293868	0.080466851	0.086204562	0.092335414	0.099455727	0.105279247	0.111503022	0.007412
	6	0.044419838	0.055162218	0.065996656	0.074484454	0.082690762	0.089242801	0.095738705	0.102909085	0.109490944	0.114932223	0.007696
	10	0.04914108	0.060347707	0.069819626	0.078678539	0.086979	0.093149129	0.10031655	0.106307324	0.112548194	0.118220283	0.007519
	15	0.050787423	0.062657169	0.072441229	0.081902826	0.089261812	0.099997491	0.104845558	0.112910682	0.142085108	0.126622043	0.009215
CNNLSTM	4	0.099765033	0.106091671	0.11252466	0.117827202	0.122591089	0.127053786	0.131319969	0.135009839	0.138994129	0.142209037	0.004665
	5	0.106044348	0.113066169	0.11808925	0.122771859	0.127519877	0.132071988	0.135264138	0.139412114	0.14317467	0.146820121	0.004402
	6	0.106903652	0.112852903	0.118780329	0.123624549	0.128127248	0.132835822	0.136576498	0.140358792	0.143416865	0.146604864	0.00438
	10	0.119492608	0.125033583	0.129855168	0.134133404	0.138550992	0.142553106	0.145497329	0.148802789	0.152650985	0.155837782	0.003959
	15	0.133341731	0.138604642	0.142311865	0.146467767	0.149957708	0.154012476	0.156992135	0.160107154	0.164114108	0.167989133	0.003727
ConvLSTM	4	0.044173703	0.055026353	0.06484055	0.073667119	0.081091166	0.087000984	0.093232107	0.099540381	0.105781054	0.112046126	0.007298
	5	0.046669374	0.057130239	0.066492421	0.075282809	0.083294719	0.090481069	0.09724656	0.104815107	0.112623397	0.118527865	0.007878
	6	0.04976296	0.059728774	0.069605646	0.079203053	0.087452186	0.096058369	0.105490855	0.114572869	0.116322785	0.12280955	0.008278
	10	0.057156224	0.067079216	0.077460655	0.087769773	0.102995586	0.128640299	0.377663136	0.149010355	0.141517492	0.148884207	0.015756
	15	0.06368755	0.073814645	0.083856564	0.094794058	0.108179965	0.129426888	0.138152639	0.84337832	0.146440403	0.161275807	0.032337

TABLE A.5: Walkforward models using the share price only as its input

A.6 Single train models using the share price and the JSE All Share Index as its inputs

Model	Number of Inputs	1	2	3	4	5	6	7	8	9	10	Gradient
RNN Simple	4	0.281657	0.284079	0.286089	0.287887	0.289898	0.291757	0.293489	0.295019	0.296505	0.298326	0.00182
	5	0.281848	0.283616	0.285743	0.287695	0.289767	0.291741	0.293309	0.295067	0.296709	0.298353	0.001852
	6	0.284917	0.287346	0.28942	0.291408	0.293079	0.29517	0.296663	0.298289	0.299932	0.301804	0.001832
	10	0.286702	0.288791	0.290744	0.292837	0.295059	0.297235	0.299078	0.300813	0.30325	0.305004	0.002043
	15	0.293885	0.296388	0.299156	0.301289	0.303668	0.306018	0.308092	0.309673	0.311688	0.313738	0.002189
RNN Bidirectional	4	0.284476	0.286769	0.28887	0.290648	0.292579	0.294355	0.296013	0.297379	0.298967	0.30063	0.001765
	5	0.281343	0.28362	0.285969	0.287815	0.289796	0.291817	0.293462	0.29506	0.296744	0.298486	0.001882
	6	0.282711	0.285082	0.28724	0.289267	0.291167	0.292792	0.29455	0.295899	0.297607	0.299564	0.001819
	10	0.286079	0.288351	0.290692	0.292469	0.294769	0.29714	0.299045	0.300739	0.302481	0.304455	0.00204
	15	0.292858	0.295262	0.297653	0.300326	0.302784	0.304827	0.30701	0.308564	0.310435	0.31244	0.002176
LSTM Simple	4	0.282879	0.284933	0.286642	0.288609	0.290683	0.29289	0.294331	0.296081	0.297767	0.299483	0.001854
	5	0.281751	0.283756	0.285979	0.287958	0.28982	0.291894	0.29369	0.295355	0.297077	0.298806	0.001896
	6	0.283396	0.285949	0.288368	0.290053	0.292183	0.294464	0.296097	0.297696	0.29962	0.301622	0.001981
	10	0.289677	0.292259	0.294577	0.296656	0.298745	0.301234	0.303228	0.305396	0.307464	0.309233	0.002174
	15	0.296623	0.299346	0.301842	0.304245	0.306915	0.308897	0.311102	0.313023	0.315051	0.317366	0.002273
LSTM Bidirectional	4	0.280902	0.282921	0.284914	0.2868	0.288712	0.291021	0.292585	0.293993	0.295666	0.297425	0.001836
	5	0.283327	0.285358	0.287381	0.28932	0.291187	0.293211	0.294701	0.296204	0.297848	0.29954	0.001792
	6	0.278965	0.281096	0.283568	0.285663	0.287398	0.289479	0.291329	0.292868	0.29435	0.296173	0.001898
	10	0.289808	0.292034	0.294656	0.296998	0.299008	0.301354	0.303262	0.305109	0.307347	0.309438	0.002165
	15	0.302661	0.30544	0.30837	0.310261	0.312855	0.315638	0.317164	0.319508	0.321707	0.323614	0.002313
GRU Simple	4	0.28052	0.282773	0.284896	0.286738	0.28865	0.290686	0.292478	0.293861	0.29557	0.297066	0.001834
	5	0.282709	0.284881	0.287021	0.28888	0.291139	0.293063	0.294936	0.296408	0.297874	0.29966	0.001882
	6	0.28509	0.287107	0.289351	0.291473	0.293652	0.295493	0.29693	0.298771	0.300652	0.302476	0.001919
	10	0.293154	0.295159	0.297362	0.299308	0.301617	0.303752	0.305785	0.307625	0.309607	0.311368	0.002048
	15	0.287478	0.289894	0.292495	0.294865	0.297196	0.299757	0.30155	0.303429	0.305464	0.307359	0.002213
GRU Bidirectional	4	0.28217	0.284399	0.286649	0.288421	0.290326	0.292273	0.293779	0.295431	0.296923	0.298617	0.001804
	5	0.284113	0.286106	0.288263	0.290031	0.29209	0.29411	0.295873	0.297541	0.299414	0.301309	0.001902
	6	0.283269	0.285734	0.288061	0.289941	0.292066	0.294186	0.295923	0.297804	0.299369	0.301143	0.00197
	10	0.299167	0.301136	0.303558	0.305739	0.308127	0.310121	0.312194	0.314101	0.316093	0.31791	0.002106
	15	0.290023	0.292696	0.29512	0.297599	0.299845	0.3023	0.30428	0.306096	0.308021	0.31009	0.002214
CNNLSTM	4	0.291761	0.293964	0.298576	0.298317	0.30166	0.301283	0.304365	0.305433	0.307336	0.309658	0.001859
	5	0.286342	0.286126	0.291075	0.292381	0.294618	0.296882	0.299038	0.299654	0.298807	0.304612	0.001929
	6	0.29091	0.293142	0.292834	0.296443	0.297998	0.297188	0.299736	0.301592	0.303791	0.3046	0.001519
	10	0.287915	0.290231	0.291765	0.294206	0.297887	0.299242	0.300726	0.302002	0.304139	0.308007	0.002123
	15	0.294375	0.295633	0.298961	0.301519	0.305448	0.306477	0.308281	0.309779	0.311974	0.314298	0.002237
ConvLSTM	4	0.331334	0.333611	0.335053	0.33677	0.338445	0.340442	0.341957	0.343293	0.344585	0.346409	0.001644
	5	0.326919	0.328942	0.330988	0.332809	0.334681	0.336892	0.338202	0.339838	0.341312	0.343208	0.001793
	6	0.318679	0.320729	0.323044	0.324818	0.326445	0.328321	0.330315	0.331619	0.333278	0.33536	0.001813
	10	0.308153	0.310347	0.312546	0.314516	0.316795	0.318996	0.320945	0.322716	0.324369	0.326323	0.002024
	15	0.308055	0.310763	0.313091	0.315458	0.317899	0.320172	0.322115	0.323857	0.325862	0.327867	0.002182

TABLE A.6: Single train models using the share price and the JSE All Share Index as its inputs

A.7 Total time taken to complete all computations for univariate models

Model	Number of Inputs	Train Once	Walkforward	Change	Train Once Rank	Walkforward Rank	Change In Rank
RNN Simple	4	254.16	9 934.32	39.09	1	1	0
	5	257.97	10 642.49	41.26	2	2	0
	6	262.98	11 570.29	44.00	3	3	0
	10	271.75	12 749.28	46.91	4	4	0
	15	274.60	13 986.68	50.93	5	7	2
RNN Bidirectional	4	298.28	13 534.70	45.38	6	5	-1
	5	308.35	13 768.84	44.65	8	6	-2
	6	314.00	15 035.11	47.88	10	8	-2
	10	334.39	16 860.48	50.42	14	9	-5
	15	353.60	18 042.83	51.03	16	12	-4
LSTM Simple	4	314.11	17 070.87	54.35	11	10	-1
	5	324.26	18 633.90	57.47	13	13	0
	6	336.99	20 083.52	59.60	15	15	0
	10	411.96	24 469.55	59.40	30	25	-5
	15	435.49	27 977.83	64.25	35	35	0
LSTM Bidirectional	4	397.12	22 114.27	55.69	24	17	-7
	5	410.41	24 045.51	58.59	29	23	-6
	6	430.58	25 784.44	59.88	34	26	-8
	10	512.14	29 806.92	58.20	37	37	0
	15	588.15	33 773.82	57.42	40	38	-2
GRU Simple	4	300.37	17 839.67	59.39	7	11	4
	5	313.85	18 660.07	59.46	9	14	5
	6	323.90	20 211.34	62.40	12	16	4
	10	370.20	23 404.39	63.22	19	22	3
	15	408.56	25 956.12	63.53	27	27	0
GRU Bidirectional	4	396.70	27 418.60	69.12	23	34	11
	5	413.20	27 138.66	65.68	31	32	1
	6	444.90	29 288.57	65.83	36	36	0
	10	512.16	38 179.10	74.54	38	39	1
	15	580.21	50 185.41	86.50	39	40	1
CNN-LSTM	4	407.91	22 522.45	55.21	26	19	-7
	5	403.94	22 320.78	55.26	25	18	-7
	6	408.57	23 219.64	56.83	28	21	-7
	10	422.11	22 556.22	53.44	32	20	-12
	15	426.47	24 260.97	56.89	33	24	-9
ConvLSTM	4	369.69	26 223.62	70.93	18	28	10
	5	373.45	26 678.42	71.44	20	30	10
	6	374.84	27 231.70	72.65	21	33	12
	10	378.24	26 771.20	70.78	22	31	9
	15	367.25	26 511.42	72.19	17	29	12

TABLE A.7: Total time taken to complete all computations for univariate models (Time noted in seconds)

A.8 Total time taken to complete all computations for multivariate models

Model	Number of Inputs	Train Once	Walkforward	Change	Train Once Rank	Walkforward Rank	Change In Rank
RNN Simple	4	298.99	10 206.66	34.14	1	1	0
	5	301.93	10 885.47	36.05	2	2	0
	6	315.58	11 434.09	36.23	3	3	0
	10	366.72	12 651.17	34.50	5	4	-1
	15	346.65	13 644.68	39.36	4	6	2
RNN Bidirectional	4	369.70	13 492.05	36.49	6	5	-1
	5	384.86	14 615.29	37.98	9	7	-2
	6	386.75	15 291.92	39.54	10	8	-2
	10	433.32	17 813.77	41.11	21	12	-9
	15	449.02	18 257.69	40.66	22	13	-9
LSTM Simple	4	378.54	16 717.10	44.16	7	9	2
	5	393.31	21 511.78	54.69	12	19	7
	6	418.55	17 604.64	42.06	19	11	-8
	10	485.92	25 173.87	51.81	29	24	-5
	15	558.56	30 034.36	53.77	37	36	-1
LSTM Bidirectional	4	489.76	25 392.87	51.85	30	25	-5
	5	524.96	30 889.28	58.84	35	37	2
	6	534.78	24 845.43	46.46	36	22	-14
	10	647.68	29 976.59	46.28	39	35	-4
	15	746.73	33 321.27	44.62	40	38	-2
GRU Simple	4	381.33	17 284.28	45.33	8	10	2
	5	388.48	19 009.45	48.93	11	14	3
	6	394.57	20 611.23	52.24	13	16	3
	10	409.85	25 086.69	61.21	16	23	7
	15	505.52	27 447.90	54.30	32	33	1
GRU Bidirectional	4	451.77	26 412.86	58.47	23	29	6
	5	453.07	29 508.72	65.13	24	34	10
	6	469.20	26 174.57	55.78	25	28	3
	10	511.12	35 830.39	70.10	33	39	6
	15	578.73	44 903.14	77.59	38	40	2
CNN-LSTM	4	400.50	21 377.78	53.38	15	18	3
	5	400.43	20 920.27	52.24	14	17	3
	6	412.96	20 437.49	49.49	17	15	-2
	10	415.28	23 371.04	56.28	18	20	2
	15	420.85	24 247.17	57.61	20	21	1
ConvLSTM	4	481.90	26 843.34	55.70	28	32	4
	5	477.64	26 573.46	55.63	27	30	3
	6	475.25	25 979.85	54.67	26	26	0
	10	517.66	26 779.91	51.73	34	31	-3
	15	495.47	26 077.72	52.63	31	27	-4

TABLE A.8: Total time taken to complete all computations for multivariate models (Time noted in seconds)