MSc Research Report

Modelling implied volatility in South African stock options

A comparison of statistical and machine learning methods

By

Marike Harmse

Student Number: 1423066

Supervisor

Dr DM Rose

A research report submitted to the Faculty of Science, University of the Witwatersrand, in partial fulfilment of the requirements for the degree of Master of Science

June 20, 2023

Declaration

I declare that this Research Report is my own, unaided work. It is being submitted for the Degree of Master of Science at the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination at any other University.

(Signature of candidate)
Date: 20 June 2023

Abstract

The Black-Scholes model is used to derive the price of an option. Two of the underlying assumptions is that of constant volatility and normality in stock price returns. Volatility is often modelled using various statistical and machine learning methods. This research focused on the use of GARCH, EGARCH, APARCH and LSTM models to predict the volatility underlying the All Share Index, a South African stock index. Exploratory data analysis indicated that the ALSI exhibited the leverage effect, long memory properties and asymmetry in its returns and that the traditional models such as ARCH and GARCH may not be sufficient to model stock price volatility. These models will therefore be used as benchmark models against the APARCH, EGARCH and LSTM models. The ARMA(3,3)-EGARCH(1,1) model outperformed the other models considered. While LSTM models can add value in the estimation of stock price volatility, they are highly sensitive to the selected hyperparameters and architecture used.

Keywords: ALSI, ARIMA, APARCH, ARCH, GARCH, EGARCH, LSTM Neural Networks, Statistical Learning, Deep Learning

Dedication

This research is dedicated to the three people whose contributions in my life shaped me to be who I am. My mother, Sarie Harmse, made an incredible number of sacrifices to provide me with the best education and future possible. My grandmother, Sarie Smith, always brought coffee, words of encouragement and more snacks than I could eat. My late grandfather, Danie Smith, filled the role of the father I never had. Thank you for everything.

Acknowledgements

A special thanks goes to my supervisor, Dr David Rose, whose expertise and input was invaluable to me. Your insightful feedback pushed me not only to bring my work to a higher level, but to also enjoy the research component and learning associated with it.

In addition, thank you to the following lecturers at the University of the Witwatersrand: Dr Chimedza, Dr Hove, Dr Chipoyera, Dr Nasejje and Dr Docrat. While their contribution was towards the coursework component of my degree, I gained considerable knowledge and insights that contributed not only towards my research, but also to my growth in the field.

Special thanks go to the following colleagues at True North Partners: Raymond Sinclair, Jaco Booysen, Annemie Badenhorst, Bayanda Dumezweni, Julia Krähe, and Ornella De Rose.

Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

ACF Autocorrelation Function

ADF Augmented Dickey-Fuller

AIC Akaike Information Criterion

ALSI All Share Index

ANN Artificial Neural Network

APARCH Asymmetric Power Auto Regressive Conditional Heteroskedasticity

AR Auto Regressive

ARCH Auto Regressive Conditional Heteroskedasticity

ARFIMA Autoregressive Fractionally Integrated Moving Average

ARMA Auto Regressive Moving Average

CEC Constant Error Carousel

EGARCH Exponential Generalised Auto Regressive Conditional Heteroskedasticity

ENN Ensemble Neural Network

FNN Feedforward Neural Network

GARCH Generalised Auto Regressive Conditional Heteroskedasticity

IID Independent and Identically Distributed

JB Jarque-Bera

JSE Johannesburg Stock Exchange

LB Ljung-Box

LM Lagrange-Multiplier

LSTM Long Short Term Memory

MA Moving Average

MLP Multi Layer Perceptron

PACF Partial Autocorrelation Function

QQ Quantile-Quantile

RMSE Root Mean Squared Error

S&P Standard and Poor's

SGD Stochastic Gradient Descent

STL Seasonal and Trend Decomposition Using Loess

TGARCH Threshold General Auto Regressive Conditional Heteroskedasticity

WNP White Noise Process

Glossary of Terms

Derivative Products - A financial product whose value is determined by the value of other products such as stocks, currencies and commodities.

Discounting - Discounting is used to determine the current value of payments that will be received in the future.

Efficient Markets - Markets are defined as efficient when prices reflect available information in a timely manner.

Hedging - A transaction using financial instruments that aims at offsetting another transaction in order to minimise losses from potential adverse movements.

Option - An option is a subclass of derivative products where the trader can trade a specified asset at a pre-agreed price on a stipulated date, but there is no obligation to do so.

Over-The-Counter - Over-the-counter products are traded between two parties instead of on an exchange and have features specific to the need of the traders.

Shock - A shock to the value of an asset is an unexpected, large movement in the asset price.

Speculation - The investment in an asset with the purpose of realising a financial gain.

Volatility - Volatility measures the fluctuation of an asset's price from its mean.

Chapter 1

Introduction

1.1 Introduction

The Black-Scholes model is used to derive the price of an option. Since inception, derivative products have become increasingly popular, both from a hedging as well as a speculative perspective and are a valuable component of market risk management. It is important to price the asset correctly, which is often complicated by the variety and complexity of these products. The importance increases when the product is over-the-counter and is designed explicitly to cater for the needs of the trader. European options are a popular subclass of derivative products, structured in such a way that the underlying asset can only be traded on a specified date, but there is no obligation to do so (Hull, 2003).

An important input in the Black-Scholes model is volatility. In practice, volatility is estimated since it is not possible to observe it directly. There are various methods used for volatility estimation. Of particular interest are statistical learning techniques; as a result, these methods' popularity has increased over time. Time series models are popular statistical learning methods used to estimate volatility. These include the Autoregressive Conditional Heteroskedasticity (ARCH), Generalised ARCH (GARCH), Exponential GARCH (EGARCH) and Asymmetric Power ARCH (APARCH) models.

Recently the focus has shifted towards the use of machine learning methods. Artificial Neural Networks (ANNs), especially methods with long memory components such as the Autoregressive Neural Network with External Input and Long Short Term Memory (LSTM), have proved useful in volatility estimation.

This research report aims to compare the results of statistical and machine learning volatility estimates. The daily returns of the All Share Index (ALSI), a South African stock index, will be used. In addition, the accuracy of the models will be analysed by comparing the estimates obtained.

1.2 Background

While the Black-Scholes model is used in practice, it has shortcomings due to its underlying assumptions (Hull, 2003):

1. The only asset priced by the model is a European option;
2. No dividends are payable;
3. It is impossible to predict market movements;
4. Zero transaction costs are incurred when purchasing the option;
5. Asset returns follow a normal distribution; and
6. The discounting interest rate and volatility are known and constant.

1.3 Statement of the Problem

When there are no fluctuations in stock prices over time, asset returns and volatility are constant. However, changes in the bid and offer of the asset result in changes in volatility and the assumption of constant volatility does not hold. Volatility may be estimated using one of two methods. The first is known as historical volatility, where historical asset price returns are used. The second is implied volatility and is based on option contract prices. The latter method is preferred due to the use of current rather than historical data. As a result, unrelated historical events do not skew volatility estimates. If an event such as the COVID-19 crisis occurs, the volume of traded stocks and their prices change, resulting in large increases in volatility. When historical data are used to estimate volatility, these movements may not be reflected in a timely manner. Implied volatility estimates use the view of the market on price movements and therefore provides improved accuracy (Hull, 2003).

The Black-Scholes model assumes that asset returns are distributed normally. In practice, asset returns are often skewed with heavier tails than those for which the normal distribution accounts. The research proposes methods that capture the true nature of asset returns.

1.4 Aims and Objectives

1.4.1 Aims

This research aims to apply methods that address the constant volatility and normally distributed asset return assumptions. Statistical and machine learning methods can be used to model implied volatility. These methods are compared to address the identified shortcomings and to determine which method has the highest predictive power. The implied volatility underlying the ALSI will be applied to fit the models.

1.4.2 Objectives

The objectives of this research report are as follows:

- To establish whether the assumptions of the Black-Scholes model, especially the normality of asset returns and constant volatility assumptions, hold for the ALSI return series;
- To determine which of the ARCH/GARCH, APARCH and EGARCH models has the highest prediction accuracy;
- To compare statistical against machine learning methods by comparing the GARCH-type model with the highest prediction accuracy to an LSTM neural network; and
- To investigate the effect of changing specific hyperparameters on the prediction accuracy of the LSTM model in order to obtain the model with the highest prediction accuracy

Chapter 2

Literature and Technical Review

2.1 Introduction

This chapter provides both a literature and technical review, which are discussed in Sections 2.2 and 2.3 respectively. The literature review focuses on findings that prove that the constant volatility and normality assumptions made by the Black-Scholes model are unrealistic. It also compares how each of the proposed models addresses the potential shortcomings of previous models. The technical review provides a mathematical overview of the models that will be used to model implied volatility.

2.2 Literature Review

2.2.1 Black-Scholes and Market Theory

An important component of any investment decision is the risk/return trade-off. When additional risk is taken on an investment, an investor expects to be compensated with a higher return on the investment. Risk is measured through the volatility of the stock's return. Volatility in stock prices is used in financial models to estimate the correct return/price for a given level of risk. The accuracy of volatility estimates is therefore key in determining the required return on an asset.

Black and Scholes (1973) argued that if an option is priced correctly at any risk level, it should be impossible to create portfolios consisting of options and their underlying stocks such that the risk of owning the one asset is mitigated by owning the other. In order to derive the formula, certain assumptions were made with regard to the market conditions of both the asset and the option. Based on this, the option's value is dependent only on the stock price and time, the difference between the current and expiration dates.

While the Black-Scholes model is frequently used for option pricing, the validity of the constant volatility and normally distributed asset returns have been challenged. French and Roll (1986) considered stocks listed on American stock exchanges. Volatility was found to be higher during the

trading hours of an exchange rather than during non-trading hours. This led to the conclusion that the availability of information, both public and private, and the process of trading lead to volatility in stock prices. Schwert (1989) showed that the stock price volatility of the Standard & Poor's (S&P) composite index is correlated to the uncertainty in future macroeconomic conditions. This analysis was performed on data between January 1928 and December 1987.

Fama (1965) compared the distribution of the returns of 30 Dow Jones listed stocks to that of a normal distribution. Figure 2.1 describes the findings visually, where empirical and standard normal density functions are shown by the solid and dotted lines respectively. The following was found:

- The empirical density function shows excess observations within 0.5 standard deviations of the mean. This results in the empirical curve having a higher peak than that of the standard normal density function;
- The support of the normal density function has more observations in the 0.5 to 2 standard deviations from the mean compared to the empirical density function. This results in the normal density function presenting a curve with higher variability; and
- The normal density function has fewer observations within more than 2 standard deviations from the mean than that of the empirical density function. This results in kurtosis that exceeds that of a normal density function and the heavier tails observed in stock return distributions when compared to those of the standard normal distribution. The kurtosis of a random variable is a measure of how much the excess tail values exceed that of a normal distribution. Kurtosis is used to compare the tail values of a distribution to that of a normal distribution and is positive if there are excess tail values present. If excess tail values are present, a normal distribution will underestimate the heaviness of the tails.

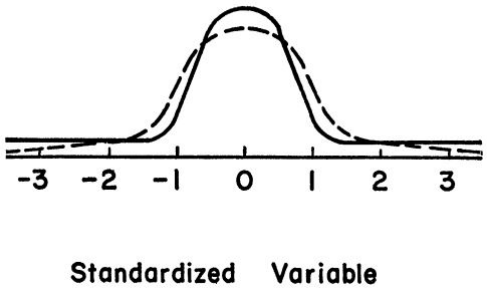

Figure 2.1: Comparison of empirical (solid line) and standard normal (dotted line) density functions

Christie (1982) argued that in financial markets, stock price increases lead to lower volatilities than stock price decreases of the same magnitude. The impact of negative shocks lasts longer, which results in a longer recovery time to the market's pre-shock level. As a result, a symmetric distribution such as the normal distribution may not be appropriate and the model should be adjusted to account for potential asymmetries. When there is bad news, a firm's stock decreases, leading to an increase in

the financial leverage (debt-to-equity ratio), risk of investment in the firm and an increase in future expected volatility, known as the leverage effect. Since a stock index is the weighted sum of the market capitalisation of all stocks on the index, changes in the volatility of a stock would result in changes in the volatility of the index as well.

2.2.2 Statistical Learning

Popular methods for the estimation of volatility include time series models. The ARCH model proposed by Engle (1982) is typically used for econometric applications where the prediction of the future is dependent on different time periods. It recognises unconditional and conditional variance appropriately. The conditional variance is accounted for through the inclusion of past variances in the calculation of the current variance. The clustering of small/large errors is also captured properly by the incorporation of past periods in predicting the future. The ARCH model is simplistic but has the drawback that the inclusion of a long lag may be needed. In addition, a fixed lag structure is also required to ensure positive variance estimates. Bollerslev (1986) proposed the GARCH model, which improves on the ARCH model through the inclusion of lagged variance terms.

The identification of the drawbacks of the GARCH model led to the proposal of the EGARCH model. The drawbacks are with regards to the volatility's reaction to news and the restrictions on some of the parameter estimates (Nelson, 1991). Ding et al. (1993) investigated the correlation structures of the absolute returns of stocks over various exponents, $d > 0$. It was found that over long lags, serial correlation exists. This resulted in the proposal of the APARCH model, which characterises the return series as having a "long memory" property, as well as the incorporation of asymmetry in stock returns as captured by the leverage effect.

Kim et al. (1998) modelled exchange rate volatility using GARCH, Threshold GARCH (TGARCH) and Markov Chain Monte Carlo Stochastic Volatility models. The TGARCH model aims to address the leverage effect by separating the impacts of past shocks into positive and negative shocks. It was found that the return series had a positive skewness and excess kurtosis and as a result the TGARCH model outperformed the standard GARCH, which assumes normality. A likelihood ratio test was performed and it was found that the TGARCH model and stochastic volatility models performed similarly.

Ferreira et al. (2007) used TGARCH, EGARCH, Threshold AR and Multivariate Threshold AR volatility models to assess asymmetric volatility in the Portuguese Stock Market Index, PSI 20. The results obtained from the Portuguese stock market were compared to that of larger stock markets. This was done by comparing results based on the PSI 20 index to results obtained using the ASE 20, CAC 40, DAX 30, FTSE 100, IBEX 35 and S&P 500 indices. The TGARCH model proposed by Glosten et al. (1993) is commonly used to incorporate the leverage effect. Negative changes are assigned a larger weight and lead to a larger impact on estimated volatility. The impacts of past

shocks are separated using a zero threshold. The GARCH-type models captured the asymmetries in the stock markets and in addition showed that the effects from negative shocks to the Portuguese markets take longer to dissipate than those of the larger markets. Markets are defined as efficient when prices reflect available information in a timely manner. This leads to the assumption that the Portuguese market is less efficient than the larger markets it was compared against and that the leverage effect was stronger in the former. Macroeconomic variables are often correlated with the volatility of index returns. The price earnings per share and dividend yield were utilised to determine this relationship in the Portuguese market. With these inputs, the results from the Threshold AR and Multivariate Threshold AR volatility models showed that asymmetric returns are not observed if the effect of changes in macroeconomic variables is controlled. The Threshold AR model is a piecewise linear, regime-switching model where the one-dimensional Euclidean space is divided into k regimes, each with its own AR model (Tong & Lim, 2009). The multivariate Threshold AR is used when threshold nonlinearity exists in a time series (Tsay, 1998). The Schwartz Information Criterion was used to determine the optimal model (Ferreira et al., 2007).

2.2.3 Machine Learning

Bain (1873) proposed a theoretical base for neural networks. He argued that thought and body activity occur due to interactions between neurons in the brain and that the firing of a specific set of neurons is linked to specific activity. The repetition of an activity would result in a strengthening in the connection between the set of neurons, thus resulting in the formation of memory. James (1890) developed a similar theory, but suggested that the flow of electrical currents between neurons in the brain resulted in memories and actions. This implies that individual connections for each memory or action is not required. The theories of Bain and James were tested over the next few years:

- The concept of habituation, where the response to a stimulus decreases after prolonged exposure, was developed by Sherrington (1898);
- The threshold logic model was created based on mathematics and algorithms by McCulloch and Pitts (1943). One branch of research that stemmed from this was the use of neural networks in artificial intelligence;
- Hebbian learning, an unsupervised learning rule that suggests that the persistent stimulation of a synaptic cell results in synaptic efficiency, was created by Hebb (1949); and
- Pattern recognition utilises the use of the perceptron that was created by Rosenblatt (1958) .

Due to constraints in processing power at the time, research on neural networks stagnated after 1969. Recent developments include deep feedforward and recurring neural networks that are applied amongst others, to pattern recognition and machine learning.

Siami-Namini et al. (2018) compared the error rates of estimates obtained using ARIMA and LSTM models. The purpose was to assess the difference in the predictive power of traditional and

deep-learning-based algorithms. The study was conducted using monthly financial time series covering the period January 1985 – August 2018 and economic time series for periods ranging between 1959 and 2017. Error rates were assessed by calculating the Root Mean Squared Error (RMSE) for each method and comparing the change in RMSE as a result of using deep-learning algorithms over traditional methods. The study concluded that LSTM yielded superior estimates. Sensitivity testing was performed on one of the hyperparameters, the number of epochs. It was found to have no effect on the predictive power of the LSTM networks.

The Autoregressive Fractionally Integrated Moving Average (ARFIMA) process is used in time series to model long-memory processes. Bucci (2020) compared the performance of ARFIMA models to that of a selection of ANNs. It was found that the Nonlinear Autoregressive Neural Network with External Input and Long Short Term Memory (LSTM) methods performed better than ANNs such as feedforward and Ensemble Neural Networks that did not have the ability to forecast a long-term dependence. The models performed particularly well during periods that exhibited high volatility. The Diebold-Mariano test, a loss-differential hypothesis test used to determine whether two methods have the same forecasting accuracy, was combined with minimum Mean Squared Error to determine the optimal model. While the ANNs were more difficult to interpret due to the structure of the layers, the performance relative to that of the ARFIMA models showed that the ANNs outperformed the ARFIMA model.

The success of machine learning methods is dependent on multiple factors such as the choice of hyperparameters and the dataset size used to train the algorithm. Time series prediction in the Indian stock market using LSTMs was investigated by Yadav et al. (2020). For this study, stocks were selected from the NIFTY 50 index, an Indian index consisting of 50 stocks. The four largest companies by market capitalisation were selected from the four largest sectors on the index. As a result, four stocks were selected for the period December 2008 – May 2019. The largest sectors on the index were financial services, energy, information technology and automobiles. The sensitivity of the prediction error to the number of hidden layers, a hyperparameter used to optimise the performance of the LSTMs, was of particular interest. It was found that one hidden layer resulted in the minimum RMSE. The importance of specific values for each hyperparameter on the performance of an LSTM was highlighted in this study. The author argued that no bespoke guidelines to determine the optimal hyperparameters of an LSTM exist. The conclusion that the architecture of an LSTM contributes to its performance was made.

2.2.4 Normality Tests

There are multiple methods used to assess the normality assumption of a data sample. Visual aids include the Quantile-Quantile, box and density plots. Summary statistics calculated include skewness and kurtosis. Hypothesis tests are also a popular method to conclude whether the variables in the

dataset are normal.

The Kolmolgorov-Smirnov test determines whether a sample originates from a specified distribution by determining the distance between the cumulative distribution and empirical functions. It is, however, not as powerful for testing normality. One of the shortcomings highlighted by Ghasemi and Zahediasl (2012), is its sensitivity to extreme values, which may result in unreliable results when applied to stock market return series. Anderson and Darling (1954) base their test on the difference between the theoretical and actual distribution functions while Shapiro and Wilk (1965) rely on a regression coefficient. The original Shapiro-Wilk test is only reliable for small samples (typically 50 observations or less). Royston (1992) adjusted the test to increase the maximum sample size to 2000 observations. This was then increased further to a restriction of up to 5000 observations (Royston, 1995). The Jarque-Bera (JB) test presents a joint hypothesis with regards to skewness and excess kurtosis. It uses the first four central moments to calculate the non-negative test statistic. The critical statistic is a Chi-squared distribution with two degrees of freedom (Jarque & Bera, 1980). Thadewald and Büning (2007) compared the JB, Kolmogorov–Smirnov and Cramer-Von Mises tests in terms of power. It was found that the JB test performs best under one of two scenarios:

- When the underlying distribution is symmetrical with medium to long tails; and
- When the underlying distribution is slightly skewed with long tails.

The scenarios under which the JB test outperforms the Cramer–Von Mises and Kolmogorov-Smirnov tests are often characteristic of stock market return series. Recently, Jelito and Pitera (2021) proposed a heavy-tail normality test, which was derived using conditional second moments. This test, referred to as Test Statistic N, was applied to financial returns data and compared to selected benchmark tests. When compared to the Shapiro-Wilk, JB and Andersen-Darling tests, it was shown that Test Statistic N provided the best results based on the total and unique rejection ratios. The JB test performed second best, with results not far from that of Test Statistic N.

The aim of model development is often prediction. Prediction accuracy methods are often used as a relative quality measure to identify the method with the highest predictive power. Any statistical model used to represent an underlying dataset is subject to error. The Akaike Information Criterion (AIC) is used to estimate prediction error and to compare the quality of different models. The aim, therefore, is to minimise the AIC (Akaike, 1998). The Bayes Information Criterion is a method similar to the AIC. Stoica and Selen (2004) concluded that the penalty term is proportionate to the number of variables selected. The Bayes Information Criterion, however, imposes a larger penalty than the AIC. Other methods used to measure prediction accuracy include the RMSE, R-squared and Adjusted R-squared.

2.2.5 Validation Methods

The power of any model lies in its ability to provide accurate estimates based on data that are independent of the data used to train it. To achieve this, the available dataset is often divided into subsets, known as training and test sets. The model parameters are derived from the training dataset, while the test set serves as a source of unseen data used to evaluate model performance. There are multiple validation methods available, which assume independence between observations in the dataset and thus select observations at random (Hastie et al., 2009). In cross-validation methods, the dataset is divided into subsets and used to obtain multiple training and test datasets. Geisser (1975) proposed the well-established k-fold cross-validation method. A dataset is partitioned into k subsets after which training occurs on $k-1$ of the samples, while the k^{th} sample is used as a test set. This process is repeated until all different combinations of the $k-1$ samples have been used to train the model. The results are then averaged to obtain the final estimation error.

With time series data, the temporal order in which observations are observed is important. As a result, the data must be split into subsets that take this order into account. Snijders (1988) proposed a method similar to k-fold cross-validation for time series data, known as blocked cross-validation. This involves the division of the data into k time-continuous blocks, which are aimed at maintaining the temporal order of the data. The application is then similar to that of standard k-fold cross-validation.

Another popular validation method applied to time series data is forward-validation. This results in test datasets that succeed training datasets in terms of the time periods selected. The simplest example of this is holdout forward-validation, where a dataset is partitioned into two mutually exclusive sets over the selected time period. Observations are selected such that no disruption to the temporal order of the data occurs (Devroye & Wagner, 1979).

2.3 Technical Review

The technical review provided in this research report is aimed at providing the reader with only the relevant information required to support the results presented.

2.3.1 Return Series

The return achieved over a specified time period t is used to calculate an asset's volatility. If an asset's price is X_t at time t, the return, log return and absolute log return are calculated using Equations 2.1 – 2.3 (Hull, 2003):

$$R_t = \frac{X_t - X_{t-1}}{X_{t-1}} \tag{2.1}$$

$$r_t = \ln \frac{X_t}{X_{t-1}} \tag{2.2}$$

$$|R_t| = \left| \frac{X_t - X_{t-1}}{X_{t-1}} \right| \tag{2.3}$$

2.3.2 Stationarity and White Noise Processes

- **First-order stationarity:** A stochastic process (such as a time series), $\{Z_t\}$, is first-order or strictly stationary if the joint distribution of $Z_{t_1}, Z_{t_2}, \ldots, Z_{t_n}$ is the same as the joint distribution of $Z_{t_1+k}, Z_{t_2+k}, \ldots, Z_{t_n+k}$ for all positive integers n, all integers k and all time points, $t_i, i = 1, \ldots n$.

- **Second-order stationarity:** First-order stationarity is often too restrictive for real-life application. A stochastic process (such as a time series), $\{Z_t\}$, is second-order or weakly stationary if its expected value and variance do not depend on time, and the covariance depends only on the lag, i.e., $cov(Z_s, Z_t) = cov(Z_{s+k}, Z_{t+k})$ depends only on $t - s$ for all s, t and all integers k.

- **White Noise Process (WNP):** A WNP has the characteristics of a second-order stationary process and a covariance function of the form:

$$\Psi_k = \sigma_k = \begin{cases} \sigma_e^2, & \text{if} \quad k = 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.4}$$

- **Differencing to Achieve Stationarity:** A time series is stationary if its properties do not change with time. As a result, the presence of trend and/or seasonality results in the non-stationarity of a time series. For second-order stationarity, differencing leads to stability in the mean of a time series. Differencing removes trend from a time series by removing variation in the level over time. A first-order differenced series is defined as:

$$\Delta Z_t = Z_t - Z_{t-1} \tag{2.5}$$

To achieve second-order stationarity, more than one difference may be required. In a similar way, seasonal differencing may be required to achieve seasonal stationarity. Transformations such as logarithms can help with second-order non-stationarity.

2.3.3 Correlation Functions

The autocorrelation function (ACF) and partial autocorrelation function (PACF) are useful descriptive tools in time series modelling. Autocorrelation is defined as the correlation between observations and their lags in the same time series of size lag k and is useful to identify repeating patterns in data. For a sequence of observations, Z_1, Z_2, ..., Z_L of length L and a lag of size k, the sample autocorrelation function is estimated as:

$$\hat{\rho}_k = \frac{\sum_{t=1}^{L-k}(Z_t - \bar{Z})(Z_{t+k} - \bar{Z})}{\sum_{t=1}^{L}(Z_t - \bar{Z})^2} \tag{2.6}$$

By plotting the sample autocorrelation coefficients, a graphical representation of $\hat{\rho}_k$ against k, the sample correlogram, is obtained. This can be interpreted as follows:

- If the correlogram decays slowly to zero with many $\hat{\rho}_k$'s significantly different from zero, the time series is non-stationary; and
- If the correlogram exhibits oscillations, the time series contains seasonal fluctuations.

The PACF accounts for terms with shorter lags than accounted for by the ACF. The equations used to calculate the partial autocorrelations were derived by Durbin (1959):

$$\hat{\phi}_{m+1,j} = \hat{\phi}_{mj} - \hat{\phi}_{m+1,m+1}\hat{\phi}_{m,m+1-j} \tag{2.7}$$

$$\hat{\phi}_{m+1,m+1} = \frac{\hat{\rho}_{m+1} - \sum_{j=1}^{m}\hat{\phi}_{mj}\hat{\rho}_{m+1-j}}{1 - \sum_{j=1}^{m}\hat{\phi}_{mj}\hat{\rho}_j} \tag{2.8}$$

2.3.4 ARIMA Class Models

If ϵ_t is presumed to be a WNP, the autoregressive process, AR(m), of order m is defined as:

$$Y_t = \sum_{b=1}^{m}\phi_b Y_{t-b} + \epsilon_t \tag{2.9}$$

Similarly, the moving average process, MA(n), of order n is defined as:

$$Y_t = \epsilon_t + \sum_{a=1}^{n}\theta_a \epsilon_{t-a} \tag{2.10}$$

The ARMA(m, n) process combines an AR(m) process with an MA(n) process:

$$Y_t = \epsilon_t + \sum_{b=1}^{m} \phi_b Y_{t-b} + \sum_{a=1}^{n} \theta_a \epsilon_{t-a} \tag{2.11}$$

Finally, an ARIMA(m, d, n) process is defined as an ARMA(m, n) process where differencing of order d is performed to obtain second-order stationarity. If no differencing is required, the model reverts to an ARMA(m, n) process. ARIMA class models are also known as Box-Jenkins models.

2.3.5 Autoregressive Conditional Heteroscedastic (ARCH)

The Autoregressive Conditional Heteroscedastic (ARCH) model of order p was first proposed by Engle (1982):

$$\begin{aligned} \sigma_t^2 &= \alpha_0 + \alpha_1 \epsilon_{t-1}^2 \sigma_{t-1}^2 + \cdots + \alpha_m \epsilon_{t-p}^2 \sigma_{t-p}^2 \\ &= \alpha_0 + \alpha_1 a_{t-1}^2 + \cdots + \alpha_p a_{t-p}^2 \\ &= \alpha_0 + \sum_{c=1}^{p} \alpha_c a_{t-c}^2 \end{aligned} \tag{2.12}$$

where $\alpha_0 > 0$ and $\alpha_c \geq 0$ for $c = 1, \ldots, p$ to ensure positive variance. The ϵ_t typically follows a standard normal or Student's t-distribution. The ARCH process differentiates between the conditional and unconditional variance, with the unconditional variance updating as a function of past errors. An ARIMA-type process may be applied to the stationary time series to remove any linear dependencies. The residuals from this model, a_{t-p}, provide insights into any potential ARCH effects and the lag, p needed when fitting the ARCH process.

The model structure implies that large historical shocks will result in a large conditional variance, σ_t^2. This will in turn cause a_t to be large. The lagged variance effect may cause a large shock to be followed by further large shocks. The ARCH model assumes that the direction of the shock does not affect volatility. This effect is further exacerbated due to the squaring of past shocks. Negative shocks have a different effect on volatilities and take longer to decrease to the pre-shock level. As a result, ARCH models may therefore overpredict volatility due to their slow response to large shocks (Engle, 1982).

2.3.6 Generalised ARCH (GARCH)

Bollerslev (1986) proposed the Generalised ARCH (GARCH) model. It provides an improvement on the ARCH model of order p by also incorporating lagged conditional variances of order q:

$$\sigma_t^2 = \alpha_0 + \sum_{c=1}^{p} \alpha_c a_{t-c}^2 + \sum_{d=1}^{q} \beta_d \sigma_{t-d}^2 \tag{2.13}$$

where $\epsilon_t \sim \mathcal{N}(0, h_t)$, $p \geq 0$, $q \geq 0$, $\alpha_0 > 0$, $\alpha_c \geq 0$ for $c = 1, \ldots, p$, and $\beta_d \geq 0$ for $d = 1, \ldots, q$.

2.3.7 Exponential GARCH (EGARCH)

Nelson (1991) introduced the Exponential GARCH (EGARCH) model. It has the following form:

$$\ln \sigma_t^2 = \alpha_t + \sum_{i=1}^{\infty} \beta_i g(z_{t-i}) \tag{2.14}$$

where

$$g(z_t) = \theta Z_t + \gamma(|Z_t| - E|Z_t|) \tag{2.15}$$

The EGARCH model addresses the shortcomings of the GARCH model as follows:

- The inclusion of both "sign" and "magnitude" effects introduced through the function $g(z_{t-i})$ in Equation 2.14 produce asymmetry in volatility estimates. The "sign" effect is introduced by the θZ_t term, while the "magnitude" effect is introduced by the $\gamma(|Z_t| - E|Z_t|)$ term in Equation 2.15. θ and γ are coefficients that determine the size of the "sign" and "magnitude" effects. Good and bad news, therefore, has different consequences on changes in volatility;
- The restriction of the β_i parameter to be non-negative has been relaxed in the EGARCH model. The model is therefore less restrictive and oscillatory behaviour is allowed;
- Asset prices are often asymmetrical: bad news results in larger changes in stock prices and volatility than good news (the leverage effect). GARCH models incorporate only the magnitude of changes in asset returns, but not the direction of the change. As a result, GARCH models are therefore more suited for modelling asset returns that are symmetrical and consequently perform worse on asymmetric data; and
- The parameter estimates of the GARCH model are always non-negative, which implies that oscillatory behaviour in volatility is not allowed.

2.3.8 Asymmetric Power ARCH (APARCH)

The APARCH(p, q) model has the following form:

$$\sigma_t^{\lambda} = \omega + \sum_{c=1}^{p} \alpha_c(|\epsilon_{t-c}| - \gamma_c \epsilon_{t-c})^{\lambda} + \sum_{d=1}^{q} \beta_d \sigma_{t-d}^{\lambda} \tag{2.16}$$

where $\epsilon_t = \sigma_t e_t$ with $e_t \sim \mathcal{N}(0, 1)$, $\omega > 0$, $\lambda \geq 0$, $\alpha_c \geq 0$ for $c = 1, \ldots, p$, -1 $< \gamma_c <$ 1 for $c = 1, \ldots, p$, $\beta_d \geq 0$ for $d = 1, \ldots, q$ (Ding et al., 1993).

2.3.9 Goodness of Fit Tests

2.3.9.1 Ljung-Box (LB) Test for Autocorrelation

The LB test is used to test for the presence of serial autocorrelation in the residuals after an ARMA(m, n) model was fitted to the series. It, therefore, considers the dependence of the first moments with a time lag up to lag k. The hypothesis test, with $\hat{\rho}_k$ calculated using Equation 2.6, is as follows:

H_0: $\hat{\rho}_k = 0$
H_A: $\hat{\rho}_k \neq 0$

The test statistic, Q, of length N and lag k is calculated as:

$$Q(k) = N(N+2)\sum_{j=1}^{k}\frac{r_j^2}{N-j} \tag{2.17}$$

where r_j^2 are the residuals at lag j. N is also interpreted as the sample size. The critical value for this test is a chi-squared statistic with significance level α and $\nu = k - m - n$ degrees of freedom. k is the order of the lag, while m and n are the parameters of the ARMA(m, n) model. Rejection of the null hypothesis results in the assumption that the specified process does not fit the data (Box & Pierce, 1970).

The weighted LB test on standardised residuals was adapted by Fisher and Gallagher (2012) and investigates the dependence of a second moment with a time lag. The trace of the k^{th} dimensional correlation matrix is used to provide a weighted sum of squares statistics. The same hypothesis test used for the LB test holds, but since the second moments are considered, this test is useful for determining whether the long-memory components have been modelled sufficiently. The test statistic, $\tilde{Q}$, was adjusted to:

$$\tilde{Q}(k) = N(N+2)\sum_{j=1}^{k}\frac{k-j+1}{k}\frac{r_j^2}{N-j} \tag{2.18}$$

2.3.9.2 Augmented Dickey-Fuller (ADF) Test

The ADF test, developed by Dickey and Fuller (1979), is a hypothesis test to determine whether a time series is stationary. It is adapted from the Dickey-Fuller test to allow for stationarity testing on more complicated time series models. Under the null hypothesis, a unit root is not present and the data are stationary. The alternative hypothesis states non-stationarity in the data. This should be removed by differencing the time series, using Equation 2.5.

The test statistic is calculated as:

$$\Delta Y_t = \alpha + \beta t + \gamma Y_{t-1} + \delta_1 \Delta Y_{t-1} + \cdots + \delta_{m-1}\Delta Y_{t-m+1} + \varepsilon_t$$

where α is a constant, β is the coefficient of the linear trend, γ and δ are the coefficients of the random variable Y_t, m is the order of the AR model, ΔY_t is the first order difference applied to the series Y_t and ε_t is the error term at time t. The test statistic is compared to the critical statistic from the Dickey-Fuller table to determine whether the null hypothesis is accepted or rejected. The null and alternative hypotheses are defined as:

H_0: $\alpha = 1$
H_A: $\alpha < 1$

2.3.9.3 Lagrange-Multiplier (LM) Test for ARCH Effects

Engle (1982) developed an LM test to determine the significance of ARCH effects in a time series. It is useful for determining whether conditional heteroskedasticity is present. The hypotheses are:

H_0: $\alpha_0 = \alpha_1 = \ldots \quad \alpha_p = 0$
H_A: $\exists i \neq j$ such that $\alpha_i \neq \alpha_j$

where α_i $i = 0, \ldots, p$ are the ARCH coefficients. The lag that should be used in the chosen ARCH model is denoted as p. The test statistic of the LM test for the regression on the squared residuals is the F-test with p degrees of freedom. This is compared to a χ^2–distribution with p degrees of freedom. A large critical value leads to the rejection of the null hypothesis.

Li and Mak (2008) argued that a test of the autocorrelations of the squared residuals would result in beneficial conclusions. Fisher and Gallagher (2012) further expanded on this by developing a methodology based on the calculation of a weighted sum of squares of the squared residuals. Both methods use the same hypothesis test as the LM Test for ARCH effects proposed by Engle (1982).

2.3.9.4 Sign-Bias Test

The Sign-Bias Test proposed by Engle and Ng (1993) is used to identify whether a model captured the leverage effect sufficiently. It regresses the squared standardised residuals onto lagged shocks resulting from both good and bad news:

$$\hat{z_t}^2 = \iota_0 + \iota_1 \cdot I_{\{\hat{\varepsilon}_{t-1}<0\}} + \iota_2 \cdot I_{\{\hat{\varepsilon}_{t-1}<0\}}\hat{\varepsilon}_{t-1} + \iota_3 \cdot I_{\{\hat{\varepsilon}_{t-1}\geq 0\}}\hat{\varepsilon}_{t-1} + u_t \tag{2.19}$$

where $\hat{\varepsilon}_{t-1}$ are the residuals from the estimated model at time $t-1$, ι_0, ι_1, ι_2 and ι_3 are the regression coefficients, u_t is the regression residual and I is the indicator function defined based on the sign of the estimated residuals:

$$I_{\{\hat{\varepsilon}_{t-1}<0\}} = \begin{cases} 1, & \text{if} \quad \hat{\varepsilon}_{t-1} < 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.20}$$

$$I_{\{\hat{\varepsilon}_{t-1}\geq 0\}} = 1 - I_{\{\hat{\varepsilon}_{t-1}<0\}} \tag{2.21}$$

The indicator function allows for the inclusion of volatility return shocks not accounted for in the specified model. The hypotheses are as follows:

H_0: $\iota_0 = \ldots \quad \iota_3 = 0$
H_A: $\exists i \neq j$ such that $\iota_i \neq \iota_j$, where $i, j = 0, 1, 2, 3$

The positive Sign-Bias test considers only the presence of positive shocks through the indicator function $\hat{\varepsilon}_{t-1} \geq 0$. The negative Sign-Bias test only utilises $\hat{\varepsilon}_{t-1} < 0$. The joint effect considers both positive and negative shocks.

2.3.9.5 Jarque-Bera Test

The JB test for normality was proposed by Jarque and Bera (1980). The test statistic incorporates the skewness and kurtosis of the variables in the data and is calculated as:

$$JB = \frac{N}{6}\left(S^2 + \frac{1}{4}(K-3)^2\right)$$

where N is the number of observations and S and K are the skewness and kurtosis of the variables respectively. The critical statistic is based on a χ^2–distribution with two degrees of freedom. The null hypothesis is a joint hypothesis of 0 skewness and 0 excess kurtosis in the data. Rejection of the null hypothesis results in the conclusion that the variables in the data are not normally distributed.

2.3.10 Artificial Neural Networks (ANN)

ANNs were developed with the aim of imitating the structure of the human brain. The underlying structure consists of a collection of nodes, called artificial neurons, that process the signals they receive and transmit to other neurons connected to them. Each connection is assigned a weight which determines the strength of the signal. The weights are updated as learning takes place. For each node other than the input nodes, the weighted sum of the inputs is converted to an output with the aid of an activation function (Gers et al., 2000).

The first set of nodes that receives inputs into the system for further processing is known as the input layer. All processing and calculations are performed in the next set of nodes, known as the hidden layer. ANNs learn by creating probability-weighted associations between a known input and its output within each node. These associations are stored within the network itself. The sum of the weights is input into an activation function to obtain the output in a process known as forward propagation. The weights are updated from the left of the network to the right. An alternative method is backward propagation, where the weights are updated according to a specific learning rule and an error value (loss). With backwards propagation, the weights are updated from right to left in the network. The error value is calculated as the difference between the value predicted by the ANN and the actual output value. The training process is terminated based on convergence to pre-defined criteria. The output layer provides the final results produced by the processing steps performed in the hidden layer(s) (Miljanovic, 2012).

The relationships between the nodes within each layer determine the structure of the ANN. Feedforward neural networks move information between layers in a forward direction only. Recurrent

neural networks allow signals to travel in both directions, thus creating loops in the system. This is often used when past predictions are used to make a current prediction (Miljanovic, 2012). These two networks are shown in Figure 2.2.

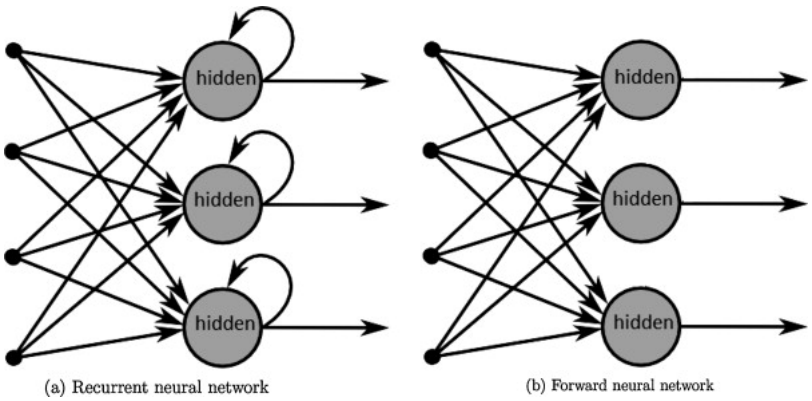

Figure 2.2: Recurrent and feedforward ANN structures

The architecture of an ANN determines the results obtained by the ANN. An important aspect is the hyperparameters selected. Hyperparameters control the results of the training process and are updated at each training iteration. The starting values are selected upfront and refined until the process is terminated. There are multiple hyperparameters that affect an ANN. A neural network with three or more layers is known as a deep-learning neural network and due to its complexity, has more hyperparameters (Goodfellow et al., 2017).

Activation functions derive outputs from a given set of inputs provided to a node. Each activation function has its own benefits and limitations that affect the obtained outputs. For the purposes of this research report, the sigmoid/logistic function was selected:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.22}$$

where $f(x) \in [0, 1]$. The function is presented visually in Figure 2.3. A sigmoid function is continuously differentiable, which means that its derivative is also continuous. It is not symmetric around zero and therefore requires scaling to ensure that the output values can have different signs. Popular scaling methods include normalisation and min/max scaling, which ensure that the range of the features is constrained, usually to [0,1] or [-1,1] (Sharma et al., 2017).

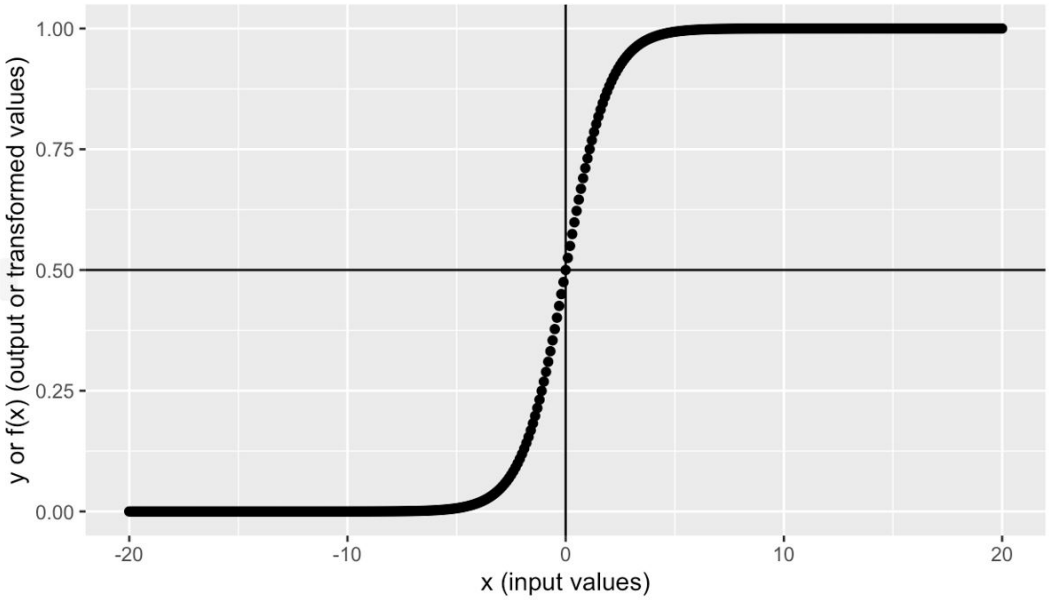

Figure 2.3: Sigmoid function

The minimisation of the loss function is achieved when the function reaches a local minimum. In order to obtain the minimum, an optimisation algorithm is used. The learning rate determines the speed at which the algorithm approaches the minimum and therefore the convergence to a solution. The learning rate is usually confined to a value between 0 and 1. A larger value will result in the model converging more quickly but may result in overfitting. A smaller learning rate will cause the model to take longer to converge (Goodfellow et al., 2017). Figure 2.4 shows a difference in convergence to the local minimum when small and large learning rates are used. The small learning rate requires more iterations of the algorithm to obtain the local minimum. The large learning rate achieves a minimum faster, but it may not be the true local minimum.

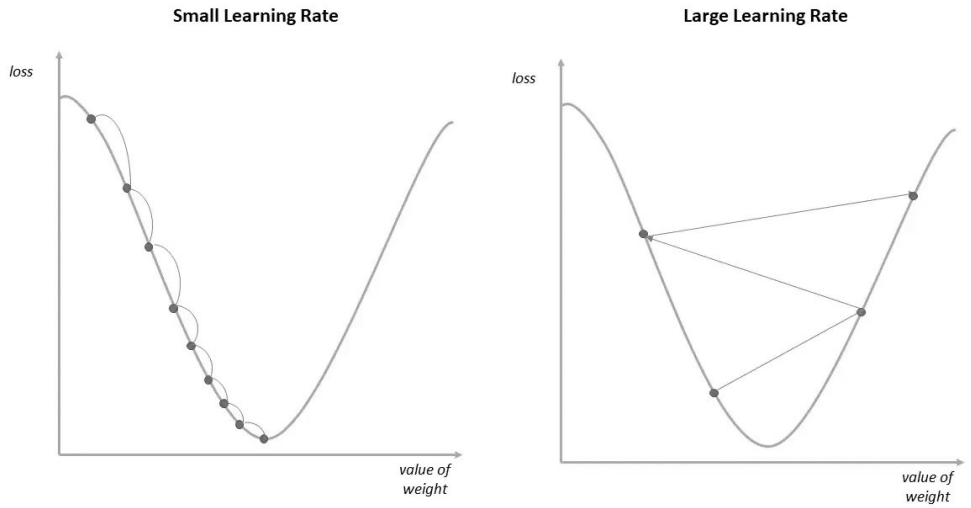

Figure 2.4: Small vs large learning rate comparison

An epoch occurs when an entire dataset is passed through the neural network exactly once. When a sizeable dataset is used, it may be too large to pass through the neural network, it may be divided into smaller samples. These samples are known as batches and the sample size is referred to as the batch size (Goodfellow et al., 2017).

The back propagation algorithm may suffer from exploding or vanishing gradients depending on which activation function is used. Exploding gradients occur when the assigned weights grow too large, while vanishing gradients occur when the weights are too small. Both cases result in instability in the neural network. Weight decay assists in training a neural network that performs well when applied to an independent dataset by restricting the change in weights at each update (Krogh & Hertz, 1991).

2.3.11 Long Short Term Memory (LSTM) Networks

A popular example of recurrent neural networks is the Long Short Term Memory (LSTM) model. It consists of cells and gates. The three gates (known as input, output and forget gates) control the movement of information through the cell, allowing specific values to be captured at different time intervals (Gers et al., 2000).

The memory cell allows for the recurrence of an input. Memory cells are contained in the hidden layer of the LSTM network and there can be multiple memory cells per block. A linear unit, known as the Constant Error Carousel (CEC), allows for the recurrence of an input. Irrelevant noise and inputs that enter a cell may cause disruptions in the system. To control this, input and output gates are used. If the gates are closed, the activation of the cell is zero. The state of the cell is updated using the following components:

- The present state of the cell, s_{curr};
- The input to the cell , n_c;
- The information conveyed to the input gate, n_{in} ; and
- The information conveyed to the output gate, n_{out}.

At each time step $t = 1, 2, \ldots$, the cells are updated and the error for each weight is computed. Information passes into the k^{th} cell via the input gate n_{in} after being adjusted by an activation function f_{in_k}. A popular choice for the activation function is the sigmoid function.

For a signal to move from the input gate to the next unit, k, a weight is calculated. Signals move from the input gate to the m^{th} unit by adjusting a weight, $w_{in_k m}$. The weight is then applied to the

information at time $t-1$ to convey the information to the input gate. The input gate activation in the j^{th} block with k memory blocks and n memory cells, is thus computed as follows:

$$\mathrm{n}_{in_k}(t) = \sum_m w_{in_k m} y^m(t-1); \quad y^{in_k}(t) = y_{in_k}(t) f_{in_k}(\mathrm{n}_{in_k}(t)) \tag{2.23}$$

The output gate activation is computed in a similar fashion, with a different set of weights, $w_{out_k m}$, and function, f_{out_k}:

$$\mathrm{n}_{out_k}(t) = \sum_m w_{out_k m} y^m(t-1); \quad y^{out_k}(t) = y_{out_k}(t) f_{out_k}(\mathrm{n}_{out_k}(t)) \tag{2.24}$$

The input to the cell, which is adjusted by a function g, follows in a similar fashion as discussed for Equations 2.23 and 2.24:

$$\sum_m w^n_{c_k m} y^m(t-1); \quad y^{out_k}(t) = y_{out_k}(t) f_{out_k}(\mathrm{n}_{out_k}(t)) \tag{2.25}$$

The gated input and state at time step $t-1$ is used to determine the internal state of the memory cell. This holds for for $t > 0$:

$$s_{c^k_n}(0) = 0; \quad s_{c^k_n}(t) = s_{c^k_n}(t-1) + y^{in_k}(t) g(\mathrm{net}_{c^n_k}(t)) \tag{2.26}$$

The cell output is then adjusted by an activation function h, with a multiplicative gate added to the output gate function:

$$y^{c^n_k}(t) = y^{out_k}(t) h(s_{c^n_k}(t)) \tag{2.27}$$

The use of lags of different lengths imply that LSTM models may provide good results when used for time series modelling (Gers et al., 2000).

Chapter 3

Methodology

3.1 Introduction

The first step in statistical and machine learning is to understand the underlying data. This ensures that the data are modelled accurately. The data source's background, including the origination of the data, is summarised in Section 3.2.

Section 3.3 describes the process followed when splitting the dataset into pre-COVID and COVID datasets. The pre-COVID dataset, which accounts for most of the observations, is then further divided into training and test sets.

The aims, objectives and outcomes of research are brought together by a well constructed methodology outline. This helps the researcher to plan each step in the research process and meet the outlined objectives. This is presented in Section 3.4

3.2 Data Source

The Johannesburg Stock Exchange (JSE) currently located in Sandton, South Africa, was formed in 1887 and is the largest stock exchange in Africa. It encourages the free trade of securities in the market and serves the purpose of channelling funds into the economy and providing investors with dividends as a return on investment. Trading hours are on all business days, excluding holidays declared by the JSE. Almost 400 companies are listed on the exchange and it has a market capitalisation (market cap) of more than $1,000bn. The market cap of an exchange is determined by first calculating the market cap of each of the listed companies as the total number of issued shares times the current stock price of a single share. The company market caps are then added to obtain the market cap of the exchange.

Exchanges typically provide investors with a variety of investment options, depending on the needs of the investors. Indices, which consist of a subset of the shares listed on the exchange, are a popular investment option. The All Share Index (ALSI) is one of the indices traded on the JSE and consists of the shares of the 164 companies with the largest market cap. This equals approximately 99% of total market cap, since the JSE has a few companies that contribute to most of the market cap of the stock exchange.

The data used in this document are publicly available and were sourced from the JSE's website (source: https://www.jse.co.za/data/historical-data; access date: 01 July 2020). The dataset comprises the following variables:

- Date: The date at which the ALSI is observed. Only dates that correspond to trading days are captured;
- Price: The price of the ALSI when trading ends for the day;
- Open: The price at which the ALSI was trading at when the JSE opened for trading on that particular day;
- High: The maximum price at which the ALSI was trading during the day;
- Low: The minimum price at which the ALSI was trading during the day;
- Volume: The volume of traded shares during the trading day; and
- Chq%: The daily relative change in the price of the ALSI.

The *Price* variable is used to create the variables shown in Equations 2.1 – 2.3.

The frequency of trading data available for modelling purposes depends on the stock exchange. The most readily available data sources are typically daily and consist of a single opening and closing price. Intraday trading data capture movements in stock prices at a higher frequency, such as quarter-hourly or hourly. Due to the availability of data, intraday trading data was not considered for this research report.

3.3 Dataset Split

The daily closing prices of the ALSI for the period June 2009 – May 2020 (referred to as the ALSI dataset) were used. This period was selected to exclude the effects of the 2008/09 Financial crisis and the COVID-19 crisis. This resulted in a total of 2 737 observations. The dataset was split as follows:

- The ALSI dataset was grouped into a pre-COVID (4 June 2009 – 29 November 2019) and a COVID (2 December 2019 – 18 May 2020) dataset. The models were developed on the pre-COVID dataset as this provides data prior to a regime switch as a result of COVID. Since the outbreak of COVID-19 in December 2019, data from 01 December 2019 to 18 May 2020 were considered as COVID data; and

- The pre-COVID dataset was further divided into training and test datasets under the holdout approach. The splits are 80% and 20% respectively.

The dataset sizes and time periods for each dataset are summarised in Table 3.1. The holdout approach and variable creation were applied prior to importing the data into R.

Table 3.1: Datasets used in this research report

Dataset	**Date**	**Volume**
Training	4 Jun 2009 – 25 Oct 2017	2 099
Test	26 Oct 2017 – 29 Nov 2019	525
COVID	2 Dec 2019 – 18 May 2020	113

3.4 Data Analysis and Summary Statistics

Section 2.2.1 discussed the Black-Scholes and Market Theories, with a focus on the asymmetrical distribution, leverage effect, volatility clustering and long-memory properties of asset returns. To model this accurately, the data analysis performed needs to provide evidence for or against any potential trends. All results presented were obtained using R software, Version 4.1.3. The packages used are referenced in the relevant sections in Chapter 4.

To analyse the normality of the return series, summary statistics, probability plots and hypothesis tests were used. The Quantile-Quantile (QQ) plot provided a visual aid used to assess the normality assumption of the data.

Finally, the JB test was considered to test the assumption of normality. The results presented by the summary statistics and probability plots showed whether the conditions under which the JB test outperforms other tests (highlighted in Section 2.2.4) hold for the ALSI dataset. If the conditions do not hold, other normality tests will be considered. These measures and the results are discussed in more detail in Section 4.2.

3.5 Stationarity of Time Series Data

Time series data are required to be second-order stationary prior to fitting a specific model. Second-order stationarity was defined in Section 2.3.2. The ACF and PACF were used to analyse stationarity for multiple lags of the time series. The calculation of the ACF and PACF was discussed in Sections 2.3.3 The degree of volatility persistence/clustering was investigated by calculating the autocorrelation of $|R_t|^a$ over multiple lags and for positive values of a. A series of significant

autocorrelations with a slow decaying trend over long lags is evidence of volatility persistence.

Unit root stationarity was tested using the ADF test described in Section 2.3.9.2. In addition to unit root stationarity, analysis is usually performed to investigate any trends and seasonal components present. For the purposes of this research paper, Seasonal and Trend Decomposition using Loess (STL) was used. This is presented in Section 4.2.3. The *nsdiff()* function was used to determine the number of seasonal differences required.

3.6 Specification of the Mean Equation

The specification of an appropriate mean equation to explain the serial dependence in the first moment of the process is the first step of the ARCH modelling process. The ACF and PACF are used to determine the values for m and n of the ARMA(m, n) model. This is done by considering whether the functions show a slow decay or significance at specific lags:

- If the series decays to zero, an AR model is appropriate;
- If there are one or more spikes in the series, an MA model is appropriate; and
- If the series is mainly decaying within a few lags, an ARMA model is appropriate;

The residuals were then tested using a Ljung-Box (LB) test to determine whether the residuals are independent. Independence in the residuals is required to conclude that a particular ARMA-type model will fit the data accurately. This is discussed in more detail in Section 4.2.4.

3.7 Testing for ARCH Effects

The Box-Jenkins/ARIMA approach assumes homoskedastic residuals. Prior to fitting conditional heteroskedasticity models, the LM test described in Section 2.3.9.3 was used to determine whether the residuals of the mean model remain constant over time. In addition, the squared residuals were plotted and investigated for trends prior to applying heteroskedasticity models, as seen in Section 4.3.2. If heteroskedasticity was present, the next step was to apply GARCH-type models.

3.8 Statistical Learning Models Fitted

ARCH processes are the first of the GARCH-type models developed. As a result, ARCH models can oftentimes be used as a baseline model for comparative purposes. ARCH-type models assume a symmetrical distribution, but can be fitted with a skew conditional distribution. ARCH processes are typically used when either an AR or MA model describes the mean equation. When an ARMA-type process is used, the GARCH model is more appropriate to model volatility. As a result, the baseline

model used for ARMA-type mean equations is the GARCH model. The specification of the mean equation would assist in determining whether an ARCH- or GARCH-type model is more appropriate. Similar to the ARCH model, the GARCH model assumes symmetry which can be addressed by fitting a skew conditional distribution. ARIMA class models are discussed in more detail in Section 2.3.4.

Once the baseline ARCH or GARCH model has been determined, the next step is to address the long memory and asymmetrical properties of stock price returns. To achieve this, an APARCH model was used and compared to the baseline model. The potential improvement in model fit, as measured by comparing the RMSEs of the APARCH and baseline models, is then attributed to the model capturing the long memory and asymmetry trends accurately. ARCH, GARCH and APARCH models and their differences are discussed in Sections 2.3.5 – 2.3.8.

The aim of the EGARCH model is to capture the leverage effect which results in asymmetry of stock prices. Similar to the methodology followed for the APARCH model, the EGARCH model was be compared to both the baseline and APARCH models. The technical review of the comparison is summarised in Section 2.3.7.

3.9 Machine Learning Models Fitted

3.9.1 Further Data Analysis

The LSTM model requires input data to contain both a target variable and a predictor. This was achieved by lagging the series by one step. The lagged series was then set as the target variable, while the unlagged variable was used as a predictor.

The activation function considered in this research report was the sigmoid function. It has a range of [0,1]. Min-max scaling/normalisation is useful to scale data to be within the range of [0,1]. The data were therefore normalised to ensure that the predictor was within the activation function's range.

3.9.2 Hyperparameters used in the LSTM Model

LSTM neural networks are used in deep learning for their feedback properties. This enables the neural network to process both single data points, as well as sequences of data and adds a memory component to the estimate. The purpose of training a neural network is to obtain accurate predictions when introduced to new data. This is controlled for by using hyperparameters in the model such that it is trained optimally in terms of run time and fit. Hyperparameters are used to change the model until an optimal fit is obtained and are thus subject to change throughout the training process. The initial trained model relies on pre-selected starting values which are updated at each iteration until the

optimal fit is obtained. As a result, LSTMs are highly dependent on the choice of hyperparameters as well as the architecture of the network.

The first hyperparameter introduced in the model was the batch size. The batch size specifies the number of observations used to learn from before the internal model parameters are updated. The training dataset can be partitioned into one or more batches as follows:

- A batch size equal to the number of observations in the training set is known as Batch Gradient Descent;
- When a single observation is used, it is known as Stochastic Gradient Descent (SGD); and
- Mini-Batch Gradient Descent occurs when the batch size is between one and the total number of observations in the training set. Batch sizes of 32, 64 and 128 are often used.

SGD significantly reduces the computation time of the algorithm by finding minima or maxima from the single selected observation. It updates weights more frequently than with batch gradient descent, but suffers from noisier errors. In time series analysis, predictions are prone to decreased accuracy over time. To improve this, walk-forward training is often used. In this method, the model is retrained every time new data become available. This ensures that the most recent data are used to obtain more accurate predictions. For the purposes of this research report, a batch size of one (SGD) was used. This allowed for walk-forward training as the model parameters were updated after each new observation.

The second hyperparameter that was used in the model was the number of epochs. This is the number of iterations the learning algorithm performs when training on the entire dataset. A large number of epochs is often used to allow the learning algorithm to train sufficiently. The optimal number of epochs was selected based on the minimum error. The effect of the number of epochs on the error rate was investigated by performing a sensitivity analysis. The RMSE was calculated at different values for the epoch hyperparameter and compared.

The third hyperparameter considered was the learning rate. The speed at which the model adapts to the problem is determined by the learning rate. An inverse relationship exists between the learning rate and the number of epochs. The larger the learning rate, the faster the model converges to a solution. This solution may be sub-optimal as the model was not allowed to train properly. When the learning rate is too small, the model may not converge to a solution. A sensitivity analysis similar to that performed for the epoch hyperparameter was performed as part of this research paper.

Finally, the fourth hyperparameter used in the model was weight decay. Weight decay is used to prevent the overfitting of the model. This is done by keeping weights small, which avoids the exploding gradient problem where weights grow too large. A larger learning rate is chosen initially and then decreased multiple times. For the purpose of this study, a value of 1×10^{-6} was used.

Activation functions are used in neural networks to assist with the learning of patterns from the data. It uses the output signal from a preceding cell and transforms it into an input to the next cell. A popular activation function for time series models is the sigmoid function. The *tensorflow* and *keras* packages in R are used for the analysis. Alternative activation functions were not investigated for this research report.

Hidden layers are layers in between input and output layers where calculations are performed. Section 2.2.3 discussed hidden layers in more detail. A single hidden layer was considered for the purpose of this research.

The optimal training model was selected based on minimum RMSE. It was then applied to the test dataset to allow for comparison to the statistical learning methods discussed in Section 3.8.

Chapter 4

Analysis and Results

4.1 Introduction

Exploratory data analysis is performed to provide insights into a dataset. Summary statistics are useful to identify the shape of the underlying data and form an integral part of exploratory data analysis. Outliers are identified using tools such as box plots, while missing observations can be identified and treated by imputing the value or by deleting the observation. In this research report multiple analyses are performed to ensure that the time series data are modelled correctly:

- Section 4.2.1 provides summary statistics useful to determine if the data follow a normal distribution and is symmetrical;
- The ACF and PACF of the return series are investigated in Section 4.2.2 to determine whether there is correlation present. In addition, the ACF and PACF are calculated using different transformations over longer lags to determine whether mean-reversion and long-term memory are observed in the return series. This is useful when using an APARCH model;
- Stationarity analysis is performed with the help of STL. The data are divided into trend, seasonal and random fluctuation components to ensure that any trends in the data are accounted for. Section 4.2.3 presents these results;
- Section 4.2.4 presents the specification of the mean equation used in the ARCH modelling process. Three different ARIMA-type models, selected based on the ACF and PACF, are presented; and
- Finally, Section 4.2.5 tests for ARCH effects both visually and with the help of the LM test.

4.2 Data Analysis

4.2.1 Summary Statistics

Table 4.1 shows the summary statistics of the ALSI return series (R_t), the log return series (r_t) and the absolute return series ($|R_t|$). All results are obtained using the *moments* package (Komsta & Novometsky, 2022). The following observations are made:

- Kurtosis equals 3 if the data are normally distributed. The kurtosis of all three time series exceed this value. The kurtosis for R_t, r_t and $|R_t|$ are 10.33, 11.03 and 22.97 respectively and therefore indicate excess values in the tails, thus leading to heavy tailed behaviour;
- The skewness of a normal distribution is 0. The skewness of the time series R_t, r_t and $|R_t|$ are -0.53, -0.68 and 3.09 respectively, which are very different from 0. The negative skewness associated with R_t and r_t shows that the returns of the ALSI exhibit the leverage effect, which implies longer tails to the left of the distribution. The positive skewness exhibited by $|R_t|$ is expected since only positive values are presented in this time series; and
- For all three time series, the p-value of the JB test is very small (<0.001). At a 95% confidence level, this leads to a rejection of H_0. The assumption of normality is therefore rejected.

Table 4.1: Summary statistics of the ALSI returns, log returns and absolute returns

Dataset	Mean	Std Deviation	Skewness	Kurtosis	Minimum	Maximum	JB p-value
R_t	0.0003	0.01	−0.53	10.33	−0.1	0.08	<0.001
r_t	0.0003	0.01	−0.68	11.03	−0.1	0.07	<0.001
$\|R_t\|$	0.0077	0.01	3.09	22.97	0.0	0.10	<0.001

Figure 4.1 shows the QQ–plot of R_t, obtained using the *tidyverse* package (Wickham, 2023). Deviation from normality is observed in the tails of the sample distribution. The deviation of the scatterplot from the straight line indicates that R_t exhibits heavy tails. Based on the spread of the observations in the tails, it is noted that the right tail is shorter than the left tail, indicating the presence of the leverage effect.

One of the conditions under which the JB test performed better than the benchmark tests is when the underlying distribution is slightly skewed with long tails. The skewness of R_t is summarised in Table 4.2.1 as -0.53, which indicates that the distribution is slightly negatively skewed. The kurtosis of 10.33 and the QQ-plot in Figure 4.1 showed the heavy-tailed behaviour of the data. As a result, it can be concluded that the distribution is slightly skewed with long tails. The JB test is therefore a reliable test for normality on the ALSI dataset.

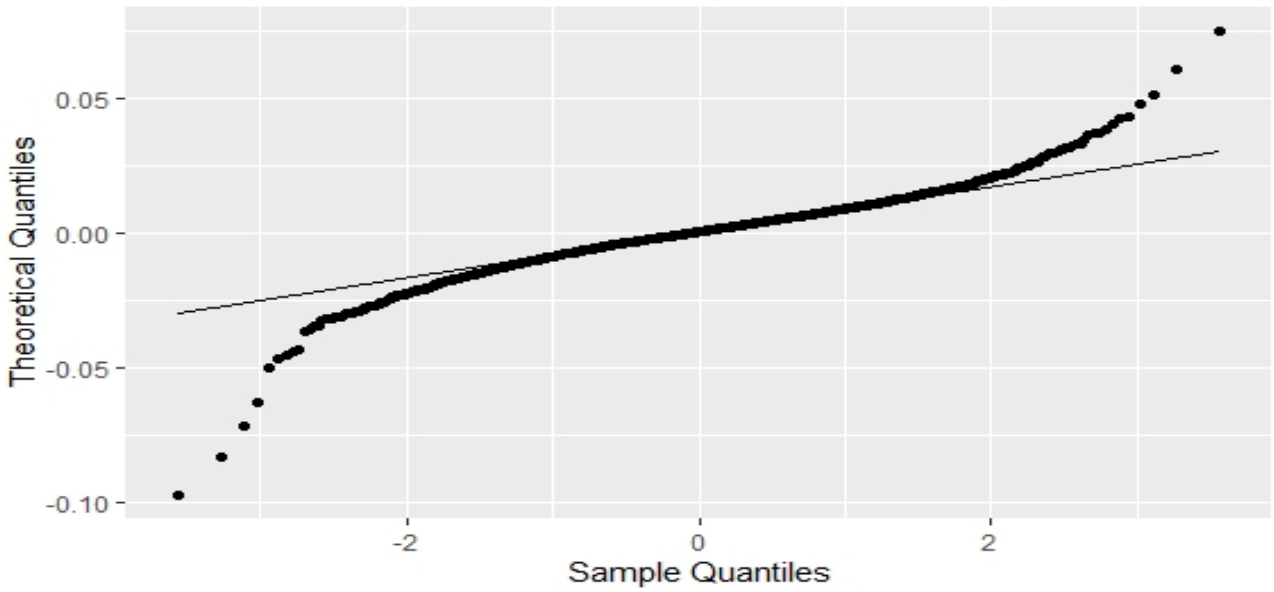

Figure 4.1: Normal Q-Q Plot of R_t

Figures 4.2 – 4.4 show the ALSI time series R_t, r_t and $|R_t|$ for the period April 2009 – June 2020, calculated using Equations 2.1 – 2.3. An upward trend in R_t can be observed for most of the period, followed by a large decline at the start of 2020, associated with the COVID-19 crisis. There is no visible seasonal trend in the data, with r_t showing stability around the mean, $\bar{r_t} \approx 3 \times 10^{-4}$. Figure 4.4 is consistent with the observation of Fama (1965) . The market shows volatile behaviour over time, with evidence of volatility clustering. As a result, the data should be modelled using a time-varying volatility structure.

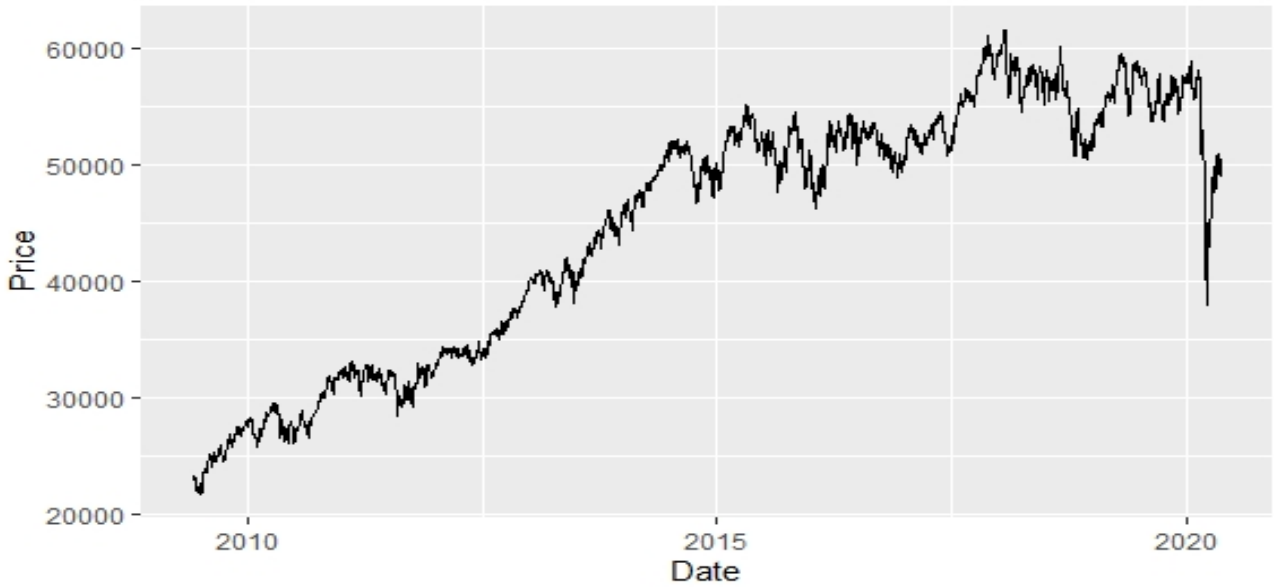

Figure 4.2: ALSI daily returns time series, R_t (2009 to 2020)

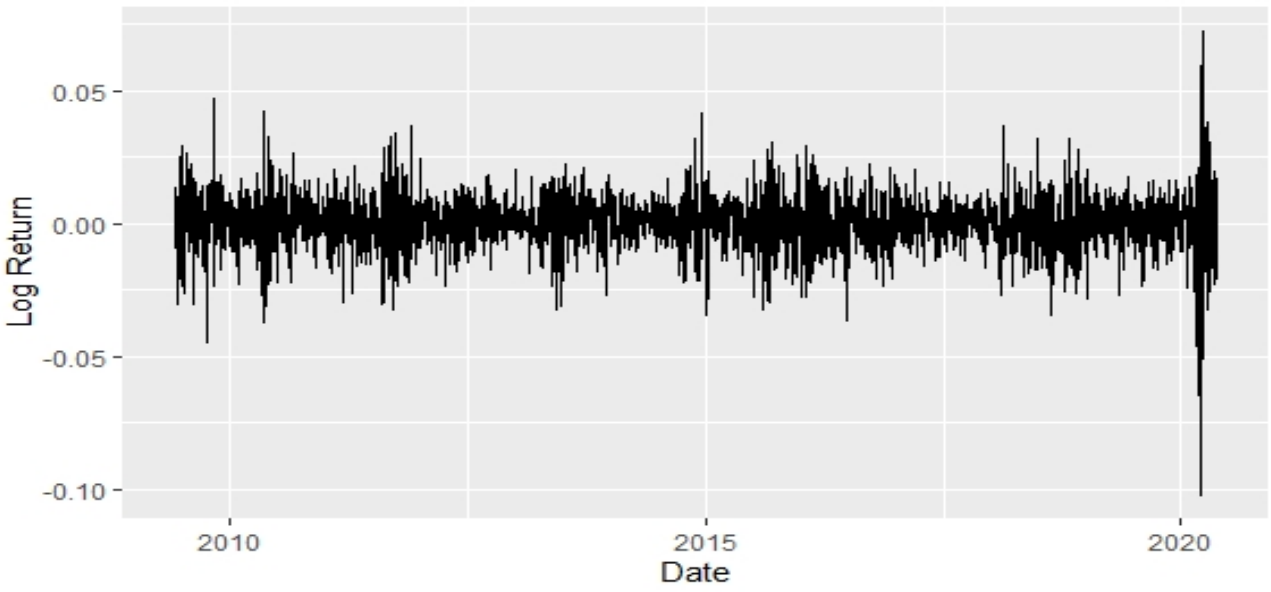

Figure 4.3: All Share Index daily log returns time series, r_t (2009 to 2020)

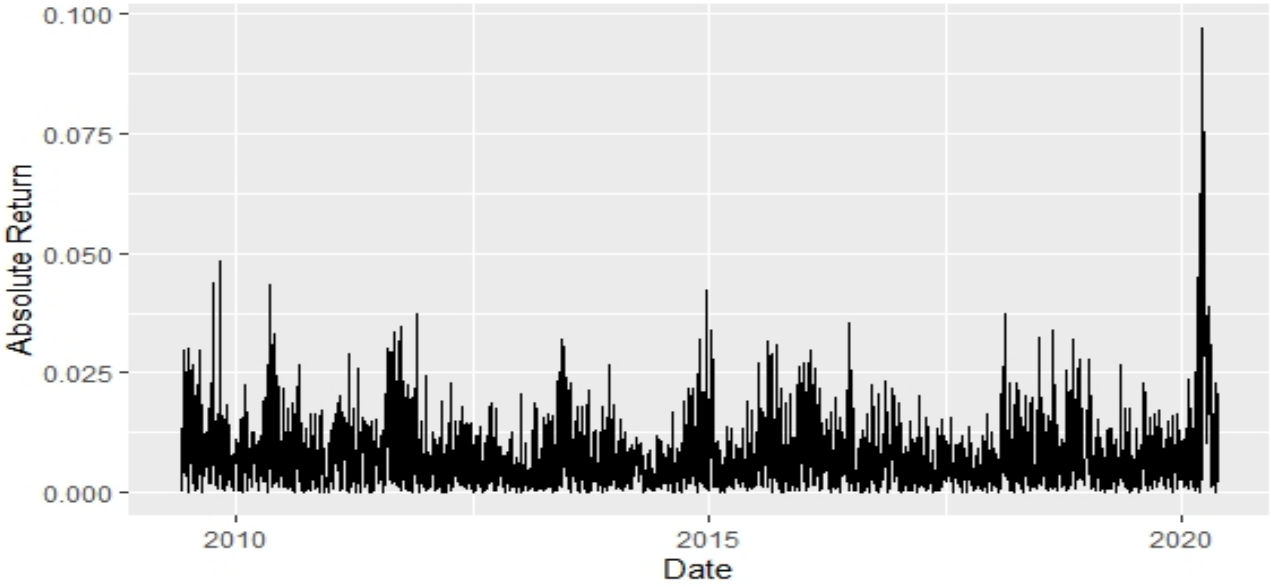

Figure 4.4: All Share Index absolute daily returns time series, $|R_t|$ (2009 to 2020)

4.2.2 Autocorrelation Analysis of the Return Series

The concept of an efficient market was defined in the glossary. While the returns of stocks are not serially correlated, independence is not guaranteed (Fama, 1965; Samuelson, 1973). Financial models often assume that the returns are independent and identically distributed (IID). The autocorrelation analysis of the return series is useful to investigate the correlation between the returns on a stock, R_t, over different lags, R_{t+k}, for all non-negative integers t and k.

Table 4.2 shows the sample autocorrelations of R_t, r_t and $|R_t|$ for lags 1 – 5, 10, 20, 40, 70 and 100. The first lag autocorrelation for the return and log return series, R_t and r_t, is negative and not significant at a 95% confidence level. At the third lag, the autocorrelation is positive and significant at 0.06. The autocorrelation at the fourth and twentieth lags are negative. The alternating positive and negative autocorrelations support the mean-reversion behaviour of stocks. Mean-reversion occurs when stock price returns and volatilities revert to their long-term average levels after short-term fluctuations around this average. The ACF and PACF for the ALSI return series are shown in Figures 4.5 and 4.6. The ACF shows a significant correlation spike at lag three, while the PACF shows significant correlation at lags three and six. Both the ACF and PACF were obtained using the *acf()* and *pacf()* functions in the *stats* package (Komsta & Novometsky, 2022).

Table 4.2: Autocorrelations over different lags for R_t, r_t and $|R_t|$

	Lag									
Series	**1**	**2**	**3**	**4**	**5**	**10**	**20**	**40**	**70**	**100**
R_t	−0.032	−0.021	0.060	−0.012	0.006	0.009	−0.017	0.015	0.009	0.004
r_t	−0.033	−0.022	0.059	−0.013	0.008	0.009	−0.018	0.013	0.008	0.003
$\|R_t\|$	0.612	0.651	0.617	0.640	0.609	0.593	0.577	0.507	0.471	0.455

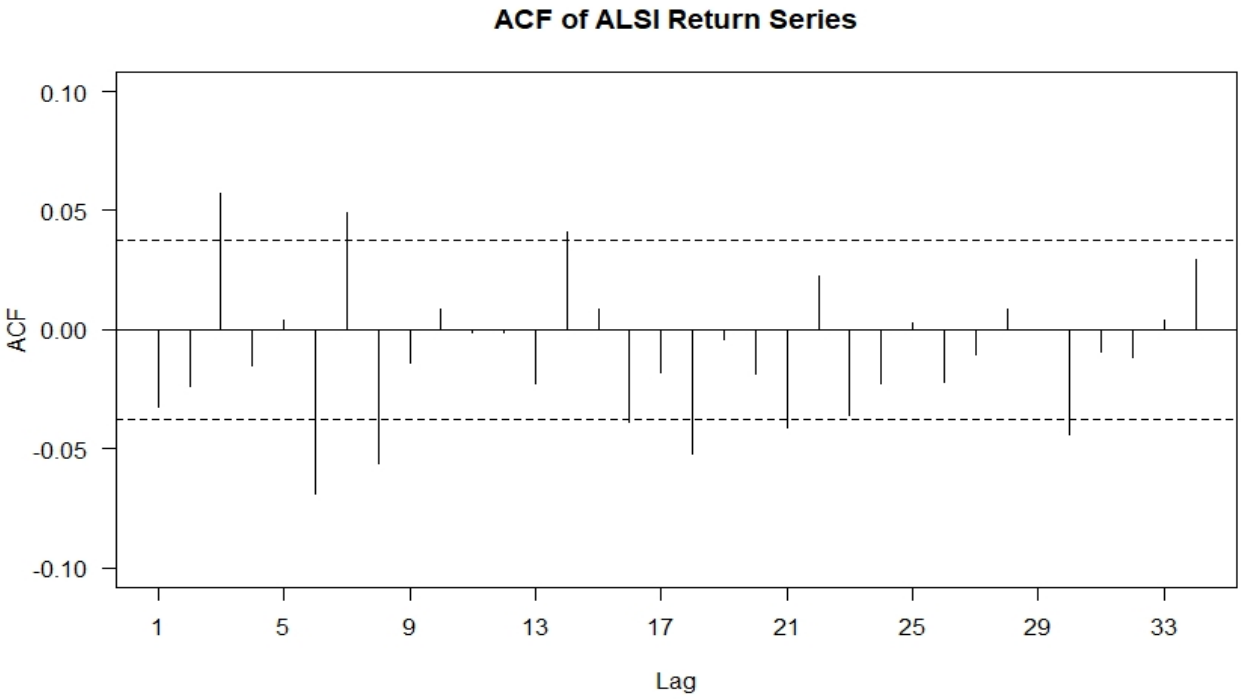

Figure 4.5: ACF of ALSI with 95% confidence level

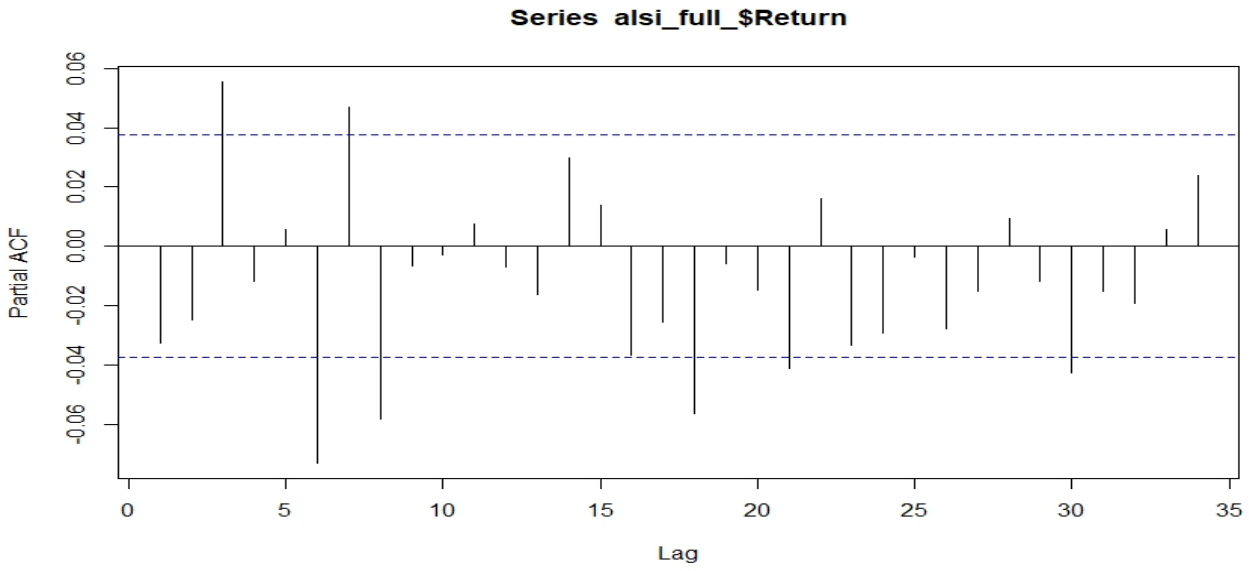

Figure 4.6: PACF of ALSI with 95% confidence level

The degree of volatility persistence is investigated by calculating the autocorrelation of $|R_t|$ and then plotting the ACF and PACF. From Table 4.2 and Figures 4.7 – 4.8, the slow decay and significance in the autocorrelation show that volatility persistence is present. Table 4.2 shows that the autocorrelation of $|R_t|$ is significant up to a lag of at least 100.

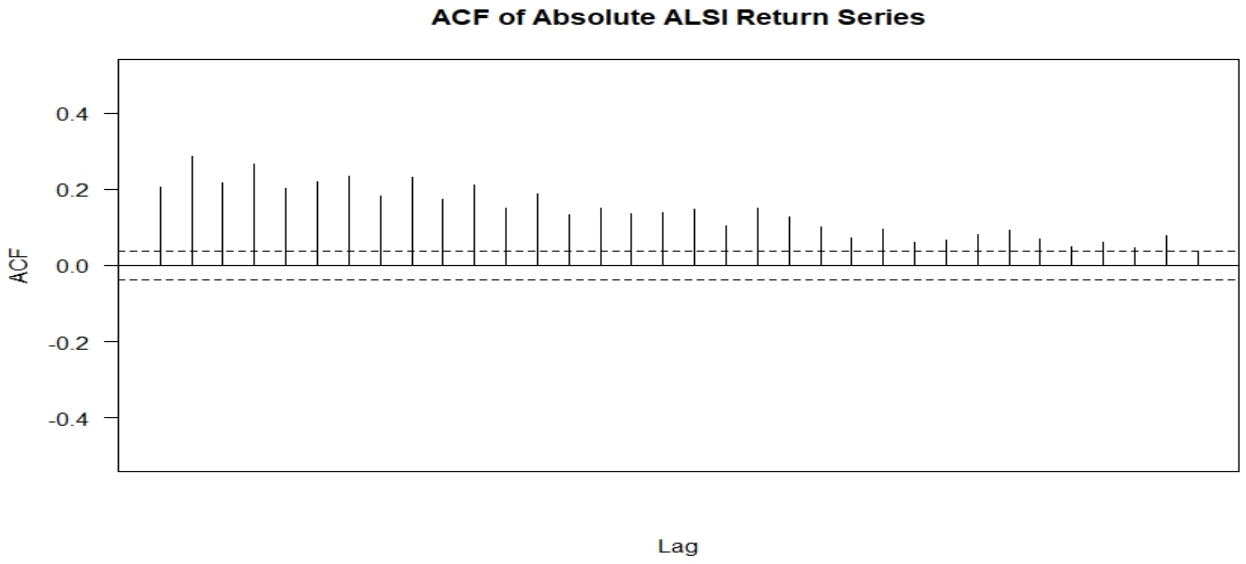

Figure 4.7: ACF of absolute ALSI returns with 95% confidence level

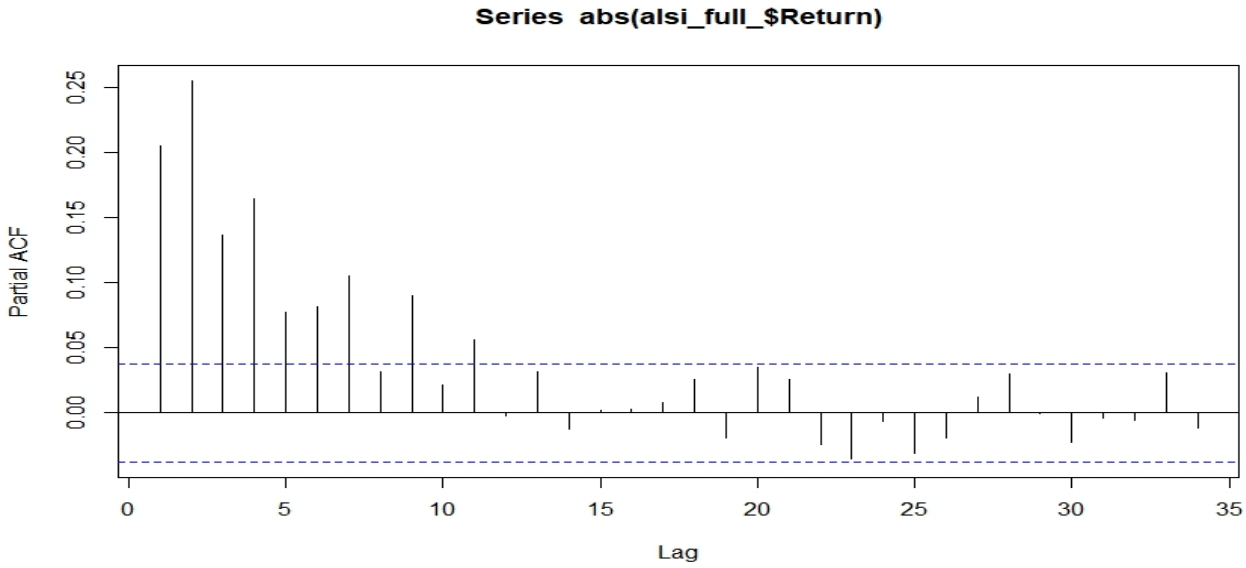

Figure 4.8: PACF of absolute ALSI returns with 95% confidence level

To further investigate the time series for long memory properties, the sample autocorrelations of the power transformed absolute series, $|R_t|^a$, are calculated for positive values of a. Table 4.3 shows the autocorrelations for a = 0.125, 0.25, 0.50, 0.75, 1, 1.25, 1.5, 1.75, 2 and 3 at lags 1 – 5 and 10, 20,40, 70, 100. All the power transformations except for a = 2 and a = 3 have significant positive autocorrelations up to a lag of at least 100. The autocorrelation decreases once values of a larger than 1 is used. This shows that long memory is observed in the returns of the ALSI.

Table 4.3: Autocorrelations of $|R_t|^a$

	Lag									
a	**1**	**2**	**3**	**4**	**5**	**10**	**20**	**40**	**70**	**100**
0.125	0.984	0.982	0.982	0.982	0.982	0.979	0.975	0.965	0.953	0.942
0.250	0.946	0.944	0.944	0.945	0.943	0.939	0.936	0.921	0.905	0.894
0.500	0.840	0.840	0.838	0.843	0.835	0.829	0.824	0.796	0.773	0.761
0.750	0.724	0.736	0.723	0.735	0.718	0.709	0.700	0.653	0.622	0.608
1.000	0.612	0.651	0.617	0.640	0.609	0.593	0.577	0.507	0.471	0.455
1.250	0.511	0.593	0.527	0.566	0.514	0.488	0.463	0.368	0.331	0.316
1.500	0.422	0.565	0.453	0.514	0.435	0.395	0.360	0.247	0.214	0.202
1.750	0.346	0.559	0.396	0.482	0.371	0.316	0.273	0.154	0.121	0.119
2.000	0.281	0.565	0.350	0.460	0.319	0.251	0.203	0.090	0.068	0.065
3.000	0.119	0.568	0.224	0.392	0.183	0.100	0.059	0.008	0.005	0.004

The mean-reversion behaviour of the ALSI returns series and the persistence in volatility presented in Tables 4.2 and 4.3 and Figures 4.7 and 4.8 lead to the conclusion that the return series is not a realisation of an IID process.

4.2.3 Stationarity Analysis

Prior to fitting a volatility model on the ALSI training dataset, the data are investigated for seasonal trends and non-stationarity. STL decomposition was performed to split the time series into three components: seasonality, trend and random fluctuation.

Figure 4.9 shows the STL decomposition of the ALSI log returns. The *stl()* function in the *stats* package was used (Komsta & Novometsky, 2022). The input data are decomposed as the sum of the trend, seasonal and residual/random fluctuation components.

The grey bars on the side of each graph show the relative contribution of each component. It should be noted that since each plot is on a different scale, the associated bar varies in size. The variation in the trend and seasonality components is the smallest compared to the variation in the remainder and it can be concluded that these components contribute the least to the variation observed in the data. As a result, no seasonal or trend differencing is needed to obtain second-order stationarity.

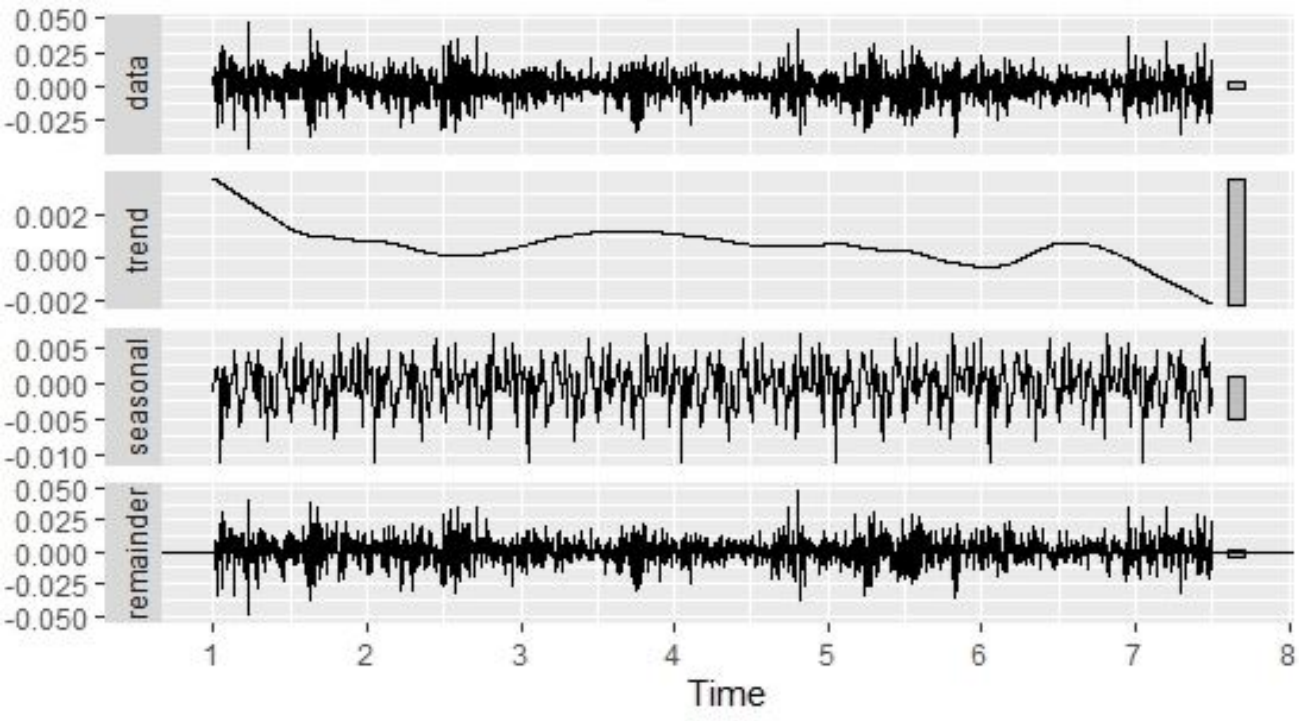

Figure 4.9: STL Decomposition of returns of ALSI

In addition, the *nsdiff()* function in the *forecast* package was used to determine whether seasonal differencing was required (Hyndman et al., 2022). The output indicated that the data are non-seasonal and that no seasonal difference is required.

To determine unit root (second-order) stationarity, the ADF test was performed using the *adf.test()* function in the *tseries* package (Trapletti et al., 2023). The test statistic was -14.111, with a p-value less than the default p-value of 0.05. The null hypothesis of the presence of a unit root (thus that the data are non-stationary) is rejected. As a result, the data are assumed to be stationary. No differencing is therefore needed to achieve second-order stationarity.

4.2.4 Specification of the Mean Equation

The first part of the ARCH modelling process is to specify an appropriate mean equation that explains the serial dependence in the first moment of the process. The ACF and PACF are used to identify the appropriate mean equation. The ACF plotted in Figure 4.5 and the PACF plotted in Figure 4.6 indicate that AR(3), MA(3) and ARMA(3,3) models should be tested using the *forecast* package (Hyndman et al., 2022). Higher order ARIMA-class models (models with long lags) often overfit the data and are not considered in this research report.

Table 4.4 shows the estimated coefficients for each model, as well as the AIC for each of the three models. The AICs are similar with a minimum of -15 240.33 for the ARMA(3,3) model.

The LB test is used to determine serial autocorrelation in the residuals. The LB test p-value of the residual series is 0.9816 and is much higher than the 0.05 threshold, thus implying that H_0 is not rejected. The residuals are therefore independently distributed at a 95% level of significance. The ARMA(3,3) model provides a suitable fit for the mean equation.

Table 4.4: Comparison of the coefficients of the mean models for ALSI time series

Model	ϕ_1	ϕ_2	ϕ_3	θ_1	θ_2	θ_3	**AIC**
AR(3)	−0.018	−0.06	0.01				−15 225.70
MA(3)				−0.02	−0.06	0.001	−15 226.16
ARMA(3,3)	−0.08	−0.09	0.89	0.06	0.05	−0.93	−15 240.33

After fitting an ARMA(3,3) model to the ALSI returns series, the ACF and PACF of the residuals are plotted in Figures 4.10 and 4.11. No significant correlations are observed.

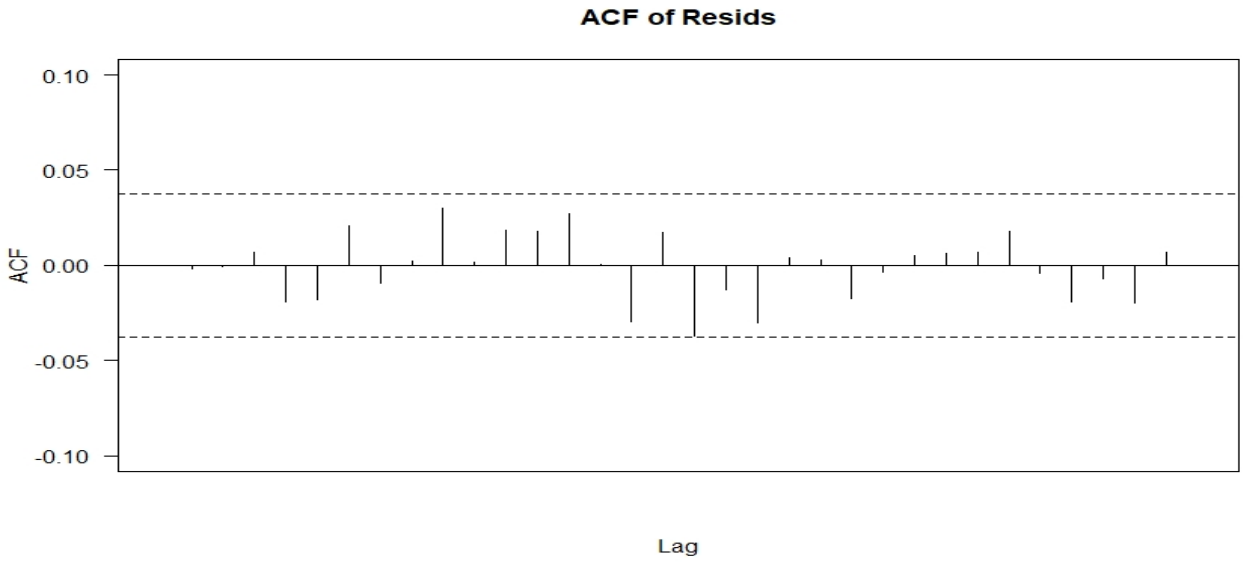

Figure 4.10: ACF of residuals for the ARMA(3,3) model with 95% confidence level

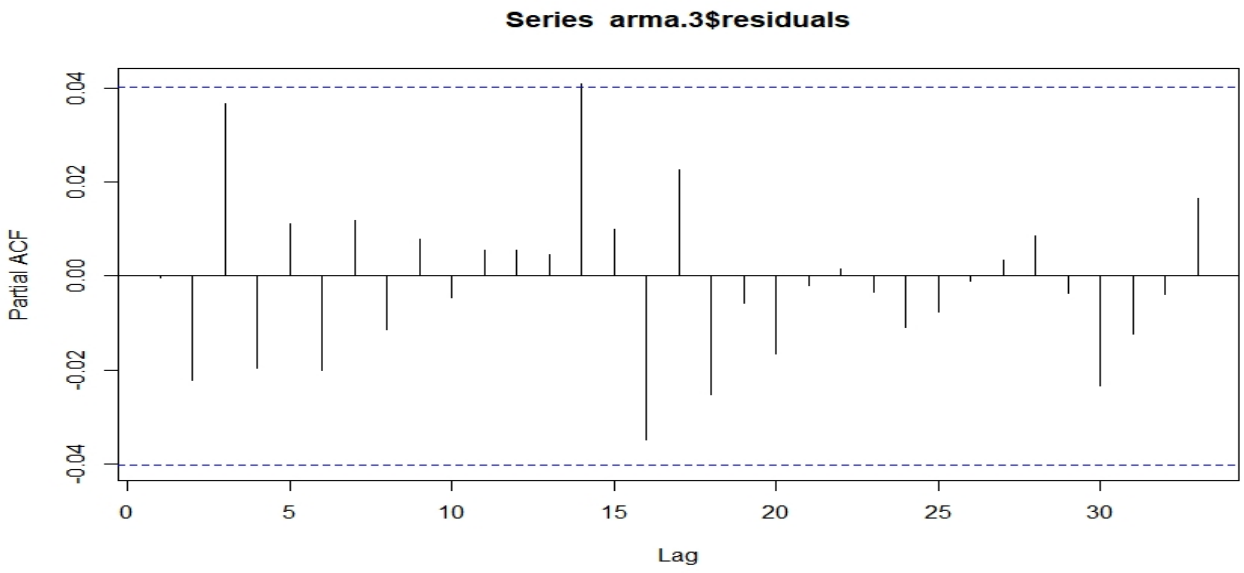

Figure 4.11: PACF of residuals for the ARMA(3,3) model with 95% confidence level

4.2.5 Testing for ARCH Effects

The LM test was conducted using the *Lm.test()* function in the *nortsTest* package (Matamoros et al., 2021). The test statistic is 2 540.10 with a p-value < 0.05. The null hypothesis is therefore rejected and it can be assumed that the residuals are heteroskedastic.

The ARCH effects can also be investigated by plotting the ACF and PACF of the squared residuals. ARCH models are used to describe the second moment of the error term, i.e. the variance. The residuals obtained from the ARMA model are therefore squared to determine whether there is variance present that can be explained by an ARCH-type model. Persistent significant correlation is observed in the ACF and PACF, shown in Figures 4.12 and 4.13, indicating heteroskedasticity in the residuals.

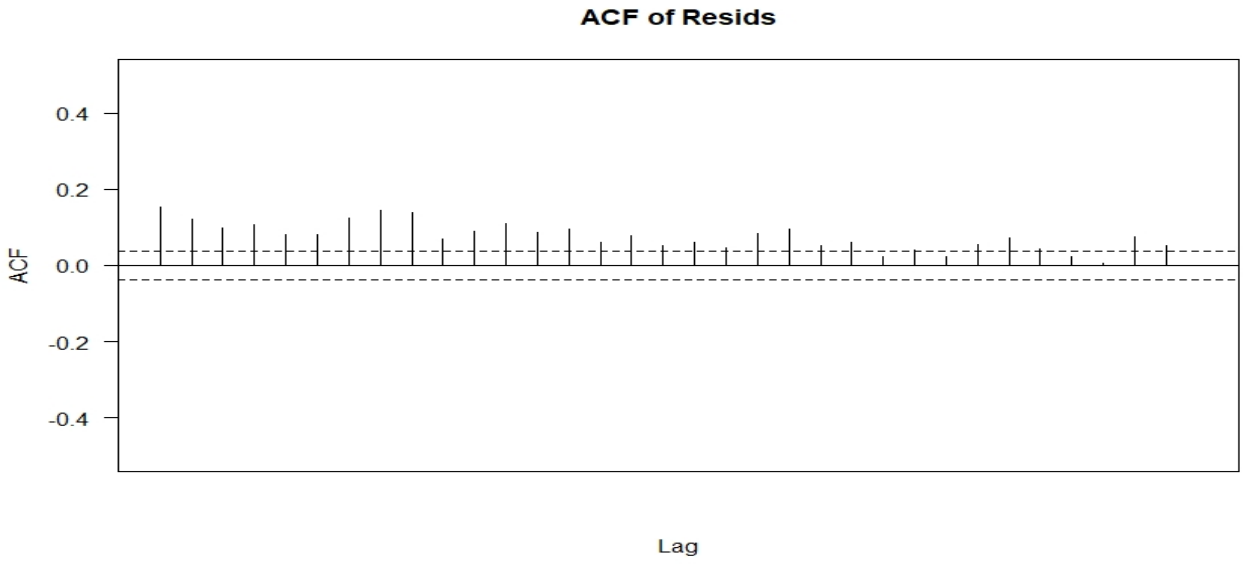

Figure 4.12: ACF of squared residuals for the ARMA(3,3) model at a 95% confidence level

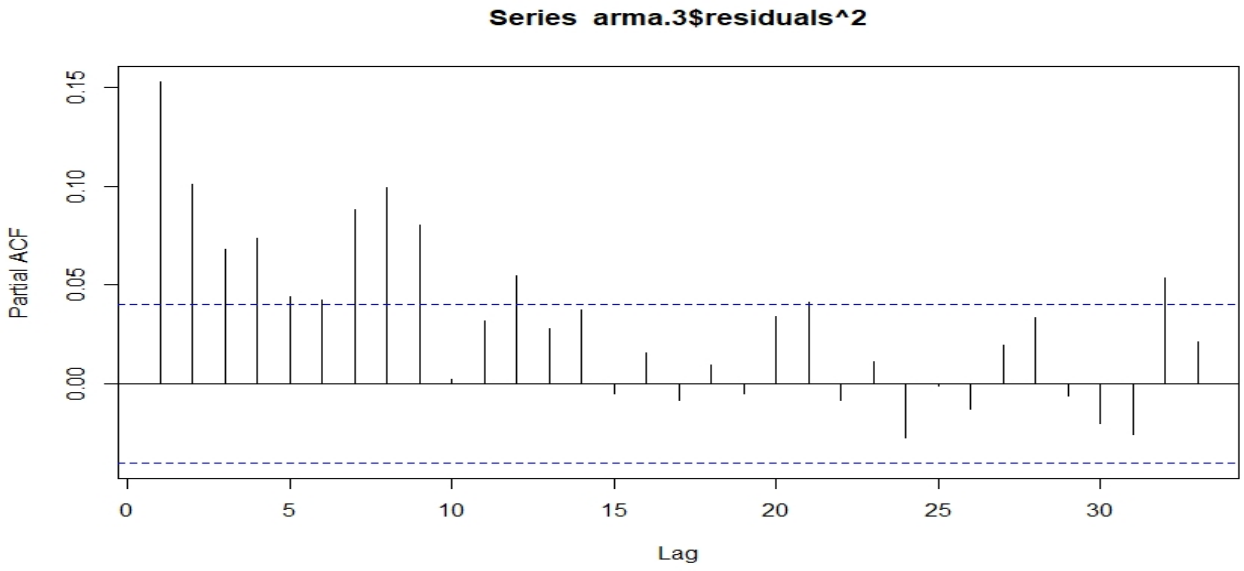

Figure 4.13: PACF of squared residuals for the ARMA(3,3) model at a 95% confidence level

4.3 Results

The ARMA(3,3) conditional mean model is used to determine the conditional variance model. As a part of the statistical learning component of this research, ARCH, GARCH, APARCH and EGARCH models are fitted where appropriate. ARCH models are used when an AR model describes the mean model, while GARCH models are used when an ARMA model describes the mean model. The ACF and PACF of the squared residuals presented in Figures 4.12 and 4.13 show a tapering trend, thus suggesting that a GARCH(1,1) model may be appropriate. GARCH(1,1) models are often used in

financial series such as stock price returns and inflation. The summary statistics presented in Section 4.2.1 showed that the ALSI return series is not normally distributed. An asymmetric model may therefore be appropriate. The GARCH(1,1) model will therefore be used as a baseline to show whether the EGARCH(1,1) and APARCH(1,1) models provide improved predictive power over the GARCH(1,1) model. The APARCH(1,1) model is, however, not expected to perform as well as the GARCH(1,1) and EGARCH(1,1) model since the mean model was described by an ARMA-type model. All results are obtained using the *rugarch* package (Trapletti et al., 2023).

4.3.1 ARMA(3,3)-GARCH(1,1) Model

Table 4.5 shows the parameter estimates and fit statistics when the model is fit to the training dataset. The AIC of the model is -6.52. With the exception of the ω parameter, the intercept term of the GARCH(1,1) process, all parameters are significant at a 95% confidence level.

The results in Table 4.6 provide the following conclusions:

- The Weighted LB (Standardised Residuals) tests show p-values exceeding 0.05. The null hypothesis cannot be rejected and it is deduced that there is no correlation present in the residuals. This holds over longer lags (lag 17 and lag 29) as well;
- The Weighted LB Test (Standardised Squared Residuals) tests result in p-values in excess of 0.05. The null hypothesis that the ARCH process is adequately fitted cannot be rejected; and
- The Sign-Bias tests show p-values below 0.05, thus leading to the conclusion that the leverage effect is present. The use of an EGARCH-type model would therefore be beneficial. It should be noted that the positive Sign-Bias is more significant than the negative Sign-Bias. This implies that positive changes in the ALSI return series lead to larger changes in volatility than negative changes.

Table 4.5: Parameter Estimates for the ARMA(3,3)-GARCH(1,1) Model

Parameter	Estimate	t-value	p-value
μ	0.001	4.900	<0.001
AR(1)	0.462	29.912	<0.001
AR(2)	−0.326	−6.053	<0.001
AR(3)	0.714	57.003	<0.001
MA(1)	−0.479	−67.429	<0.001
MA(2)	0.279	5.610	<0.001
MA(3)	−0.701	−116.322	<0.001
ω	0.000	1.349	0.117
α_1	0.079	4.553	<0.001
β_1	0.902	45.504	<0.001

Table 4.6: Diagnostic Test Results for the ARMA(3,3)-GARCH(1,1) Model

Test	Statistic	p-value
Weighted LB Test$_{StdRes;Lag1}$	0.943	0.331
Weighted LB Test$_{StdRes;Lag17}$	4.982	1.000
Weighted LB Test$_{StdRes;Lag29}$	9.181	0.999
Weighted LB Test$_{StdSqRes;Lag1}$	0.222	0.643
Weighted LB Test$_{StdSqRes;Lag5}$	0.802	0.908
Weighted LB Test$_{StdSqRes;Lag9}$	2.268	0.873
Weighted ARCH LM Test$_{Lag3}$	0.612	0.432
Weighted ARCH LM Test$_{Lag5}$	1.114	0.702
Weighted ARCH LM Test$_{Lag7}$	2.031	0.713
Sign-Bias	1.701	<0.001
Negative Sign-Bias	0.562	0.062
Positive Sign-Bias	2.035	<0.001
Joint Effect	26.501	<0.001

The conclusion reached from the results in Tables 4.5 and 4.6 is that the GARCH model provides a decent fit to the ALSI Return series. The ARCH effects identified in Section 4.3.2 are adequately described by the GARCH process, resulting in uncorrelated residuals. However, the leverage effect is not accurately captured by the GARCH model. This is expected as the GARCH model assumes

normality while the leverage effect is associated with non-normality.

The next step is to evaluate the model using the test series. The *rugarch* package has two forecasting options. The first is the bootstrapping method, which is based on the resampling of standardised residuals from the empirical distribution of the fitted model. The second is the rolling density forecast method (Ghalanos, 2022). For the purposes of this research report, the bootstrap method is used. The *forecast* package is used to obtain forecast accuracy measures, one of which is the RMSE. The RMSE for the forecasted model is 9.385×10^{-3}.

4.3.2 ARMA(3,3)-EGARCH(1,1) Model

Table 4.7 shows the parameter estimates and fit statistics of the ARMA(3,3)-EGARCH(1,1) model. The AIC of the model is -6.57, which is a small improvement on the ARMA(3,3)-GARCH(1,1) model. With the exception of the mean of the ARMA(3,3) model, μ, all parameters are significant at a 95% confidence level.

Table 4.7: Parameter Estimates for the ARMA(3,3)-EGARCH(1,1) Model

Parameter	**Estimate**	**t-value**	**p-value**
μ	0.000	1.439	0.150
AR(1)	−0.050	−6.901	<0.001
AR(2)	−0.029	−4.515	<0.001
AR(3)	0.938	146.171	<0.001
MA(1)	0.058	9.061	<0.001
MA(2)	0.027	6.680	<0.001
MA(3)	−0.949	−10545.01	<0.001
ω	−0.211	−146.65	<0.001
α_1	−0.125	−10.195	<0.001
β_1	0.977	3894.10	<0.001
γ_1	0.099	12.6555	<0.001

The results in Table 4.8 provide the following insights:

- The Weighted LB Test (Standardised Residuals) tests show correlation present in the residuals from lag 17 onwards;
- The Weighted LB Test (Standardised Squared Residuals) tests result in p-values in excess of 0.05. The null hypothesis that the EGARCH process fits adequately cannot be rejected; and

- The Sign-Bias tests show p-values below 0.05, thus indicating the presence of the leverage effect. However, the tests for the positive and negative biases cannot be rejected and it can be assumed that the asymmetry has been accurately captured.

Table 4.8: Diagnostic Test Results for the ARMA(3,3)-EGARCH(1,1) Model

Test	**Statistic**	**p-value**
Weighted LB Test$_{StdRes;Lag1}$	0.000	0.972
Weighted LB Test$_{StdRes;Lag17}$	14.953	<0.001
Weighted LB Test$_{StdRes;Lag29}$	20.981	0.023
Weighted LB Test$_{StdSqRes;Lag1}$	2.873	0.092
Weighted LB Test$_{StdSqRes;Lag5}$	4.271	0.223
Weighted LB Test$_{StdSqRes;Lag9}$	6.234	0.276
Weighted ARCH LM Test$_{Lag3}$	1.352	0.252
Weighted ARCH LM Test$_{Lag5}$	1.460	0.609
Weighted ARCH LM Test$_{Lag7}$	2.732	0.574
Sign-Bias	2.133	0.033
Negative Sign-Bias	0.642	0.522
Positive Sign-Bias	0.119	0.911
Joint Effect	7.281	0.060

The conclusion reached from the results in Tables 4.7 and 4.8 is that the EGARCH model describes the ALSI Return series appropriately. The ARCH effects are adequately described by the EGARCH process, resulting in uncorrelated residuals. The RMSE for the forecasted model is 9.367×10^{-3}.

4.3.3 ARMA(3,3)-APARCH(1,1) Model

Table 4.9 shows the parameter estimates and fit statistics of the ARMA(3,3)-APARCH(1,1) model. The AIC of the model is -6.55. The estimates for four of the parameters are insignificant at a 95% confidence level.

The results in Table 4.10 can be interpreted as follows:

- The Weighted LB Test (Standardised Residuals) tests show p-values exceeding 0.05. The null hypothesis cannot be rejected and it can be assumed that the residuals are uncorrelated. This holds over longer lags (lag 17 and lag 29) as well;

- The Weighted LB Test (Standardised Squared Residuals) tests result in p-values in excess of 0.05. The null hypothesis that the APARCH process is adequately fitted cannot be rejected; and
- The Sign-Bias tests show p-values below 0.05, thus leading to the conclusion that the leverage effect is present. However, the tests for the positive and negative biases cannot be rejected and it can be concluded that the asymmetry is accurately captured.

Table 4.9: Parameter Estimates for the ARMA(3,3)-APARCH(1,1) Model

Parameter	**Estimate**	**t-value**	**p-value**
μ	0.000	1.968	0.049
AR(1)	−0.317	−1.693	0.090
AR(2)	0.619	27.765	<0.001
AR(3)	0.044	1.179	0.238
MA(1)	0.320	1.678	0.093
MA(2)	−0.679	−34.844	<0.001
MA(3)	−0.072	−1.009	0.313
ω	0.000	0.108	0.914
α_1	0.035	2.257	0.024
β_1	0.914	55.726	<0.001
η_1	0.527	3.579	0.003
λ	2.597	26.195	<0.001

The conclusion reached from the results in Tables 4.9 and 4.10 is that the APARCH model describes the ALSI Return series appropriately. The ARCH effects identified in Section 4.3.2 are adequately described by the APARCH process, resulting in uncorrelated residuals. However, the leverage effect is not accurately captured by the APARCH model. This is expected as the APARCH model assumes normality, thus not capturing the difference in volatility changes as a result of good or bad news. The RMSE for the forecasted model is 9.375×10^{-3}.

Table 4.10: Diagnostic Test Results for the ARMA(3,3)-APARCH(1,1) Model

Test	**Statistic**	**p-value**
Weighted LB Test$_{StdRes;Lag1}$	0.018	0.894
Weighted LB Test$_{StdRes;Lag17}$	6.827	1.000
Weighted LB Test$_{StdRes;Lag29}$	12.609	0.791
Weighted LB Test$_{StdSqRes;Lag1}$	0.108	0.743
Weighted LB Test$_{StdSqRes;Lag5}$	2.922	0.421
Weighted LB Test$_{StdSqRes;Lag9}$	4.886	0.446
Weighted ARCH LM Test$_{Lag3}$	2.922	0.087
Weighted ARCH LM Test$_{Lag5}$	3.244	0.256
Weighted ARCH LM Test$_{Lag7}$	4.508	0.279
Sign-Bias	2.277	0.022
Negative Sign-Bias	0.749	0.453
Positive Sign-Bias	1.064	0.287
Joint Effect	15.544	0.001

4.3.4 LSTM Model

All results presented in this paper are obtained using the *keras* and *tensorflow* packages (Kalinowski et al., 2022) in R. The R code used to obtain the results was adapted from an online source (Wanjohi, 2018). The loss function and optimisation algorithm used are RMSE and ADAM respectively.

To determine the optimal LSTM model, the hyperparameters need to be optimised.

4.3.4.1 Effect of epochs on RMSE

The number of times a dataset is used to obtain the optimal weights is referred to as an epoch. One epoch, therefore, implies that an LSTM network was trained only once using the entire dataset. By using more than one epoch, the network often identifies trends in the data more accurately. This, however, depends on the underlying data and as a result, the optimal number of epochs required may differ between datasets. Larger values for the number of epochs also correspond to longer training run times. While faster computing systems may reduce run times, the trade-off between the change in RMSE and the run time to achieve the minimum RSME must be considered.

The LSTM model considered in this research report was trained assuming a range of values for the epoch hyperparameter. Values of 1, 10, 20, 50, 100, 200 and 500 were tested and the RMSE and run time were calculated after each iteration. The results are summarised in Table 4.11. From the results

obtained, changing the number of epochs does not result in a significant change in RMSE. In some cases, the RMSE increases, which indicates that the trained models are being overfitted. The increase in the number of epochs increases the run time of the LSTM model. The final value of 50 is chosen for the optimal number of epochs, as this is where the RMSE is minimised. With this value, further training is not needed to obtain a better performing model.

Table 4.11: Comparison of the RMSE for different values of epoch

epoch	RMSE	Run Time (Seconds)
1	0.2079	5
20	0.2084	100
50	0.2077	210
100	0.2084	450
200	0.2079	950
500	0.2082	2200

4.3.4.2 Effect of Learning Rate on the RMSE

Table 4.12 summarises the values for the RMSE for selected values of the learning rate, while keeping the value for the epochs hyperparmeter at 50. Values of 0.2, 0.25, 0.35, 0.45, 0.55 and 0.65 were tested and the RMSE and run time were calculated after each iteration. The minimum RMSE (0.2077) was obtained corresponding to a learning rate of 0.35. An increase in learning rate results in an increase in the RMSE for learning rates above 0.35.

Table 4.12: Comparion of the RMSE at different values for learning rate

Learning Rate	RMSE
0.2	0.2080
0.25	0.2079
0.35	0.2077
0.45	0.2083
0.55	0.2098
0.65	0.2106

4.3.4.3 Optimal Model Results

The optimal model is summarised in Table 4.13. The RMSE of the forecasted model is 0.208, which is the highest of the models presented in this research report. The model fails to provide improvement in capturing the long term memory properties of the time series. It is therefore outperformed by the ARMA(3,3)-EGARCH(1,1) model.

Table 4.13: Test Results for the LSTM ANN

Hyperparameter	**Value**
Batch Size	1
Activation	Sigmoid
Hidden Layers	1
Epochs	50
Learning Rate	0.35
Decay Rate	1×10^{-6}

The training dataset consists of 2 099 observations. Financial time series are often characterised by multiple regime switches over time. For the ALSI data, the Financial and COVID crises each correspond to a different regime. The small sample size available between these regime switches may not be large enough to sufficiently train the LSTM and capture long-memory trends.

From the results presented in Sections 4.3.1 – 4.3.4, the ARMA(3,3)-EGARCH(1,1) outperforms all the other models considered. This is discussed in more detail in Sections 5.3 and 5.4.

Chapter 5

Summary and Conclusion

5.1 Introduction

The purpose of this chapter is to provide the reader with a summary of key points and results provided in previous chapters. Recommendations and limitations to this study are also outlined.

5.2 Summary

The Black-Scholes model is used to derive the fair price of an option. One of the underlying assumptions is that of constant volatility, which does not hold in the markets. The leverage effect results in lower volatilities associated with increases in stock prices, while decreases in stock prices of the same magnitude result in higher volatilities. This results in stock price returns exhibiting asymmetrical behaviour. In addition, clustering of volatilities over longer periods results in the long memory property. Volatility estimates are therefore affected by large changes in past volatilities for longer than without the long memory property. To address this, volatility is often modelled using various methods. This study focused on the GARCH, EGARCH, APARCH and LSTM models to predict the volatility underlying the ALSI return series.

Another important assumption of the Black-Scholes model is that asset returns are normally distributed. In order to determine whether this holds for the ALSI return series, summary statistics and hypothesis tests were used. The following aspects of the ALSI return series were captured:

- Heavy-tailed behaviour, indicated by the kurtosis of 10.33;
- Asymmetrical returns and therefore the presence of the leverage effect, as indicated by the skewness of -0.53; and
- Non-normality in the return series, as determined by the rejection of the normality assumption under the JB hypothesis test.

The existence of long memory in the ALSI return series was tested by investigating the power of the transformed absolute series $|R_t|^d$, for positive values of d. The resulting significance in autocorrelation across long lags indicates the presence of long memory. Using an ADF test and STL decomposition, it was determined that the time series is stationary and that there are no seasonal components. The presence of long memory, asymmetry and non-normality in the ALSI return series proves that the assumptions of the Black-Scholes model do not hold. This aligns with the first objective of this research report.

The serial dependence in the first moment of the process is described with the use of an ARMA-type process. Upon investigation of the ACF and PACF of the training dataset, potential processes identified were AR(3), MA(3) and ARMA(3,3). Each of these models was fitted to the ALSI return series. The minimum AIC was obtained for the ARMA(3,3) process at -15 240.33. An LB test was then used to determine that the residuals are independently distributed and that the use of the ARMA(3,3) model is appropriate.

The presence of ARCH effects was determined using the ACF and PACF of the squared residuals in conjunction with an LM test. It was concluded that there are ARCH effects present in the residuals and the next steps were to fit GARCH-type models to the time series. The ARMA(3,3)-GARCH(1,1), ARMA(3,3)-EGARCH(1,1) and the ARMA(3,3)-APARCH(1,1) models were then fit. All models are fit using a conditional normal distribution.

The ARMA(3,3)-GARCH(1,1) model shows significance in all variables but one at a 95% level. The weighted LB Test on the standardised and squared residuals show that there is no correlation present and that the ARCH process fits adequately. The sign-bias test shows that the leverage effect is present, but not accurately captured by the GARCH model. When applying the model to the test series, the RMSE of the forecasted series using bootstrapping is 9.385×10^{-3}.

The ARMA(3,3)-EGARCH(1,1) model shows significance in all variables but one at a 95% level. The weighted LB Test on the standardised and squared residuals show that there are correlations present at longer lags, but that the EGARCH process fits adequately. The sign-bias test shows the presence of the leverage effect and that the model accounts for it. When applying the model to the test series, the RMSE of the forecasted series using bootstrapping is 9.367×10^{-3}.

The ARMA(3,3)-APARCH(1,1) model resulted in a poorer fit than the ARMA(3,3)-GARCH(1,1) and ARMA(3,3)-EGARCH(1,1) models. Four variables are insignificant at a 95% level. The weighted LB Test on the standardised and squared residuals show that there is no correlation present at longer lags and that the APARCH process fits adequately. The sign-bias test shows the presence of the leverage effect, but that it is accounted for by the model. When applying the model to the test series, the RMSE of the forecasted series using bootstrapping is 9.375×10^{-3}.

The results of the ARMA(3,3)-GARCH(1,1), ARMA(3,3)-EGARCH(1,1) and ARMA(3,3)-APARCH(1,1) models were compared. Prediction accuracy was compared through the use of the RMSE statistic. The ARMA(3,3)-EGARCH(1,1) had the lowest RMSE and thus the highest prediction accuracy. Through the comparison of these models, the second objective of this research report was achieved.

The selected LSTM model did not provide an improvement in predictive power. Its RMSE was 0.2077, which was compared to the RMSE of 9.375×10^{-3} of the ARMA(3,3)-EGARCH(1,1) model. This achieves the third objective of this research report.

Certain hyperparameters of the LSTM network were selected based on their suitability for modelling financial time series. This included the batch size, activation function, weight decay and the number of hidden layers. The effect of changes in the learning rate and number of epochs on the LSTM's performance was tested. It was found that the neural network was insensitive to changes in these hyperparameters. The sensitivity tests performed on the specified hyperparameters, therefore, concluded the final objective of this research report.

5.3 Limitations

The sample size used as a training dataset is small. This was done to remove the effects of regime switches which often occur in financial time series. This effect is exacerbated by the small size of the South African financial system in comparison to its international peers. It is therefore sensitive to changes in the international markets, which result in more frequent regime switches. The models do not account for the effects of external factors on the volatility of the JSE. In addition, these models may not account for complex relationships.

A few companies on the JSE contribute to most of the market capitalisation. As a result, a change in the stock price of one of these firms leads to stock market volatility. The statistical and machine learning methods presented in this research paper do not account for this issue as it considers the market as a whole.

5.4 Recommendation

When high-frequency intraday trading data are unavailable, the maximum and minimum prices of a particular trading day can be used as a proxy. The use of intraday data will, however, only provide more information if the stock market shows higher levels of volatility. This is typically the case with larger stock exchanges.

ANNs require large datasets to be trained adequately. Smaller datasets may therefore not provide enough observations to train an LSTM model sufficiently. Although data outside of the period in this research report consist of regime switches, the inclusion of this data may result in an ANN that outperforms the traditional models.

The models presented in this research paper consider only univariate relationships. It may be useful to consider multivariate extensions of the models.

The LSTM model may potentially be improved in the following ways:

- A comparison of RMSE for different activation functions may be useful to improve accuracy. In addition, different activation functions may be used for the input, hidden and output layers;
- Different batch sizes could lead to an improvement in the prediction accuracy of the LTSM;
- The LSTM model presented in this research report used only one hidden layer. Multiple layers may result in different results;
- The hyperparameters used in the LSTM network can be optimised with the use of maximum likelihood estimators or Bayesian optimisation techniques.

The above-suggested changes will require changes to the other hyperparameters as the entire model structure will change. As a result, the model may be overly complex without resulting in sufficient improvement in predictive power.

5.5 Conclusion

The results presented in this research report compared ARMA(3,3)-GARCH(1,1), ARMA(3,3)-EGARCH(1,1), ARMA(3,3)-APARCH(1,1) and LSTM models in terms of prediction accuracy. Although the RMSE was minimised for the ARMA(3,3)-EGARCH(1,1) model, the RMSEs for the three models were very similar. The ARMA(3,3)-EGARCH(1,1) model presented the most significant statistical test results. The LSTM model performed the worst in terms of RMSE. This is potentially attributed to the dataset size and architecture of the neural network. The latter was selected to ensure suitability for financial time series modelling. It can be concluded that for the ALSI returns data for the period 4 June 2009 – 29 November 2019, the ARMA(3,3)-EGARCH(1,1) model is best suited to predict volatility. An advantage to this is that statistical learning models are often easier to explain than neural networks.

It should be noted that the ARMA(3,3)-EGARCH(1,1) model provided sufficient predictive power, as well as a low RMSE. The use of more complicated methods to estimate implied volatility may not yield a sufficient improvement in predictive power to justify the complexity.

References

Akaike, H. (1998). Information Theory and an Extension of The Maximum Likelihood Principle. *Selected Papers of Hirotugu Akaike* (pp. 199–213). Springer.

Anderson, T., & Darling, D. (1954). A Test of Goodness of Fit. *Journal of the American Statistical Association*, *49*(268), 765–769.

Bain, A. (1873). *Mind and Body. The Theories of Their Relation*. Appleton, New York.

Black, F., & Scholes, M. (1973). The Pricing of Options and Corporate Liabilities. *Journal of Political Economy*, *81*(3), 637–654.

Bollerslev, T. (1986). Generalized Autoregressive Conditional Heteroskedasticity. *Journal of Econometrics*, *31*(3), 307–327.

Box, G., & Pierce, D. (1970). Distribution of Residual Autocorrelations in Autoregressive-Integrated Moving Average Time Series Models. *Journal of the American Statistical Association*, *65*(332), 1509–1526.

Bucci, A. (2020). Realized Volatility Forecasting with Neural Networks. *Journal of Financial Econometrics*, *18*(3), 502–531.

Christie, A. (1982). The Stochastic Behavior of Common Stock Variances : Value, Leverage and Interest Rate Effects. *Journal of Financial Economics*, *10*(4), 407–432.

Devroye, L., & Wagner, T. (1979). Distribution-Free Performance Bounds For Potential Function Rules. *IEEE Transactions on Information Theory*, *25*(5), 601–604.

Dickey, D. A., & Fuller, W. A. (1979). Distribution of the Estimators for Autoregressive Time Series With a Unit Root. *Journal of the American Statistical Association*, *74*(366a), 427–431.

Ding, Z., Granger, C., & Engle, R. (1993). A Long Memory Property of Stock Market Returns and a New Model. *Journal of Empirical Finance*, *1*(1), 83–106.

Durbin, J. (1959). Efficient Estimation of Parameters in Moving-Average Models. *Biometrika*, *46*(3/4), 306–316.

Engle, R. (1982). Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation. *Econometrica: Journal of the Econometric Society*, *50*(4), 987–1007.

Engle, R., & Ng, V. (1993). Measuring and Testing the Impact of News on Volatility. *The Journal of Finance*, *48*(5), 1749–1778.

Fama, E. (1965). The Behavior of Stock-Market Prices. *The Journal of Business*, *38*(1), 34–105.

Ferreira, N., Menezes, R., & Mendes, D. (2007). Asymmetric Conditional Volatility in International Stock Markets. *Physica A: Statistical Mechanics and its Applications*, *382*(1), 73–80.

Fisher, T., & Gallagher, C. (2012). New Weighted Portmanteau Statistics for Time Series Goodness of Fit Testing. *Journal of the American Statistical Association*, *107*(498), 777–787.

French, K., & Roll, R. (1986). Stock Return Variances : The Arrival of Information and the Reaction of Traders. *Journal of Financial Economics*, *17*(1), 5–26.

Geisser, S. (1975). The Predictive Sample Reuse Method With Applications. *Journal of the American Statistical Association*, *70*(350), 320–328.

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

Ghalanos, A. (2022). *Rugarch: Univariate GARCH Models* [R package version 1.4-7.].

Ghasemi, A., & Zahediasl, A. (2012). Normality Tests for Statistical Analysis: A Guide for Non-Statisticians. *International Journal of Endocrinology and Metabolism*, *10*(2), 486–489.

Glosten, L. R., Jagannathan, R., & Runkle, D. E. (1993). On The Relation Between The Expected Value and the Volatility of the Nominal Excess Return on Stocks. *The Journal of Finance*, *48*(5), 1779–1801.

Goodfellow, I., Bengio, Y., & Courville, A. (2017). Deep Learning (Adaptive Computation and Machine Learning Series). *Cambridge Massachusetts*, 321–359.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer New York Inc.

Hebb, D. (1949). *The Organization of Behavior*. Wiley, New York.

Hull, J. (2003). *Options, Futures and Other Derivatives*. Pearson Education, India.

Hyndman, R., Athanasopoulos, G., Bergmeir, C., Caceres, G., Chhay, L., O'Hara-Wild, M., Petropoulos, F., Razbash, S., Wang, E., & Yasmeen, F. (2022). *Forecast: Forecasting Functions for Time Series and Linear Models* [R package version 8.21].

James, W. (1890). The Principles of Psychology. *The American Journal of Psychology*, *103*(4), 433–447.

Jarque, C. M., & Bera, A. K. (1980). Efficient Tests for Normality, Homoscedasticity and Serial Independence of Regression Residuals. *Economics Letters*, *6*(3), 255–259.

Jelito, D., & Pitera, M. (2021). New Fat-tail Normality Test Based on Conditional Second Moments With Applications to Finance. *Statistical Papers*, *62*.

Kalinowski, T., Falbel, D., Allaire, J., Chollet, F., Tang, Y., Bijl, W. V. D., Studer, M., & Keydana, S. (2022). *R Interface to 'Keras'* [R package version 2.11.0].

Kim, S., Shephard, N., & Chib, S. (1998). Stochastic Volatility: Likelihood Inference and Comparison with ARCH Models. *The Review of Economic Studies*, *65*(3), 361–393.

Komsta, L., & Novometsky, F. (2022). *Moments, Cumulants, Skewness, Kurtosis and Related Tests* [R package version 0.14.1].

Krogh, A., & Hertz, J. (1991). A Simple Weight Decay Can Improve Generalization. *Advances in Neural Information Processing Systems*, *4*, 950–957.

Li, W., & Mak, T. (2008). On the squared residual autocorrelations in non-linear time series with conditional heteroscedasticity. *Journal of Time Series Analysis*, *15*, 627–636.

Matamoros, A. A., Nieto-Reyes, A., Hyndman, R., O'Hara-Wild, M., & [ctb], T. A. (2021). *Tests for Normality* [R package version 1.0.3].

McCulloch, W., & Pitts, W. (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, *5*(4), 115–133.

Miljanovic, M. (2012). Comparative Analysis of Recurrent and Finite Impulse Response Neural Networks in Time Series Prediction. *Indian Journal of Computer Science and Engineering*, *3*(1), 180–191.

Nelson, D. (1991). Conditional Heteroskedasticity in Asset Returns: A New Approach. *Econometrica*, *59*(2), 347–370.

Rosenblatt, F. (1958). The Perceptron: A Probabilistic Model For Information Storage And Organization In The Brain. *Psychological Review*, *65*(6), 386–408.

Royston, P. (1992). Approximating the Shapiro-Wilk W Test for Non-normality. *Statistics and Computing*, *2*, 117–119.

Royston, P. (1995). A Remark on Algorithm AS181: The W-Test for Normality. *Journal of the Royal Statistical Society*, *44*(4), 547–551.

Samuelson, P. A. (1973). Proof Properly Discounted Present Values of Assets Vibrate Randomly. *The Bell Journal of Economics and Management Science*, *4*(2), 369–374.

Schwert, G. (1989). Why Does Stock Market Volatility Change Over Time? *The Journal of Finance*, *44*(5), 1115–1153.

Shapiro, S., & Wilk, M. (1965). An Analysis of Variance Test for Normality (Complete Samples). *Biometrika*, *52*(3), 591–611.

Sharma, S., Sharma, S., & Athaiya, A. (2017). Activation Functions in Neural Networks. *Towards Data Science*, *6*(12), 310–316.

Sherrington, C. (1898). Experiments in Examination of the Peripheral Distribution of the Fibres of the Posterior Roots of Some Spinal Nerves. Part II. *Philosophical Transactions of the Royal Society of London. Series B, Containing Papers of a Biological Character*, *190*, 45–186.

Siami-Namini, S., Tavakoli, N., & Namin, A. (2018). A Comparison of ARIMA and LSTM in Forecasting Time Series. *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 1394–1401.

Snijders, T. A. (1988). On Cross-Validation for Predictor Evaluation in Time Series. *On Model Uncertainty and its Statistical Implications: Proceedings of a Workshop, Held in Groningen, The Netherlands, September 25–26, 1986*, 56–69.

Stoica, P., & Selen, Y. (2004). Model-Order Selection: A Review of Information Criterion Rules. *IEEE Signal Processing Magazine*, *21*(4), 36–47.

Thadewald, T., & Büning, H. (2007). Jarque–Bera Test and its Competitors for Testing Normality–a Power Comparison. *Journal of Applied Statistics*, *34*(1), 87–105.

Tong, H., & Lim, K. S. (2009). Threshold Autoregression, Limit Cycles and Cyclical Data. *Exploration Of A Nonlinear World: An Appreciation of Howell Tong's Contributions to Statistics* (pp. 9–56). World Scientific.

Trapletti, A., Hornik, K., & LeBaron, B. (2023). *Tseries: Time Series Analysis and Computational Finance* [R package version 0.10-43.].

Tsay, R. S. (1998). Testing And Modeling Multivariate Threshold Models. *Journal of the American Statistical Association*, *93*(443), 1188–1202.

Wanjohi, R. (2018). Time Series Forecasting Using LSTM in R [Online: Accessed June 2022, http://rwanjohi.rbind.io/2018/04/05/time-series-forecasting-using-lstm-in-r].

Wickham, H. (2023). *Welcome to the Tidyverse* [R package version 2.0.0].

Yadav, A., Jha, C., & Sharan, A. (2020). Optimizing LSTM for Time Series Prediction in Indian Stock Market. *Procedia Computer Science*, *167*, 2091–2100.

Appendix A

R Code

```
#Import data
> alsi_full <- read.csv("D://Users//Marik//Documents//MSc//Research Report
//Datasets//ALSINew.csv",sep = ";",header = TRUE)
> alsi_train <- read.csv("D://Users//Marik//Documents//MSc//Research Report
//Datasets//ALSINewTrain.csv",sep=";",header = TRUE)
> date_full <- as.Date(alsi_full$Date_str, "%d-%m-%Y")
> alsi_full_ <- cbind(alsi_full,date_full)
> date_train <- as.Date(alsi_train$Date_str, "%d-%m-%Y")
> alsi_train_ <- cbind(alsi_train,date_train)



#Graphs in section 4.2
> library(tidyverse)
> ts_data_price <- data.frame(alsi_full_$date_full,alsi_full_$Price,
alsi_full_$Return,alsi_full_$LogReturn)
> ggplot(alsi_full_,aes(sample=alsi_full_$Return))+stat_qq()+stat_qq_line()+
xlab("Sample Quantiles")+ ylab("Theoretical Quantiles")
> ggplot(ts_data_price, aes(x=alsi_full_.date_full, y=alsi_full_.Price))
+geom_line() +xlab("Date")+ ylab("Price")
> ggplot(ts_data_price, aes(x=alsi_full_.date_full, y=alsi_full_.LogReturn)) +
geom_line() + xlab("Date")+ ylab("Log Return")
> ggplot(ts_data_price, aes(x=alsi_full_.date_full, y=abs(alsi_full_.Return))) +
geom_line() + xlab("Date")+ ylab("Absolute Return")
```

```
> ggplot(ts_data_price, aes(x=alsi_full_.date_full, y=alsi_full_.Return)) +
geom_line() + xlab("Date")+ ylab("Return")


#Summary Statistics section 4.2
> library(moments)
> summary(alsi_full_$Return)
> summary(abs(alsi_full_$Return))
> summary(alsi_full_$LogReturn)
> c(jarque.test(alsi_full_$Return),jarque.test(alsi_full_$LogReturn),j
arque.test(abs(alsi_full_$Return)))
> c(skewness(alsi_full_$Return),skewness(alsi_full_$LogReturn),
skewness(abs(alsi_full_$Return)),kurtosis(alsi_full_$Return),
kurtosis(alsi_full_$LogReturn),kurtosis(abs(alsi_full_$Return)))
> c(sd(alsi_full_$Return),sd(alsi_full_$LogReturn),sd(abs(alsi_full_$Return)))

#ACFs and PACFs #2737 is the full dataset size
> library(stats)
> z<-acf(alsi_full_$Return)
> plot(z$acf[2:35],type = "h", main = "ACF of ALSI Return Series", xlab = "Lag",
ylab = "ACF", ylim = c(-0.1,0.1),las=1,xaxt="n")
> abline(h=0)
> abline(h=c(1.96/sqrt(2737),-1.96/sqrt(2737)),lty=c(2,2))
> x <- c(seq(1, 35, by=4))
> y <- c(seq(1, 35, by=4))
> axis(1, at=x, labels=y)
> pacf(alsi_full_$Return)
> x<-acf(abs(alsi_full_$Return))
> plot(x$acf[2:35],type = "h", main = "ACF of Absolute ALSI Return Series",
xlab = "Lag",ylab = "ACF", ylim = c(-0.5,0.5),
las=1,xaxt="n")
> abline(h=0)
> abline(h=c(1.96/sqrt(2737),-1.96/sqrt(2737)),lty=c(2,2))
> pacf(abs(alsi_full_$Return))
```

```
#Table 4.2
> print(acf(alsi_full_$Return,lag.max=100))
> print(acf(alsi_full_$LogReturn,lag.max=100))
> print(acf(abs(alsi_full_$Return),lag.max=100))

#Table 4.3
> print(acf(abs(alsi_full_$Return)^0.125,lag.max=100))
> print(acf(abs(alsi_full_$Return)^0.250,lag.max=100))
> print(acf(abs(alsi_full_$Return)^0.500,lag.max=100))
> print(acf(abs(alsi_full_$Return)^0.750,lag.max=100))
> print(acf(abs(alsi_full_$Return)^1,lag.max=100))
> print(acf(abs(alsi_full_$Return)^1.25,lag.max=100))
> print(acf(abs(alsi_full_$Return)^1.500,lag.max=100))
> print(acf(abs(alsi_full_$Return)^1.75,lag.max=100))
> print(acf(abs(alsi_full_$Return)^2,lag.max=100))
> print(acf(abs(alsi_full_$Return)^3,lag.max=100))

#STL Decomposition
> library(forecast)
> stl(ts(alsi_train$Return,freq=365.25), t.window=365.25, s.window="periodic",
robust=TRUE) %>%autoplot(main = "STL Decomposition of ALSI Training Set")
> nsdiffs(alsi_train_$Return)
> ndiffs(alsi_train_$Return)

#DF Test
> adf.test(alsi_train_$Return, alternative = "stationary")

#ARIMA Model Selection 2099 is the training dataset size
> ar.3 <- arima(alsi_train_$Return, order = c(3, 0, 0))
> ma.3 <- arima(alsi_train_$Return, order = c(0, 0, 3))
> arma.3 <- arima(alsi_train_$Return, order = c(3, 0, 3))
> y <- acf(arma.3$residuals)
> plot(y$acf[2:35],type = "h", main = "ACF of Resids", xlab = "Lag", ylab = "ACF",
```

```
ylim = c(-0.5,0.5),las=1,xaxt="n")
> abline(h=0)
> abline(h=c(1.96/sqrt(2099),-1.96/sqrt(2099)),lty=c(2,2))
> pacf(arma.3$residuals)
> Box.test(arma.3$residuals, lag = 1, type = "Ljung-Box", fitdf = 0)

#LM Test & Graphs
> Lm.test(arma.3$residuals)
> acf(arma.3$residuals^2)
> abline(h=0)
> abline(h=c(1.96/sqrt(2099),-1.96/sqrt(2099)),lty=c(2,2))
> pacf(arma.3$residuals^2)


#GARCH Models
> library(rugarch)
> mod1 <- ugarchspec(mean.model = list(armaOrder=c(3,3)),
variance.model = list(model = "fGARCH",garchOrder = c(1,1),submodel = "GARCH"),
distribution.model = "norm")
> modfit <- ugarchfit(data = alsi_train$Return, spec = mod1)
> bootp=ugarchboot(modfit,method=c("Partial","Full")[1],
n.ahead = 525,n.bootpred=1000,n.bootfit=1000)
> alsitest <- read.csv("D://Users//Marik//Documents//MSc//Research Report//
Datasets//ALSIValTest.csv",header = TRUE, sep = ",")
> s_f=bootp@forc@forecast$seriesFor #bootstrap forecasting discussed in Section 4.3.1
> s_f1 = as.vector(s_f)
> accuracy(s_f1,alsitest$Return)


#EGARCH Models
> mod.e <- ugarchspec(mean.model =
list(armaOrder=c(3,3)),variance.model = list(model = "eGARCH",garchOrder = c(1,1)),
distribution.model = "norm")
> modfit <- ugarchfit(data = alsi_train$Return, spec = mod.e)
```

```
> boote=ugarchboot(modfite,method=c("Partial","Full")[1],
n.ahead = 525,n.bootpred=1000,n.bootfit=1000)
> s_f_e=boote@forc@forecast$seriesFor #bootstrap forecasting discussed in Section 4.3.1
> s_f_e1 = as.vector(s_f_e)
> accuracy(s_f_e1,alsitest$Return)

#APARCH model
> mod.ap <- ugarchspec(mean.model = list(armaOrder=c(3,3)),variance.model =
list(model = "fGARCH",garchOrder = c(1,1),submodel = "APARCH"),
distribution.model = "norm")
> modfit <- ugarchfit(data = alsi_train$Return, spec = mod.ap)
> boota=ugarchboot(modfit,method=c("Partial","Full")[1],
n.ahead = 525,n.bootpred=1000,n.bootfit=1000)
> s_f_a=boota@forc@forecast$seriesFor #bootstrap forecasting discussed in Section 4.3.1
> s_f_a1 = as.vector(s_f_a)
> accuracy(s_f_a1,alsitest$Return)

#Applying the models to the test dataset
> alsi_test <- subset(alsi_full_, alsi_full_$Set=="Test")
> View(alsi_test)
> modfit_test <- ugarchfit(data = alsi_test$Return, spec = mod1)
> modfit_test
> modfit_test1 <- ugarchfit(data = alsi_test$Return, spec = mod.ap)
> modfit_test1
> modfit_test2 <- ugarchfit(data = alsi_test$Return, spec = mod.e)
> modfit_test2



#LTSM Model - Data Prep
> library(keras)
> library(tensorflow)
> l_trfm <- function(y, n= 1){
     l =  c(rep(NA, n), y[1:(length(y)-n)])
     DF = as.data.frame(cbind(l, y))
```

```
     colnames(DF) <- c( paste0('y-', n), 'y')
     DF[is.na(DF)] <- 0
     return(DF)
 }

> trainset = l_trfm(alsi_train_$Return, 1)
> alsi_test <- read.csv("D://Users//Marik//Documents//MSc//
Research Report//Datasets//ALSIValTest.csv",header = TRUE,sep=",")
> date_test <- as.Date(alsi_test$Date_str, "%d-%m-%Y")
> alsi_test_ <- cbind(alsi_test,date_test)
> testset = l_trfm(alsi_test_$Return, 1)

> scale_for_lstm = function(trainset, testset, feature_range = c(0, 1)) {
     x = trainset
     ftr_min = feature_range[1]
     ftr_max = feature_range[2]
     std_train = ((x - min(x) ) / (max(x) - min(x)  ))
     std_test  = ((testset - min(x) ) / (max(x) - min(x)  ))
     scaled_train = std_train *(ftr_max - ftr_min) + ftr_min
     scaled_test = std_test *(ftr_max - ftr_min) + ftr_min
     return( list(scaled_train = as.vector(scaled_train),
scaled_test = as.vector(scaled_test) ,scaler= c(min =min(x), max = max(x))) )
 }
> Scaled = scale_for_lstm(trainset, testset, c(-1, 1))
> dep_train = Scaled$scaled_train[, 2]
> indep_train = Scaled$scaled_train[, 1]
> dep_test = Scaled$scaled_test[, 2]
> indep_test = Scaled$scaled_test[, 1]

> dim(indep_train) <- c(length(indep_train), 1, 1)
> X_s2 = dim(indep_train)[2]
> X_s3 = dim(indep_train)[3]
> batchsize = 1
> units = 1
```

```
> model <- keras_model_sequential()
> model%>%layer_lstm(units, batch_input_shape = c(batchsize, X_s2, X_s3),
stateful= TRUE)%>%layer_dense(units = 1)
> model %>% compile(loss = 'mean_squared_error',optimizer =
optimizer_adam( learning_rate= 0.35, decay = 1e-6 ),metrics = c('accuracy'))
> summary(model)

> Ep = 50
> for(i in 1:Ep ){
     model %>% fit(indep_train, dep_train, epochs=1, batch_size=batchsize, verbose=1,
shuffle=FALSE)
     model %>% reset_states()
 }

> L = length(indep_test)
> scaler = Scaled$scaler
> predictions = numeric(L)
> for(i in 1:L){
     X = indep_test[i]
     dim(X) = c(1,1,1)
     y_pred = model %>% predict(X, batch_size=batchsize)
     # invert scaling
     y_predd = invert_scaling(y_pred, scaler,  c(-1, 1))
     # store
     predictions[i] <- y_predd}
> sqrt(mean((indep_test - predictions)^2))

> Ep = 1
> for(i in 1:Ep ){
     model %>% fit(indep_train, dep_train, epochs=1, batch_size=batchsize, verbose=1,
shuffle=FALSE)
     model %>% reset_states()}
```

```
> L = length(indep_test)
> scaler = Scaled$scaler
> predictions = numeric(L)
> for(i in 1:L){
     X = indep_test[i]
     dim(X) = c(1,1,1)
     y_pred = model %>% predict(X, batch_size=batchsize)
     # invert scaling
     y_predd = invert_scaling(y_pred, scaler,  c(-1, 1))
     # store
     predictions[i] <- y_predd}
> sqrt(mean((indep_test - predictions)^2))

> Ep = 20
> for(i in 1:Ep ){
     model %>% fit(indep_train, dep_train, epochs=1, batch_size=batchsize, verbose=1,
shuffle=FALSE)
     model %>% reset_states()
 }
 > for(i in 1:L){
     X = indep_test[i]
     dim(X) = c(1,1,1)
     y_pred = model %>% predict(X, batch_size=batchsize)
     # invert scaling
     y_predd = invert_scaling(y_pred, scaler,  c(-1, 1))
     # store
     predictions[i] <- y_predd
 }
> sqrt(mean((indep_test - predictions)^2))

> Ep = 100
> for(i in 1:Ep ){
     model %>% fit(indep_train, dep_train, epochs=1, batch_size=batchsize, verbose=1,
shuffle=FALSE)
```

```
     model %>% reset_states()
 }
 > for(i in 1:L){
     X = indep_test[i]
     dim(X) = c(1,1,1)
     y_pred = model %>% predict(X, batch_size=batchsize)
     # invert scaling
     y_predd = invert_scaling(y_pred, scaler,  c(-1, 1))
     # store
     predictions[i] <- y_predd
 }
> sqrt(mean((indep_test - predictions)^2))

> Ep = 200
> for(i in 1:Ep ){
     model %>% fit(indep_train, dep_train, epochs=1, batch_size=batchsize, verbose=1,
shuffle=FALSE)
     model %>% reset_states()
 }
 > for(i in 1:L){
     X = indep_test[i]
     dim(X) = c(1,1,1)
     y_pred = model %>% predict(X, batch_size=batchsize)
     # invert scaling
     y_predd = invert_scaling(y_pred, scaler,  c(-1, 1))
     # store
     predictions[i] <- y_predd
 }

> sqrt(mean((indep_test - predictions)^2))

> Ep = 500
> for(i in 1:Ep ){
     model %>% fit(indep_train, dep_train, epochs=1, batch_size=batchsize, verbose=1,
```

```
shuffle=FALSE)
     model %>% reset_states()
 }
 > for(i in 1:L){
     X = indep_test[i]
     dim(X) = c(1,1,1)
     y_pred = model %>% predict(X, batch_size=batchsize)
     # invert scaling
     y_predd = invert_scaling(y_pred, scaler,  c(-1, 1))
     # store
     predictions[i] <- y_predd
 }
> sqrt(mean((indep_test - predictions)^2))

> model <- keras_model_sequential()
> model%>%layer_lstm(units, batch_input_shape = c(batchsize, X_s2, X_s3),
stateful= TRUE)%>%layer_dense(units = 1)
> model %>% compile(loss = 'mean_squared_error',optimizer =
optimizer_adam( learning_rate= 0.2, decay = 1e-6 ),metrics = c('accuracy'))
> summary(model)

> Ep = 50
> for(i in 1:Ep ){
     model %>% fit(indep_train, dep_train, epochs=1, batch_size=batchsize, verbose=1,
shuffle=FALSE)
     model %>% reset_states()
 }

> L = length(indep_test)
> scaler = Scaled$scaler
> predictions = numeric(L)
> for(i in 1:L){
     X = indep_test[i]
     dim(X) = c(1,1,1)
```

```
     y_pred = model %>% predict(X, batch_size=batchsize)
     # invert scaling
     y_predd = invert_scaling(y_pred, scaler,  c(-1, 1))
     # store
     predictions[i] <- y_predd
 }
> sqrt(mean((indep_test - predictions)^2))

> model <- keras_model_sequential()
> model%>%layer_lstm(units, batch_input_shape = c(batchsize, X_s2, X_s3),
stateful= TRUE)%>%layer_dense(units = 1)
> model %>% compile(loss = 'mean_squared_error',optimizer =
optimizer_adam( learning_rate= 0.25, decay = 1e-6 ),metrics = c('accuracy'))
> summary(model)

> Ep = 50
> for(i in 1:Ep ){
     model %>% fit(indep_train, dep_train, epochs=1, batch_size=batchsize, verbose=1,
shuffle=FALSE)
     model %>% reset_states()
 }

> L = length(indep_test)
> scaler = Scaled$scaler
> predictions = numeric(L)
> for(i in 1:L){
     X = indep_test[i]
     dim(X) = c(1,1,1)
     y_pred = model %>% predict(X, batch_size=batchsize)
     # invert scaling
     y_predd = invert_scaling(y_pred, scaler,  c(-1, 1))
     # store
     predictions[i] <- y_predd
 }
```

```
> sqrt(mean((indep_test - predictions)^2))

> model <- keras_model_sequential()
> model%>%layer_lstm(units, batch_input_shape = c(batchsize, X_s2, X_s3),
stateful= TRUE)%>%layer_dense(units = 1)
> model %>% compile(loss = 'mean_squared_error',optimizer =
optimizer_adam( learning_rate= 0.35, decay = 1e-6 ),metrics = c('accuracy'))
> summary(model)

> Ep = 50
> for(i in 1:Ep ){
     model %>% fit(indep_train, dep_train, epochs=1, batch_size=batchsize, verbose=1,
shuffle=FALSE)
     model %>% reset_states()
 }

> L = length(indep_test)
> scaler = Scaled$scaler
> predictions = numeric(L)
> for(i in 1:L){
     X = indep_test[i]
     dim(X) = c(1,1,1)
     y_pred = model %>% predict(X, batch_size=batchsize)
     # invert scaling
     y_predd = invert_scaling(y_pred, scaler,  c(-1, 1))
     # store
     predictions[i] <- y_predd
 }
> sqrt(mean((indep_test - predictions)^2))

> model <- keras_model_sequential()
> model%>%layer_lstm(units, batch_input_shape = c(batchsize, X_s2, X_s3),
stateful= TRUE)%>%layer_dense(units = 1)
> model %>% compile(loss = 'mean_squared_error',optimizer =
```

```
optimizer_adam( learning_rate= 0.45, decay = 1e-6 ),metrics = c('accuracy'))
> summary(model)

> Ep = 50
> for(i in 1:Ep ){
     model %>% fit(indep_train, dep_train, epochs=1, batch_size=batchsize, verbose=1,
shuffle=FALSE)
     model %>% reset_states()
 }

> L = length(indep_test)
> scaler = Scaled$scaler
> predictions = numeric(L)
> for(i in 1:L){
     X = indep_test[i]
     dim(X) = c(1,1,1)
     y_pred = model %>% predict(X, batch_size=batchsize)
     # invert scaling
     y_predd = invert_scaling(y_pred, scaler,  c(-1, 1))
     # store
     predictions[i] <- y_predd
 }
> sqrt(mean((indep_test - predictions)^2))


> model <- keras_model_sequential()
> model%>%layer_lstm(units, batch_input_shape = c(batchsize, X_s2, X_s3),
stateful= TRUE)%>%layer_dense(units = 1)
> model %>% compile(loss = 'mean_squared_error',optimizer =
optimizer_adam( learning_rate= 0.55, decay = 1e-6 ),metrics = c('accuracy'))
> summary(model)

> Ep = 50
> for(i in 1:Ep ){
```

```
     model %>% fit(indep_train, dep_train, epochs=1, batch_size=batchsize, verbose=1,
shuffle=FALSE)
     model %>% reset_states()
 }

> L = length(indep_test)
> scaler = Scaled$scaler
> predictions = numeric(L)
> for(i in 1:L){
     X = indep_test[i]
     dim(X) = c(1,1,1)
     y_pred = model %>% predict(X, batch_size=batchsize)
     # invert scaling
     y_predd = invert_scaling(y_pred, scaler,  c(-1, 1))
     # store
     predictions[i] <- y_predd
 }
> sqrt(mean((indep_test - predictions)^2))


> model <- keras_model_sequential()
> model%>%layer_lstm(units, batch_input_shape = c(batchsize, X_s2, X_s3),
stateful= TRUE)%>%layer_dense(units = 1)
> model %>% compile(loss = 'mean_squared_error',optimizer =
optimizer_adam( learning_rate= 0.65, decay = 1e-6 ),metrics = c('accuracy'))
> summary(model)

> Ep = 50
> for(i in 1:Ep ){
     model %>% fit(indep_train, dep_train, epochs=1, batch_size=batchsize, verbose=1,
shuffle=FALSE)
     model %>% reset_states()
 }
```

```
> L = length(indep_test)
> scaler = Scaled$scaler
> predictions = numeric(L)
> for(i in 1:L){
     X = indep_test[i]
     dim(X) = c(1,1,1)
     y_pred = model %>% predict(X, batch_size=batchsize)
     # invert scaling
     y_predd = invert_scaling(y_pred, scaler,  c(-1, 1))
     # store
     predictions[i] <- y_predd
 }
> sqrt(mean((indep_test - predictions)^2))
```