Linköping University | Department of Computer and Information Science
Master's thesis, 30 ECTS | Division of Statistics and Machine Learning
2022 |LIU-IDA/STAT-A–22/022–SE

A study of forecasts in Financial Time Series using Machine Learning methods

Traditional vs Machine learning approach

Mowniesh Asokan

Supervisor : Sebastian Sakowski
Examiner : Josef Wilzén

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Upphovsrätt

Copyright

Abstract

Forecasting financial time series is one of the most challenging problems in economics and business. Markets are highly complex due to non-linear factors in data and uncertainty. It moves up and down without any pattern. Based on historical univariate close prices from the S&P 500, SSE, and FTSE 100 indexes, this thesis forecasts future values using two different approaches: one using a classical method, a Seasonal ARIMA model, and a hybrid ARIMA-GARCH model, while the other uses an LSTM neural network. Each method is used to perform at different forecast horizons. Experimental results have proven that the LSTM and Hybrid ARIMA-GARCH model performs better than the SARIMA model. To measure the model performance we used the Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE).

Keywords: machine learning, deep learning, Long Short-Term Memory (LSTM), Autoregressive Integrated Moving Average (ARIMA), time series, stock market, index, ADF, ACF, and PACF.

Acknowledgments

First and foremost, praises and thanks to the Almighty Creator, for his showers of blessings to complete this thesis successfully.I would wish to express my deep sense of gratitude and thanks to my supervisor, Sebastian Sakowski of the Division of Statistics and Machine Learning (STIMA), Department of Computer and knowledge Science (IDA), for his exceptional guidance, supervision, and encouragement during the thesis work. I'm grateful to my examiner Josef Wilzén of STIMA for his valuable comments and constructive feedback during this journey. I would also wish to thank my family for the unconditional love, endless support, and encouragement during the study period.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

Financial time series is one of the active research areas for economics and investment. The values of the stock market index refer to the price index or movement of fluctuation of the market. As financial time series data moves as a result of non-linear factors such as the economic rate, GDP growth of a country, and other investor sentiments, it becomes difficult to make accurate forecasts due to noise and chaos. The global stock indices of the different countries are may vary, based on significant world events such as Covid 19 pandemic rescission, and the war in Ukraine. Economic policies are developed and announced by the most influential countries in the world. These pieces of information affect the sudden change in the stock indices of all other countries. The fundamental and quantitative analysis of data such as stock price, volume, portfolio, etc. Stock-related information of the associated organization profile, strategies, [39] and derived correlation to predict the market behavior.

In practice, traders and investors anticipate different information for decision making such as corporate disclosures, stock price, macroeconomic data, news, economic fluctuation of dominant countries, and even social media. The social and economic conditions of every country were analyzed to extract the sentiments from the financial text. Those texts have a strong correlation with the stock market movement has shown in [24]. There is a good notable expansion being made in the field of text mining and natural language processing in recent years, there has also been combining textual analysis with machine learning techniques to forecast the stock values of the future[24] [21].

Multiple approaches and methods are used in economics and as well as in the computer science area to predict market behavior, including stock trend forecasting (up, down, bear market, or bull market respectively). This study aims to analyze and investigate appropriate forecasting methods that offer low forecasting error and high forecasting accuracy. Therefore, we are working on the univariate time series data i.e., stocks Close price is a covariate for each method. The results of some literature indicate that most of the data are non-stationary. The most well-known econometric and traditional method is the Autoregressive Integrated Moving Average method. This is the only statistical time series model which adds differencing ARMA process. We compare the traditional methods with the modern machine learning method like Recurrent neural network (RNN) and Long Short Term Memory (LSTM). In time series forecasting, deep learning techniques can identify structures and inherent patterns in data, such as non-linearity and complexity[28].

Stock shares are fractional ownership of a corporation, generally granting the holder the right to a portion of earnings, proceeds from liquidation events, and voting rights. Investors can buy and sell stock shares privately or through the stock exchange. As time goes on, the price of a stock may rise or fall. Generally, investors and traders buy stocks they believe will rise in price and sell stocks they believe will decrease in price. Returns of a stock can be expressed as the change in its price over time, and daily returns are the change in its price over a single day. It [38] explains how stocks and self-reported planning horizons impact individual investors' asset allocation decisions. I find that stocks of individual companies

and investment horizons play different roles in determining investment decisions in risky portfolios. This is explained by the relationship between investing time horizons and firm performance. It might tend to affect only investments in financial risky assets like stocks, options, and mutual funds. A longer horizon leads to an increasing share of risky financial investments. To make, less risk-averse investors and individuals on day trade more in stocks. We chose the different horizons as 5, 21, and 62. Finally, investors should utilize it when evaluating the risk-return relationship of forecasted values with the actual values.

This paper provides the in-depth orchestration of data preprocessing, feature engineering, and training of ML models with different financial time series data. The major work of this paper are :

- Investigate the performance of deep learning-based algorithms and traditional forecasting techniques through an empirical study.
- Compare the forecasting error between the LSTM, SARIMA, and Hybrid ARIMA-GARCH models.
- Compare the result of different forecast horizons between the three methods.

1.1 Aim & Objective

This thesis aims to study historical values of time series and build a model to describe the structure of the data and predict future values of the time series path. As time-series forecasting plays a vital role in many branches of applied sciences, it is imperative to develop an effective model to improve forecasting accuracy. To achieve this, multiple steps are derived as follows:

1. Select the best model with optimal parameters and specifications.
2. Choose the applicable metrics to evaluate the model performance.
3. Validate the approach using a suitable method.
4. Compare the performance of the classical approach with the machine learning approach.

1.2 Research question

1. Which of the methods selected for comparison, i.e. Seasonal-ARIMA, Hybrid ARIMA-GRACH, and LSTM, delivers the most accurate performance?
2. How do classical and machine learning approaches perform at different forecast horizons?

1.3 Literature Review

A variety of forecasting methods have been proposed in the literature, but in this study, only stock prices will be forecasted. [23] showed that the Autoregressive Moving Average (ARMA) was better at predicting the future values than the Kalman filter in the field of time series estimation in gas sensors. The authors built the ARMA and Kalman models on the short-term time-series data (collected for a shorter period, such as a month) and forecasted the long-term time series (for a longer period, such as a year) based on the trained models. Another appropriate method used for time series prediction problems is a recurrent neural network [41]. This is mainly due to their ability to take into account the sequential nature of time series explicitly and thus learn more efficiently. The most significant reason is classical methods are

not capturing non-linear patterns of the time series. LSTM is a good algorithm for time-series predictions due to its ability to consider long-term dependabilities and It has a strong ability to capture nonlinear patterns in time series data. Nelson et al.[8] built binary classification models using LSTM RNNs that, instead of predicting the future price of a stock, predicted whether a stock would be higher or lower than the current price 15 minutes in the future. These were trained on the trading data of stocks on the Brazilian stock exchange, along with a set of technical indicators derived from the trading data. On the five stocks for which the authors published the results, the lowest and highest accuracy of the models was 53.0 and 55.9 percent respectively.

The research on time series forecasting of the financial market originated from the efficient market hypothesis. In [34]the author Vantuch and Tomas states the theory, how the evolutionary based ARIMA model useful to analyze past events to produce valuable results for the next phase of the period. Machine Learning and Deep learning approaches are tailored to the prediction problem, where the association of variables is modeled in a deep and layered hierarchy. Prediction of the stock price using Recurrent Neural Network proposed by Zhu 2020 [41]. This paper shows that the experiment is made use of machine learning libraries like Keras and TensorFlow to train the stock trading data of Apple and predict the stock prices with good accuracy.In [26], COVID-19 time series data of America is used to forecast the future spread of diseases in the country by applying the both eXtreme Gradient Boosting and Long Short Term Memory Algorithm.

Since the 1990s, neural networks have been increasingly applied to finance in academic settings. According to Chen et al.,(2015)[7], LSTM networks can understand data structure dynamically over time with excellent prediction abilities. Using the same algorithm, Nelson et al.,(2017) performed a similar study on the Brazilian market and found the model could accurately predict price changes within the next few months with an accuracy of 55%. The results of this paper [35] show the experimental comparison of 10 stock datasets on the different parameters of each model, and the proposed model is generalized to improve the prediction result on a single ARIMA or a single XGBoost.A paper [31] compares the accuracy of ARIMA and LSTM and explains techniques when forecasting time series data. These techniques were executed on a set of financial data and the results showed that LSTM was far superior to ARIMA.To tackle these challenges, an alternate approach is a network autoregressive (NAR) model [40] with the de-GARCH technique, denoted by NAR-GARCH, which is proposed in this study. Specifically, the NAR-GARCH model first filters out the GARCH effects contained in each return process. Next, a NAR model is used to capture the joint effects in the de-GARCH processes, where a systematic scheme for accommodating the most updated market information is also proposed under the framework of the Granger causality test [13] and Pearson's correlation test with sharp price movements. In particular, XGboost and LSTM are used for handwritten digit recognition Graves et al. (2009) [14], speech recognition Robinson, 2002; Eyben 2009; Graves et al. 2013; Sak et al. 2014 [30], and text classification. Others proceed to forecast stock returns using a unique decision-making model for day trading investments on the stock market the model developed by the authors uses the support vector machine (SVM) method and the mean-variance (MV) method for portfolio selection [29]. Our primary objective is to forecast the closing prices for a portfolio of assets using statistical (ARIMA) and machine learning (ML) algorithms based on LSTM RNN and XGBoost. Predicting future portfolio values based on the most accurate algorithm is paramount to our success

Engle [12] first proposed this ARCH model to predict the conditional variance of return series. It has the key strength of producing volatility estimates with positive excess kurtosis (that is, fat tails are present about the normal distribution which is by empirical observations about returns). However, it also has some weaknesses. Firstly, due to the potentially large value of the lag q, it might be necessary to estimate a large number of parameters. Therefore, it may be difficult to estimate parameters [16]. Secondly, as we know in practice, stock prices

or financial assets, in general, react differently to positive and negative shocks. However, ARCH models assume these Kinds of shocks have the same effects on the volatility as they depend on the square of the previous shocks. It is important to take into account the leverage effect, which explains the difference in volatility's reaction between notable price rises and notable price falls. As a result of this, asymmetric GARCH models have been developed by adjusting the error term in the variance equation with a parameter to account for this effect. This was first proposed by Engle et al[11].

[3] Weiss proposed the class of ARIMA models with ARCH errors. They were applied to U.S. macroeconomic data. Many researchers later adopted and extended this method to model time series in various fields Jablecki et al. and Yaziz et al.[37] developed an ARIMA-GARCH model for forecasting gold price. Based on the empirical results of the 40-day gold price history, we can demonstrate that the hybrid ARIMA(1,1,1)-GARCH(0,2) model provides superior results and is more effective in evaluating and predicting gold prices than linear models.

Chapter 2
Data

2.1 Data Source

We collected 5 years of historical data for the stock index of 3 developing countries in the world. To collect these data, we made use of the python package which connects the Yahoo finance [36] and loads the data in the form of the Pandas data frame. The daily data includes the S&P 500, FTSE 100, and SHANGHAI Stock Exchange (SSE). This dataset was collected because the trading was continuous and there were no breaks in the market trading. Moreover, the data was collected from January 2018 to March 2022. The index stocks cannot be bought but it usually represents the overall stock movement of the country.

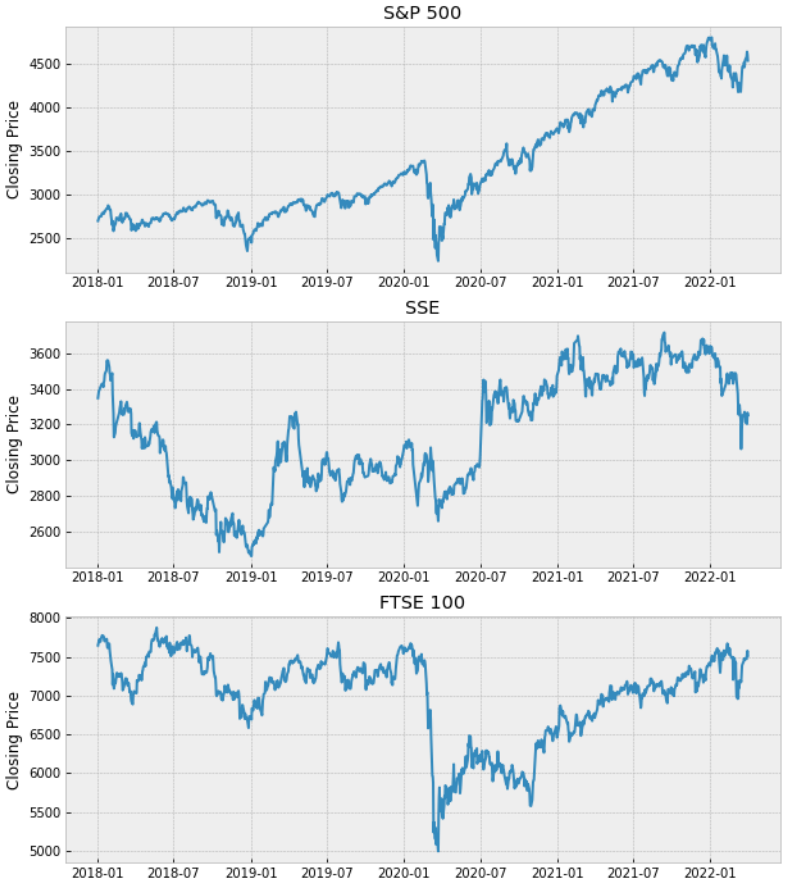


Figure 2.1: Data Overview of time series in the financial market

2.2 Data Description

Data covers price as a function of time from various financial markets. It consists of features like Open, Close, High, Low, and volume of trading. The close price is adjusted for stock splits and dividends. **figure** 2.1 displays the daily development in closing prices of the different stock indexes. The data was obtained from the different currency pairs USD, EURO, and CNY.

2.3 Data formatting

The close price is used in this thesis to determine the future value of the index. It is the price adjusted for stock splits and dividend distributions. The purpose of this measure is to ensure that such actions do not misleadingly affect the index value. The data formatting and overcome of analysis are explained below,

Normalization: In time series forecasting, the min-max normalization method presents a problem since the minimum and maximum values of the out-of-sample data are unknown. We can overcome this problem by considering the minimum (min_A) and maximum (max_A) values presented in the in-sample data set, and then mapping all out-of-sample values below min_A and above max to low and high, respectively. Due to this approach, substantial information is lost, and values are concentrated in a limited range, requiring more computation and decreasing the quality of methods to learn. This technique of normalization is used only for modern approaches like LSTM but in the case of the classical method(SARIMA and hybrid ARIMA), data is used with an original scale.

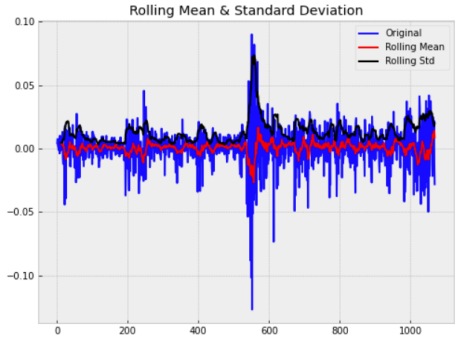
Data Cleansing: The cleansing of data is divided into two tasks. The first is dealing with missing data, while the second is aligning data. The first consideration is how to deal with missing data. The data used in this study are from different assets and different areas. In different time zones, there may be some missing data due to different holiday arrangements and different trading days. Since the paper used sufficient historical data, it is reasonable to exclude the data from certain dates that were unavailable. If the data is needed to fill on the missing place then mean imputation is used. The alignment of the data is another consideration. There are differences in the trading periods of individual assets and regions. Markets such as the S&P 500 trade in the same period, whereas markets in Asia such as the SSE trade before the markets in the US, though the dates are the same. So this paper uses the daily data of S&P 500 and SSE on the previous dates and data of FTSE 100 (Europe) and SSE (Asia) on the same date to forecast the moving direction of the S&P 500 (North America).To avoid the influence of different data scales, this paper scales feature values linearly in a range of [0,1]. In other words, features with a large numerical range will dominate those with a small numerical range.

Outcome of Analysis: From **figure** 2.2 one of the most obvious observations is that the data, there is a giant dip in the middle corresponding to the market crash in 2020 and 2021, is non-stationary. It makes sense for market data as it tends to spike up in the long run rather than fluctuate down. Time series analysis faces this problem because non-stationary data tends to be difficult to analyze. In the first instance, we can try a first difference between the series. The difference(t) is derived by subtracting the previous value t-1 from the current value t. The data no longer appears to be trending upward over time and is instead centered around 0. However, there is another problem. When we look at the variance, it shows some sudden shocks in the overall range.

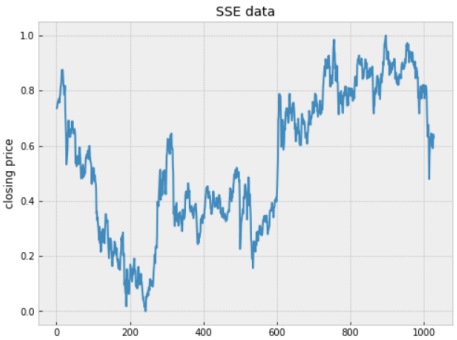
Rate of Variation: This figure shows the normalized percentage change or detrended values of the closing price, which indicates how volatile the three markets have been. Volatility corresponds to the amount of price change over an explicit time interval. In recent years the S&P 500, SSE, and FTSE 100 indexes have all been relatively stable with low volatility in

their price changes (Harris, 1989), and from **figure** 2.2 we can see that the index has also been relatively stable, once the data is undergone the differencing process.

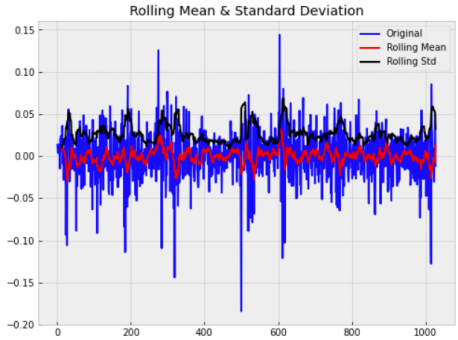
(a) Normalized Closing Price of S&P 500



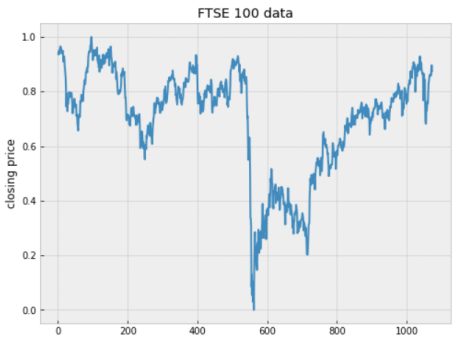
(b) Normalized rate of Change of the Closing Price



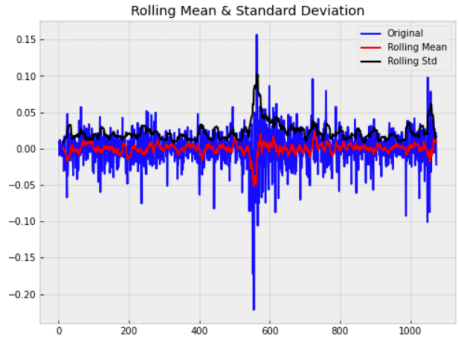
(c) Normalized Closing Priceof SSE



(d) Normalized rate of Change of the Closing Price



(e) Normalized Closing Price of FTSE 100



(f) Normalized rate of Change of the Closing Price

Figure 2.2: Illustration of Normalized and de-trended values of the closing price in each stock data.

Chapter 3
Theory

This chapter presents the theory corresponding to the background information required for understanding the techniques used in later chapters. A particular focus on the RNN architecture of LSTM and some theories relevant to the times series analysis. In addition to this neural network and some other methods, we will also discuss the Hybrid ARIMA-GRACH model, and the how models described in the following section are specified.

3.1 Time Series

Data collected over a series of time points or over some time is called a time series. Examples of time series include the start of new housing projects each month and the sale of products every week. In a time series, data are typically collected at equal intervals of time, such as hourly, daily, weekly, monthly, or yearly. The ultimate purpose of time series analysis is to develop forecasts for future values of the series [5].

3.1.1 Characteristics of the time series

In stochastic processes, we can view Y_t as a sequence of random variables. This process represents how we observe time series. According to the joint distribution of Y_t, this process has a complete probability structure. In this joint distribution, the majority of the information is represented by the mean, variance, and covariance. The main characteristic quantities are shown below:

- Mean: $\mu_t = EY_t$ —> Expected(Y_t)
- Variance: $DY_t = E(Y_t - \mu_t)^2$
- Covariance: $Cov(Y_t, Y_s) = E(Y_t - \mu_t)(Y_s - \mu_s)$
- Autocorrelation: $\rho_{t,s} = \frac{Cov(Y_t,Y_s)}{\sqrt{DY_t.DY_s}} = \frac{\gamma_{t,s}}{\sqrt{\gamma_{t,t}.\gamma_{s,s}}}$

3.1.2 Properties of Time series

Stochastic process

Stochastic processes are defined as collections of randomly arranged variables over time. Probabilistic laws determine how stochastic processes evolve. In this thesis, we observe that the stochastic process is responsible for the data points that comprise the stock index. Hence, the time series is a sample of the random variables within the stochastic process (Cryer and Chan, 2008; Box et al., 2016).

Stationarity

A time series is stationary if the statistical properties of the process generating it do not change over time. In other words, it does not mean the series does not change over time, just that its method of change is not itself changing over time. It is therefore a linear function, not a constant one, in algebra the value of a linear function changes as y increases, but the amount it changes remains constant – it has a constant slope; one value that captures that rate of change.

3.1.3 Autoregression

Autoregressive (*AR*) statistics is another widely used statistical process. Its worth at this time depends on previous time steps as well as a random shock term. In contrast to the MA process, the *AR* is not always stationary (Box et al., 2016). A general *AR* process is denoted *AR(p)*, where p is the order of the process. A general *AR* method can be written here:

$$Y_t = c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + ... + \phi_p Y_{t-p} + \epsilon_t \tag{3.1}$$

At some arbitrary time t, Y_t represents the process value, ϵ_t is the error term at time t, and ϕ_0 to ϕ_p are the parameters. So, the *AR(p)* value depends on how the process has evolved over the p time steps since the previous time step, and how the error term has changed. A process of order one is abbreviated *AR(1)*, whose value is based on the value at the time step before, and the shock or error term today. *AR(2)* values depend on the value at the two previous time steps as well as today's shock (Cryer and Chan, 2008; Box et al., 2016).

3.1.4 Moving Average

An important characteristic of the moving average *(MA)* process in time series analysis is that it is always stationary. The *MA* process is one of the most common stochastic methods used in time series analysis. Its value varies according to the current and previous value of the shock term. The general *MA* process is denoted *MA(q)*, where q stands for the order of process. the process can be written as,

$$Y_t = c + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + ... + \theta_q \epsilon_{t-q} \tag{3.2}$$

At any given point in time t, Y_t is the values of the process,t, ϵ_t, and ϵ_{t-q} are the random shock term and θ_1 to θ_q are the parameters. This means that the value of an *MA* at time t is determined by the shock term at time t, as well as all the shock terms q time steps back. For an *MA(1)*, the process value depends on the shock term in the current time and the shock term in the previous time frame. For *MA(2)* the process value is based on the shock term in the present and the previous time frames(Cryer and Chan, 2008; Box et al., 2016).

3.1.5 Differencing and integration order

It is possible to create non-stationary statistic processes to stationary. This can be accomplished by using differencing. Differences between consecutive values are calculated by transforming the process so that the data points in the process instead of the actual values become the differences. Therefore, with this method, the stochastic trend can be removed from a non-stationary process, stabilizing the mean and thus making the process stationary (Cryer and Chan, 2008). If a process is differentiated once into a new process, the original process is regarded as integrated of order one. Differentiation is used to achieve stationarity, which means that the order of integration states the number of differentiating steps required (Hamilton, 1994).

3.2 Statistical tests

3.2.1 Augmented-Dickey Fuller Test

Using ADF (Augmented Dickey-Fuller) as a statistical significance test gives results in a hypothesis test, including both null and alternative hypotheses. This will give us a p-value from which we can make inferences about whether the time series is stationary or not. Before getting into the ADF, we must know about the Unit Root test because the ADF test belongs to the unit root test.

Time series that are not stationary is said to possess a unit root. In technical terms, a unit root is defined as a time series that has a value of $\alpha = 1$ expressed in a simple regression form below,

$$Y_t = \alpha Y_{t-1} + \epsilon_t \tag{3.3}$$

where the Y_t is the value of the time series at a time t and ϵ_t is a residual of variables. An Augmented Dicker Fuller test is the extension of the unit root test, which removes the auto-correlation from the series and then tests similar to the procedure of the dickey-fuller test. Based on the statistic, the augmented dickey fuller test produces a negative result, and rejection of the hypothesis depends on that negative number; the more negative magnitude of the number represents the confidence of the presence of unit root at some level in the time series.

$$\Delta y_t = \alpha + \beta Y_{t-1} + \phi_1 \Delta_{Y_{t-1}} + \phi_2 \Delta_{Y_{t-2}} ... + \phi_{p-1} \Delta_{Y_{t-p+1}} + \epsilon_t \tag{3.4}$$

In the autoregressive process, α is a constant, β is the coefficient on a time trend, p is the lag order, and Δ is the difference operator. The unit root test is then conducted under the null hypothesis $\gamma = 0$ against the alternative hypothesis of $\gamma < 0$. Once a value for the test statistic

$$\mathrm{DF}_\tau = \frac{\hat{\gamma}}{SE(\hat{\gamma})} \tag{3.5}$$

is computed it can be compared to the relevant critical value for the Dickey-Fuller test. As this test is asymmetrical, we are only concerned with negative values of our test statistic DF_τ. If the calculated test statistic is less (more negative) than the critical value, then the null hypothesis of $\gamma = 0$ is rejected and no unit root is present.

3.2.2 The Ljung-Box Test

The Ljung-Box test verifies the randomness of a time series and also assesses its autocorrelation. For the pure randomness test, if the p value is less than 5% then we reject the hypothesis that the sequence is white noise.

A white noise series does not exhibit autocorrelation. In general, Q can be calculated as follows:

$$Q = N \sum_{r=1}^{k} \hat{\rho_r^2} \tag{3.6}$$

where ρ is the squared estimated autocorrelations, r is number of lags being tested and k is the degrees of freedom. Testing whether the sequence is a white noise sequence is equivalent to check whether the test statistic Q obeys the distribution χ_2 with the degree of freedom k.The original hypothesis $H_0 : \rho_1 = \rho_2 = \rho_m = 0$, $m \geqslant 1$ and alternative hypothesis H_1 : There is at least one $\rho_k \neq 0$,$m \geqslant 1$, $k \leqslant m$ at the significance of α.According to α, k, and χ^2. we can find out the corresponding value $\chi^2_\alpha(k)$. If $Q \leqslant \chi^2_\alpha(k)$, then we accept the original hypothesis that is, at the assumption that the sequence is a white noise sequence us accepted at the significance level α. If $Q > \chi^2_\alpha(k)$, then we reject the assumption at the significance level of α.

3.3 ACF and PACF Graph

ACF is an autocorrelation function that shows us the autocorrelation [5] of any series with its lagged values. The values are plotted along with the confidence bands. Essentially, it measures how well the present value relates to the previous value. There are various time series components, including trend, seasonality, cyclicity, and residuals. When finding correlations, the ACF considers all these components, so it is referred as a complete auto-correlation plot.

PACF stands for partial autocorrelation function. Instead of finding correlations of the present with lags, this method finds correlations of the residuals (leftover after removing effects already explained by the earlier lag(s)) with the subsequent lag value. This is why it is sometimes called partial, rather than complete because we remove already discovered variations before finding the new correlation. Therefore, if there is any hidden information in the residual that is modeled by the next lag, then we might get a good correlation, and we will keep that feature in our model.

3.4 ARMA mixed model

It is a combination of the *AR* and *MA* model, which predict future values using both the previous values and the errors, so it performs better than *AR* and *MA* model alone.

$$Y_t = c + \sum_{i=1}^{p} \phi_i Y_{t-i} + \varepsilon_t \tag{3.7}$$

$$Y_t = c + \sum_{i=0}^{q} \theta_i \varepsilon_{t-i} \tag{3.8}$$

$$Y_t = c + \sum_{i=1}^{P} \phi_i Y_{t-i} + \varepsilon_t + \sum_{i=0}^{q} \theta_i \varepsilon_{t-i} \tag{3.9}$$

where the ε_t is the white noise $\mathcal{N}(0, \sigma^2)$.By using the Backward shift operator, we can rewrite the above as θ and ϕ of B.The backward shift operator B is a useful notational device when working with time series lags: $BY_t = Y_{t-1}$ (Some references use L for lag instead of B for backshift.)

$$\theta_p(B) Y_t = \phi_q(B) \epsilon_t \tag{3.10}$$

The *ARMA* model is known for being parsimonious and redundant in its parameters. which often means it requires fewer parameters than *AR(p)* and *MA(q)* models. Moreover, when the equation is rewritten in terms of the Backward Shift Operator, then the θ and ϕ can be joined by a common factor, resulting in a simpler model. The parameter estimation of the times series model is done by different methods, like a method of moments(MOM), least-square estimates(LS), and maximum likelihood estimation(MLE). The idea for MOM is to find expressions for the sample moments and for the population moments and equate them. Since method-of-moments performs poorly for some models, we examine another method of parameter estimation: Least Squares. Even it may be more advantageous in Maximum likelihood estimates, that it uses all of the information in the data (not just the first few moments as in MOM).

3.4.1 Autoregressive Integrated Moving Average Model

As a generalization of an auto-regressive moving average - *ARMA* - model, an *ARIMA* model is composed of auto-regressive integrated moving averages. These two models are used to forecast or predict future points in time-series data. *ARIMA* is a regression method that measures the strength of a dependent variable about other variables. Rather than examining actual values, the model is designed to use differences between the values in the series to

predict future time series movement. *ARIMA* models are used when there is evidence of non-stationarity in the data. When analyzing time series, non-stationary data are always transformed into stationary data (Box et al.,2016)[5].

The trend and seasonal components are common causes of non-stationary data in time series. The differences step can be used to transform non-stationary data into stationary data. It can perform one or more differentiating steps to remove the trend component in the data. Similarly, seasonal differences can be used to remove the seasonal component.

ARIMA(p,d,q) is a general *ARIMA* model, where p refers to the order of AR, q to MA, and d stands for integration order. For example, for an *ARIMA(1,1,1)* the data has been differenced once, and the model has an $AR(1)$ and an $MA(1)$ part. An $ARIMA(0,0,1)$ is the same as an $MA(1)$, and an $ARIMA(2,0,0)$ is an AR process of order two (Pankratz, 1983) [1].The prediction is done by using the recursive forecasting strategy that is mentioned in the methods.

Once the model order has been identified (i.e., the values of p,d, and q), we need to estimate the parameters c,$\phi_1 ... \phi_p$,$\theta_1 ... \theta_q$ When python estimates the ARIMA model, it uses maximum likelihood estimation (MLE). This technique finds the values of the parameters which maximize the probability of obtaining the data that we have observed. For ARIMA models, MLE is similar to the least-squares estimates that would be obtained by minimizing

$$\sum_{t}^{T} \epsilon_t^2 \tag{3.11}$$

MLE gives exactly the same parameter estimates as a least-squares estimation. Note that ARIMA models are much more complicated to estimate than regression models, and different software will give slightly different answers as they use different methods of estimation, and different optimization algorithms. In practice, R will report the value of the log-likelihood of the data; that is, the logarithm of the probability of the observed data coming from the estimated model. For given values of p, d, and q, python will try to maximize the log-likelihood when finding parameter estimates.

3.5 ARCH & GARCH

The autoregressive conditional heteroskedasticity (*ARCH*) [4] model is an econometric model for time series data that describes the variance of the current error term as a function of the variance of the previous period's error terms; often, the variance is represented by the squares of the previous innovations. In general, the *ARCH* model is appropriate when an autoregressive model (*AR*) is assumed for the error variance, but if an autoregressive moving average (*ARMA*) model is assumed, the model would be a generalized autoregressive conditional heteroskedasticity (*GARCH*) model.

Autoregressive conditional heteroskedasticity:ARCH model can be expressed as:

$$y_t = c + \epsilon_t, \epsilon_t = \sigma_t z_t \tag{3.12}$$

where: y_t is an observed data series, C is a constant value,ϵ_t is the residual of times series values, z_t is the standardized residual, independently and identically distributed with a mean equal to 0 and variance tends toward 1 as the sample size tends toward infinity, σ_t is the square root of the conditional variance, and it is a non-negative process. *ARCH(q)* can be expressed in the following equation:

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 \tag{3.13}$$

with α_0, $\alpha_i \geqslant 0$ $(i = 1, ..., q)$. so σ_t^2 is non-negative.

Generalized autoregressive conditional heteroskedasticity: The *GARCH* model [4] can be thought of as an extension of an *ARCH* model. A generalized ARCH (GARCH) has a higher weight on recent data and a lower weight for faraway lags. When compared to ARCH, it uses only the most recent returns. Furthermore, *GARCH* is less constrained than *ARCH*, therefore its current conditional variance can be impacted by an infinite number of past squared errors, rather than overfitting. So now, the conditional variance σ^2 is expressed by *GARCH(p,q)* as:

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2 \tag{3.14}$$

Where the $\alpha_0 > 0$ and $\sum_{i,j=1}^{max(p,q)} (\alpha_i, \beta_j) < 1$.Note that α_i and β_j are the coefficients of the parameters *ARCH* and *GARCH* respectively. The ACF and PACF of the residuals help to specify the *GARCH* order p and q respectively.In order to estimate the parameters of the conditional volatility models in this paper maximum likelihood estimation is used, which is a constrained non-linear optimization problem. There exist many different algorithms to solve this problem. The focus of this paper lies on a linear search algorithm, which is the non-linear conjugate gradient method.

3.6 Seasonal-Autoregressive Integrated Moving Average Model

In a seasonal ARIMA model, seasonal AR and MA terms predict Y_t. It is similar to *ARIMA* models, we just have to increase a few parameters to account for the seasons. The seasonal component of the model consists of terms similar to non-seasonal components but involves backshifts in the seasonal period. The shorthand notation is shown in the **figure**3.1. For example, without differencing operations, the model could be written more formally as

$$(1)\Phi(B^m)\phi(B)(Y_t - \mu) = \Theta(B^m)\theta(B)\epsilon_t \tag{3.15}$$

The non-seasonal components are:

- AR: $\phi(B) = 1 - \phi_1(B) - ... - \phi_p(B^p)$
- MA: $\theta(B) = 1 + \theta_1(B) + ... + \theta_q(B^q)$

The seasonal components are:

- AR: $\Phi(B) = 1 - \Phi_1(B) - ... - \Phi_P(B^{Pm})$
- MA: $\Theta(B) = 1 + \Theta_1(B) + ... + \Theta_Q(B^{Qm})$

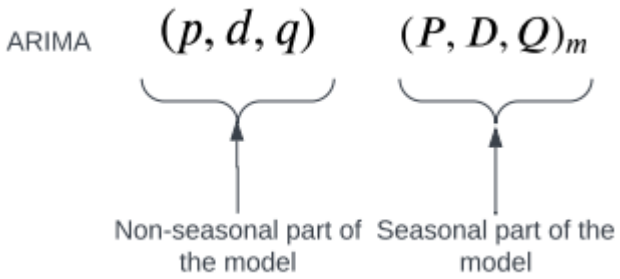


Figure 3.1: SARIMA model

where m is the period of a repeating seasonal pattern. We use the uppercase notation for the seasonal parts of the model, and lowercase notation for the non-seasonal parts of the

model. The SARIMA model can be just applied to the stationary processes. On the other hand, Augmented Dickey-Fuller is applied to test whether a time series is stationary, it is explained in section 3.2.1. Once all the initial parameters are fixed, we used a grid search to explode the meaningful parts of the space of the solution. Where the best model is assessed through the Akaike Information Criterion.

3.7 Hybrid ARIMA-GARCH

A hybrid model of $ARIMA$ and $GARCH$ is proposed with a two-phase procedure. During the first phase, the $ARIMA$ model is used to model the linear data of time series, and the residual of the linear model contains only the nonlinear data. In the second phase, the nonlinear patterns of the residuals are modeled using the $GARCH$ model. To analyze the univariate series and predict the values of the approximation series, this hybrid model combines an $ARIMA$ model with $GARCH$ error components Liu et al., 2013 [25]; Chen et al., 2011 [6]. It is observed that in this procedure, the error term ϵ_t of the $ARIMA$ model follows a $GARCH$ process of orders p and q.This hybrid model, which combines $ARIMA$ and $GARCH$ model containing nonlinear residuals patterns, is applied to analyze and forecast the returns of Close price The methodology of this hybrid procedure is shown in **figure** 3.2. In both phases, the parameter estimation is done with the help of Maximum Likelihood Estimates (MLE). Consider the returns $r1, ..., rn$ to be observations of independent and identically distributed random variables $R1, ..., Rn$ with the density function.Thus the random distribution is completely characterized with the set of unknown parameters $\alpha_i \& \beta_i$. Maximum likelihood estimation provides an estimate of the unknown parameters and as a parameter that maximizes the probability of the observed data.

3.8 Neural Network

Neural networks (NNs) are constructed from multiple artificial neurons that are connected and arranged in layers. The neural network (NN) is often described as loosely modeled after the human brain Dreyfus, 2005 [10]. It consists of neurons, each of which is dependent on input values. The system is a way to take inputs and calculate output based on those inputs. A linear combination of all these inputs is made by multiplying them by some weights, before adding a constant, or a bias. The linear combination of these inputs can be written as follows:

$$\mathrm{z} = \mathrm{w}_1 x_1 + \mathrm{w}_2 x_2 + ... + \mathrm{w}_\mathrm{n} x_n + b \tag{3.16}$$

The weights are w_1 to w_n, the inputs are x_1 to x_n, and the bias is b. This is done by the neuron using an activation function, which converts the number z to a meaningful format. This causes neurons to produce their outputs [27], which is

$$\hat{y} = f(\mathrm{z}) = f(\mathrm{w}_1 x_1 + \mathrm{w}_2 x_2 + ... + \mathrm{w}_\mathrm{n} x_n + b) \tag{3.17}$$

where f is some activation function in a NN, there are three types of layers. Normally, inputs are fed into the first layer of the model and then passed to the neurons in the second layer. The input layer is not composed of neurons because it does not compute anything based on the inputs. Last is the output layer, in which the final output of the network is generated. There are hidden layers between the input and output layers, which are composed of neurons that perform calculations on the data [10]. The layers of neurons in a neural network can be used to create neuronal networks capable of learning associations between input and output values, which can be used to solve classification and forecasting problems [33]. An example of a simple neural network can be seen in **figure** 3.3.

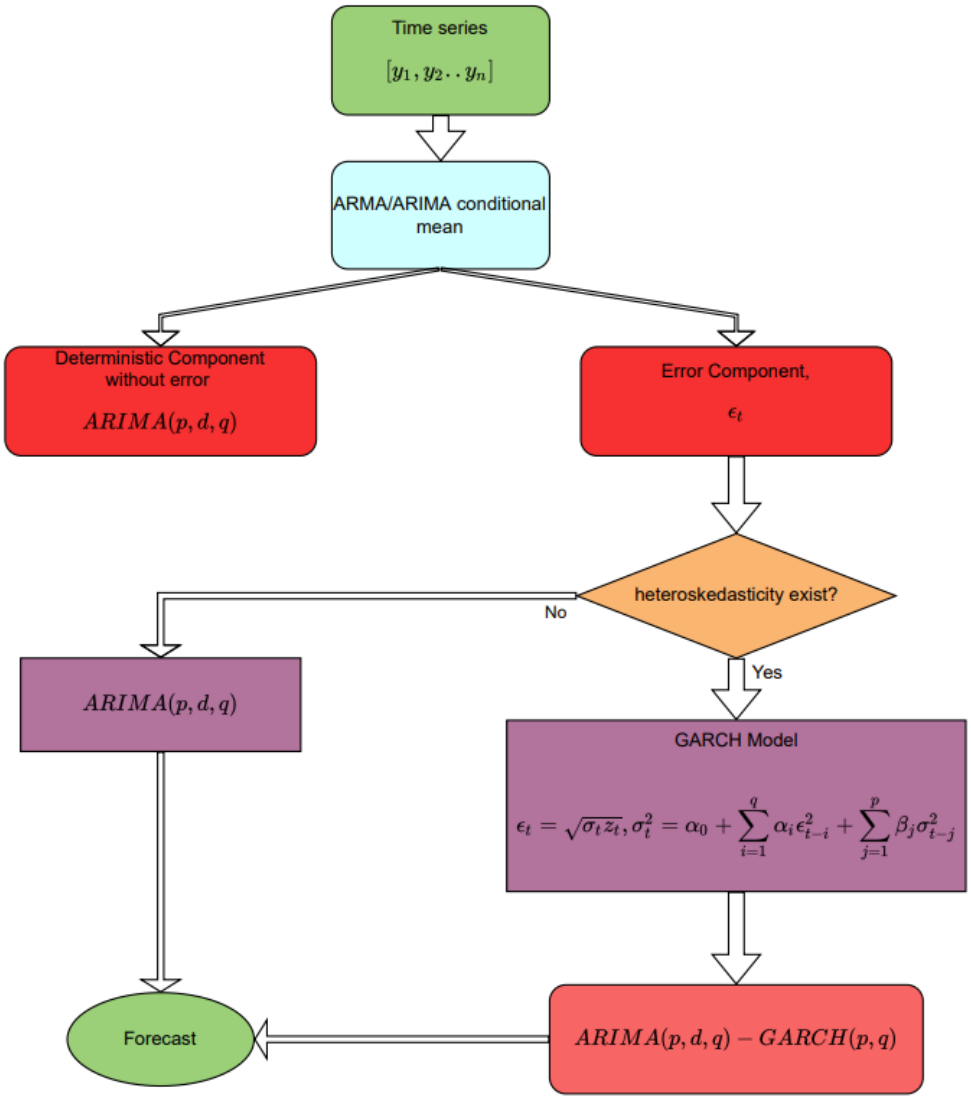


Figure 3.2: Procedure for hybridization of ARIMA and GARCH models

3.8.1 Feedforward Neural Network

A NN that moves the information from its input layer, through the hidden layer and its output layer is referred to as a feedforward network (Michelucci, 2018). This was the first kind of NN. Convolution neural networks are widely used for classification problems such as image recognition. A network can be thought of as a combination of different functions. An example of a simple FFNN is shown in **figure** 3.3. There are no limits to the number of layers and nodes, so it can be scaled up as well as down. After each epoch, the weights are updated using an optimization algorithm until the cost function has reached a minimum, which might be the global minimum or only a local minimum.

3.9 RNN

Recurrent neural networks (RNN) are more advanced than feedforward neural networks (NN). This type of network not only uses input values from the present time step but also

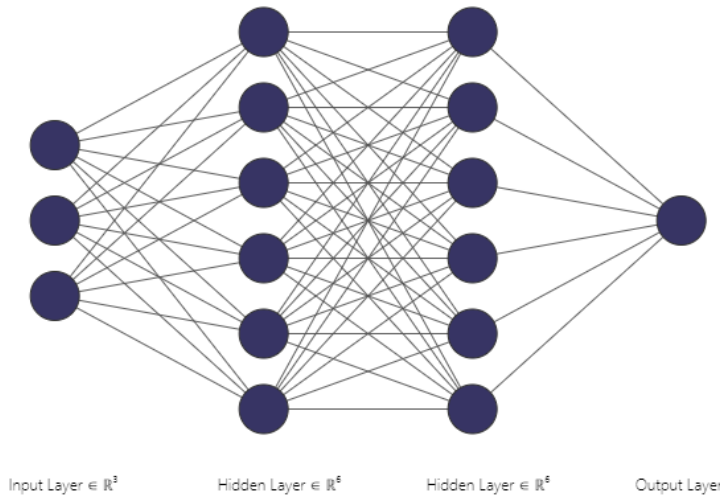


Figure 3.3: Schematic representation of a simple fully connected feedforward neural network with four layers.

the output values from the previous time step as input. Throughout a network's life, RNNs keep information that is relevant to various timesteps. It means that an RNN takes sequences and timesteps into consideration. Hence, this type of network is suitable to be used in sequential data processing applications such as speech recognition, machine translation, and time series forecasting. RNN is the second class of ANN, that has mostly succeeded in Time Series Forecasting (TSF) problems. They are thought to embed the concept of 'memory' in the system. In the feedforward neural networks, we assume that all inputs (and outputs) are independent of each other. However, in some cases, it might be a strong assumption, leading to wrong results. Therefore, to consider the correlation between the inputs and the outputs, RNNs implement a way to capture the information computed so far. Each unit takes as input the state computed in the previous iteration.

3.9.1 Long-term dependencies

Long-term dependency occurs when an RNN is required to make a prediction. Regular RNNs can handle the pattern understanding requirement. To do this, however, you need to know how far back in memory the values must be stored. An RNN is capable of predicting future sequences based on past data even in the case where it needs to look back in time. In a situation where the algorithm is required to remember a pattern, it becomes more challenging. Theoretically, an RNN could also achieve this if the parameters were adjusted by hand correctly. Researchers no longer need to spend time adjusting RNN parameters because they can instead use LSTM architecture [32].

3.9.2 Vanishing and Exploding Gradient's problem

Understanding the simplest form of RNN is useful for learning about networks that are well-suited for modeling and are widely used today. However, the simple RNN has a specific flaw which makes it not very useful. It suffers from the vanishing and exploding gradients two common problems encountered during the back-propagation of time-series data.With the first approach, the term goes to zero exponentially fast, which makes it hard to learn long-term dependencies. However, in the second case, the term goes to infinity exponentially, and thereby their value becomes a Nan due to the instability of the process. This is known as the exploding gradient. In the following sections, we use the LSTM model to deal with this problem. In theory, RNN can exploit information in arbitrarily long sequences. The main

issue of RNNs is that the cost functions are computed as a product of real numbers that can shrink to zero or explode to infinity. In literature, it refers to this problem as the exploding-vanishing gradient [17]. The problem of the disappearing gradient can occur with the tanh function because it has a steep derivative, causing a decrease in prediction and efficiency. The problem appears when the model goes through back-propagation, which is when the gradient of the cost function is calculated by the chain rule with respect to the weights and biases of each layer. The issue is generally resolved when the hyperparameters are correctly set by fine adjustment.

3.9.3 Optimizer

To improve performance without sacrificing much in training accuracy, we can use optimizers like SGD and Adam. Optimizer converges to the global minima. The basic difference between batch gradient descent (BGD) and stochastic gradient descent (SGD), is that to calculate the cost of one example for each step in SGD, but in BGD, we have to calculate the cost for all training examples in the dataset. Trivially, this speeds up neural networks greatly. Exactly this is the motivation behind SGD. Adaptive Moment Estimation (Adam) is the next optimizer, and probably also the optimizer that performs the best on average. Adam optimizer involves a combination of two gradient descent methodologies.it is a combination of the 'gradient descent with momentum' algorithm and the 'RMSP' algorithm. This algorithm is used to accelerate the gradient descent algorithm by taking into consideration the 'exponentially weighted average' of the gradients. Using averages makes the algorithm converge towards the minima at a faster pace.

3.9.4 Non-linear activation function

It is necessary to understand activation functions to understand how LSTM cells calculate. The objective of these functions is to transform values into more useful ones. The LSTM employs two activation functions: a hyperbolic tangent (tanh) function and a sigmoid (σ) function. The sigmoid function converts input values into values between zero and one, where a value of one indicates that the network keeps the input completely, while a value of zero means the network completely forgets the input (El-Amir and Hamdy, 2019)[2]. The LSTM cell uses the sigmoid function in its input, output, and forget gates. The tanh activation function is similar to the sigmoid except that its outputs range from minus one to one instead of zero to one. Activating the tanh function will negatively map negative inputs as well. When the input value is zero, the tanh function maps it as zero (El-Amir and Hamdy, 2019)[2]. Both activation functions are shown in **figure** 3.4.

3.9.5 Long Short term Memory Network

LSTM has analogous control flows to intermittent neural networks. This type of network processes data by propagating information forward. It differs from LSTM in the way it operates within its cells. The crucial conception of LSTM is the state of the cell as well as its gates. This cell state acts as a transport trace that transmits relative information down the sequence chain. This can be viewed as the network's" memory". The cell state, in the proposition, can carry applicable information throughout the processing of the sequence. So indeed information from earlier times way can make its way after time way, reducing the goods of short-term memory. As the cell state goes on its trip, information get's added or removed to the cell state via gates. The gates are different neural networks that decide which information is allowed in the cell state. During the training process, the gates are suitable to learn what information to keep or forget. In this study, we used a recursive forecasting strategy in LSTM to forecast different horizons.

$$tanh(x) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \qquad \sigma(x) = \frac{1}{1 + e^{-z}}$$

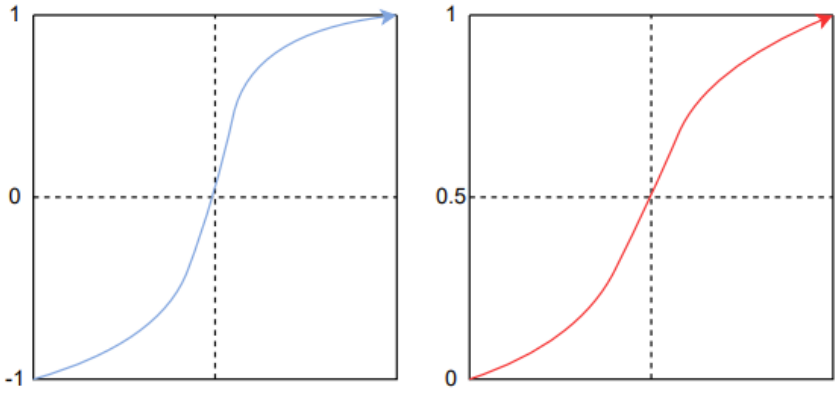


Figure 3.4: Activation function used in LSTM

3.9.6 LSTM cell structure

LSTM networks are built up from cells, each of which consists of several separate components. In **figure** 3.5, the lines represent the transmission of the vector in the direction of the arrow. There is no split in the values when the lines diverge; they are copied instead. LSTMs have three inputs: the memory from the previous timestep (c_{t-1}), the activation or input from the previous timestep (h_{t-1}), and the new data value at a time (X_t) (Purkait, 2019). The blue boxes show the activation functions that are used by the so-called gates.

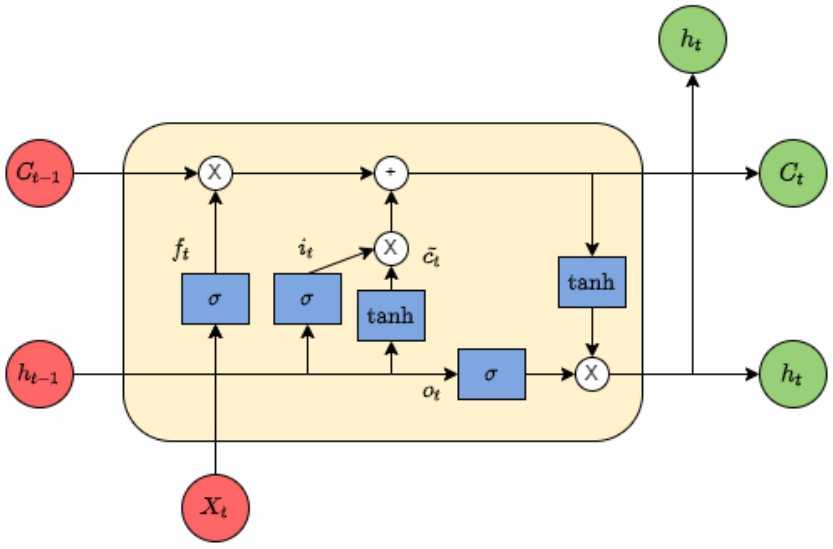


Figure 3.5: Structure of LSTM cell

According to El-Amir and Hamdy, 2019, gates are the place where the different operations occur. They are from left to right in **figure** 3.5: the forget gate, input gate, update gate, and output gate. A forget gate is a crucial part of the LSTM as it determines whether to keep a value from previous timesteps or to forget and incorporate it in the subsequent time

step. Such mechanisms work particularly well when dealing with long-term dependencies. Circular shapes represent pointwise operations, in this case, addition and multiplication.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3.18}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3.19}$$

$$\tilde{C}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{3.20}$$

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \tag{3.21}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{3.22}$$

$$h_t = o_t \times tanh(C_t) \tag{3.23}$$

It consists of W, h_{t-1}, x_t, and b, where W is the weight matrix, h_{t-1}, the previous time step input, x_t, the current time step input and b is the bias. Equation 3.18demonstrates how forget gates operate. The equation 3.19shows the calculation performed at the input gate to determine what values are updated. Following this is3.20, the update gate, where a vector of possible new memory is created. 3.21 the two previous equations are combined with the previous memory to create the new memory. 3.22 determines the output, which is then multiplied by the current memory 3.23 [18].

Chapter 4
Methods

This chapter summarizes the methods to answer the research question. This is accomplished by explaining how the different models are specified and how the parameters are chosen for each. The overall experimental procedure is shown in the **figure** 4.1

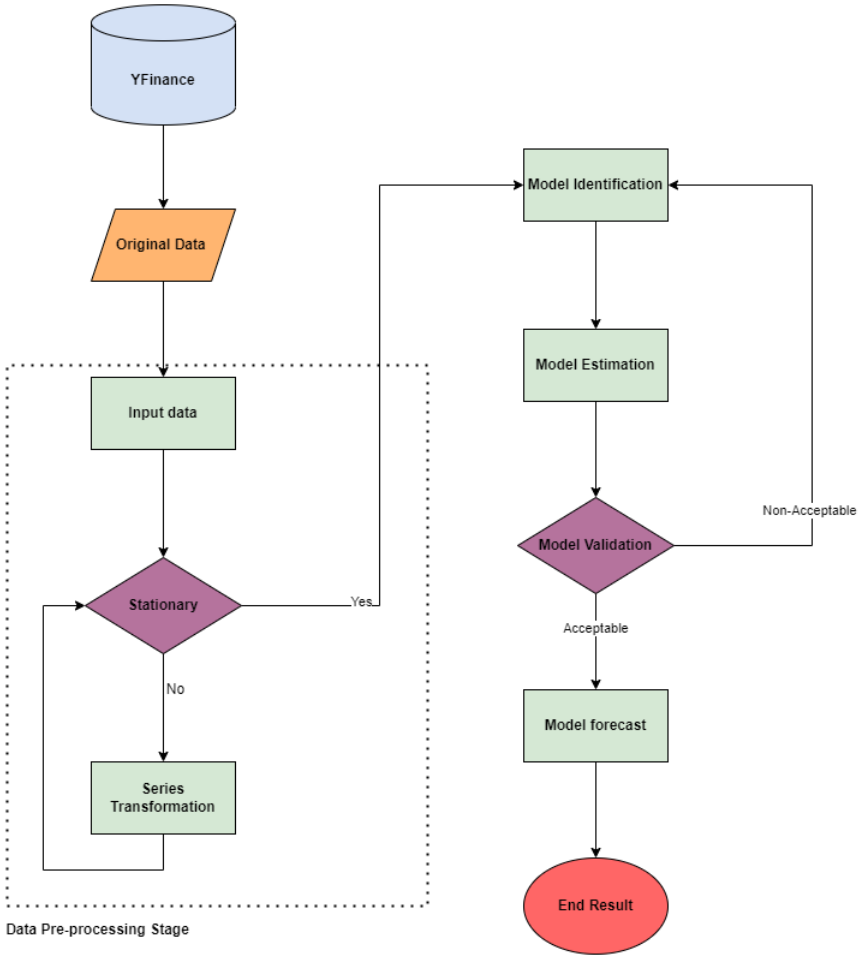


Figure 4.1: Proposed workflow for thesis

4.1 Akaike Information Criterion

This is one of the most commonly used information criteria, often abbreviated as AIC, and is designed to compare models to choose the one with the lowest AIC value (Cryer & Chan 2008). According to its definition, the AIC is a method for determining the relative loss of information for different models and is defined as

$$AIC = -2\log(maximum\ likelihood) + 2k \tag{4.1}$$

The term k is used to analyze the number of parameters in the model, for example in *ARIMA*, the term is $k = p + q + 1$ if an intercept is included and $k = p + q$ if it is not. Incorporating the term $2k$ into the model provides a penalty for overfitting the model by adding too many parameters. However, the AIC is widely regarded as biased in small samples, leading to the development of a successor, the corrected AIC (AICc), which reduces this bias by adding one more penalty term (Cryer & Chan 2008)[20]. The AICc is described as follows

$$AICc = AIC + \frac{2(k+1)(k+2)}{(n-k-2)} \tag{4.2}$$

As shown above, the AIC has been accompanied by another term considering the number of parameters. In this term, k stands for the parameters in the model as discussed above and n is the sample size. In forecasting, the AICc is preferable to other approaches to choosing models, especially when working with many parameters and small sample sizes (Cryer & Chan 2008).

4.2 Cross-validation

Time-series cross-validation can be used to include many point forecasts for evaluation when forecasting with a horizon of one or just a few steps from now (Hyndman and Athanasopoulos, 2018)[19]. Specifically, an expanding window will be used with the so-called walk-forward validation in this study. During a walk-forward validation, several forecasts with a short horizon are included by iteratively making point forecasts one step at a time, with multiple overlapping training sets. According to the expanding window, the training set gets larger with each new forecast, while the original observations are kept. The procedure of the walk-forward [8] validation is iterative and can be divided into the four following steps.

1. The different models are first estimated on the training set
2. The models are used to do a point forecast with forecast horizon h at the point t where t is the last point in the training set.
3. where the value of Y_{t+h} is predicted as the estimated value and therefore the known real value from the test set are compared.
4. For the next forecast, the training set is expanded by including the observation at $t + 1$ and the entire procedure in steps 1 to 4 is repeated for the entire test set.

4.3 Recursive Strategy

This method consists of using a one-step model multiple times and using the prediction from the prior time step as input for predicting the next time step. Our model will be trained using the normal training data. To forecast the stock closing price for the next two days, we would create a one-step forecasting model. In the next step, the model would be fit to predict day 1, and the results of that prediction would be used as a predictor for day 2. Once you reach

the necessary number of forecasting steps, depending on whether or not the parameters have already converged, you can fit the model a little further.

$$prediction(t+1) = model(obs(t-1), obs(t-2), ..., obs(t-n)) \tag{4.3}$$

$$prediction(t+2) = model(prediction(t+1), obs(t-1), ..., obs(t-n)) \tag{4.4}$$

4.4 Seasonal-ARIMA specification

At the start of the model specification, training and test sets of data are separated. There are three types of *SARIMA* models that are fitted for the three different time horizons. The financial time series are modeled based on a variety of methods, with closing prices being our target. Seasonal ARIMA is chosen since its differencing functionality can model non-stationary data components. Models are fitted using a Python package called sm-statistics. The ACF and PACF plots are used to specify the optimal parameters for the model. The grid search function yields the most appropriate model, chosen according to Akaike Information Criterion(AIC).

4.4.1 Model development for Seasonal-ARIMA

Seasonal-ARIMA forecasting requires stationary series. The ACF and PACF plot in the first row of **figure** 4.2, 4.3, 4.4 shows that lags with smaller values have greater and positive values. It also had a trend of slowly decreasing to low values with the improvement of lags. We also observed a strong seasonal pattern at the weekly level. Additionally, the data was identified as non-stationary ($H1$ rejected as p-value > 0.05) with the ADF hypothesis test. Therefore, to remove the linear trend and make the series stationary, the first difference (differences = 1) was taken. Differencing the trend was achieved by taking the differences between consecutive observations of a time series.

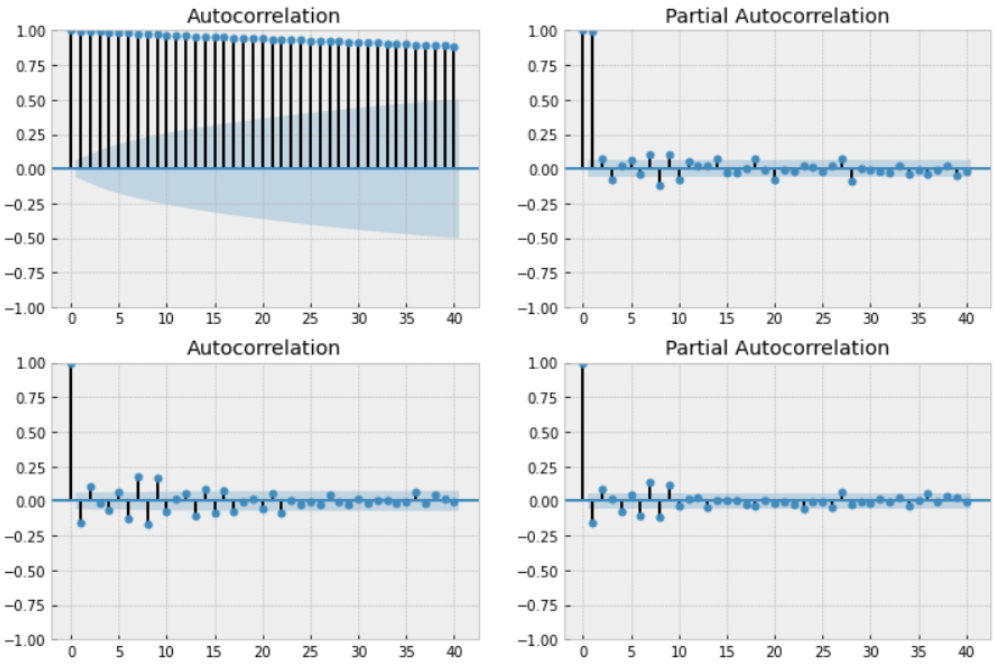


Figure 4.2: Correlation plot of S&P 500

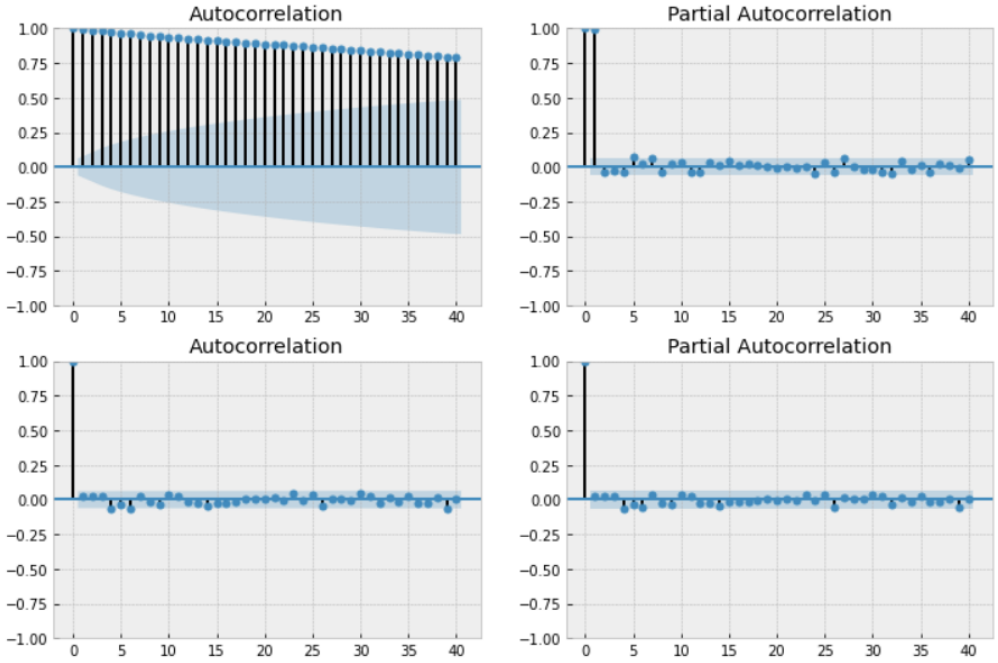


Figure 4.3: Correlation plot of Shangai stock exchange

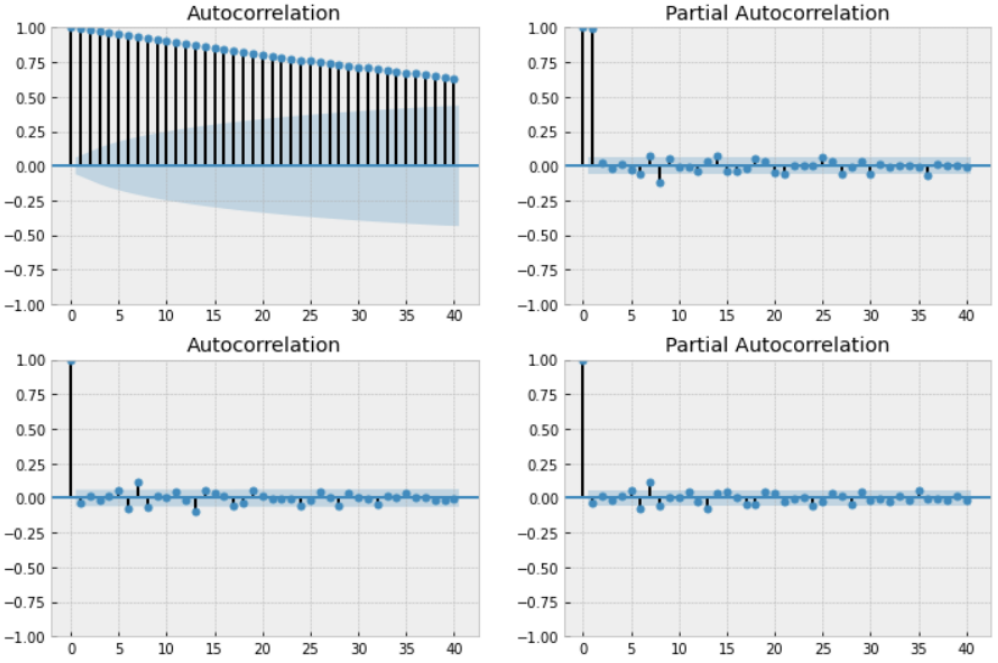


Figure 4.4: Correlation plot of FTSE 100

After transformation, the ACF and PACF plot in the second row of **figure** 4.2, 4.3, 4.4 showed the remaining correlation(s) between the observations. Over the lags, the ACF showed seasonality and trend, while the PACF has the cutoff at lag 12. From the ACF plot, we determined the MA(q) process order and from the PACF plot, we determined the AR(p) process order. The term 'd' was calculated by differencing the trend resulting in the SARIMA(p, d, q)(P, D, Q,m) model. And, the order D was based on seasonal differencing. On other hand to choose the optimal hyperparameters for the model, we used the **grid search** frame-

work which will optimize these parameters to minimize error with the help of the lowest AIC score in a model. The optimal model parameters of Seasonal ARIMA with an AIC score are shown in the table 4.1.

Parameter analysis for SARIMA model

An analysis is conducted only on the training set before implementing a systematic experiment to set the upper bounds of certain parameters. Based on the ACF/PACF graphs, all the parameters have been chosen. To validate the seasonality parameter across all datasets, we use the Grid search method. Moreover, they aren't easy to determine when it comes to P, p, and Q, q. In some cases, the autocorrelation decreases with an increasing lag. All the values up to 5 have been analyzed, but the time required to fit the model grew exponentially and little improvements were observed for values greater than 3. Therefore, the upper bound for all the parameters is set at 3. Considering the I parameters, they are set at 1 since the datasets become stationary after the first integration. With the Dickey-Fuller test, we have verified the consistency of the results. If the p-value of the time series is less than 0.05, then it is stationary. To evaluate the models, the AIC criterion was used.

stock index	Forecast days	p	d	q	P	D	Q	m	AIC score
S&P 500	1- 5	0	1	1	0	1	1	12	2276.099
	1-21	1	1	1	1	1	1	12	6375.938
	1-62	1	1	1	0	1	1	12	9992.074
SSE	1-5	0	1	1	0	1	1	12	2161.221
	1-21	0	1	1	0	1	1	12	5994.323
	1-62	1	1	1	0	1	1	12	9420.010
FTSE 100	1-5	1	1	1	0	1	1	12	2482.047
	1-21	1	1	1	0	1	1	12	7101.487
	1-62	0	1	1	0	1	1	12	11313.390

Table 4.1: Hyperparameters for SARIMA model

4.5 LSTM specification

We describe the different aspects of the LSTM NN specification in the sections below. We generate our models using the Keras API in TensorFlow, which is an open-source library for machine learning written in Python (Purkait, 2019).

4.5.1 Normalization

A Min-Max scaler should be used to normalize the input values (apart from the standardization that scales the features to have $\mu = 0$ and $\sigma = 1$). As a result, all observations will be transformed to the range [0,1], meaning that their minimum and maximum value will be respectively 0 and 1. After the forecasting using LSTM, the data were rescaled using inverse min-max normalization. y is y_{norm} which is normalized as follows,

$$y_{norm} = \frac{y_i - min(y)}{max(y) - min(y)} \tag{4.5}$$

where the $\hat{y}$ is reverted i.e., inverse transformed using the following formula,

$$y_{scaled} = \hat{y}_i.(max(y) - min(y)) + min(y) \tag{4.6}$$

4.5.2 Data Reshaping

The sequential supervised learning problem of time series forecasting with traditional ARIMA is different from LSTM forecasting, which requires another preprocessing step to be transformed into the classical supervised machine learning problem. We used the sliding window method instead of labeled data to target look-back steps of normalized data.

Sliding window approach:

Time series datasets can be restructured into supervised learning problems by using the value at the previous time step to predict the value at the next time step. The original dataset and the transformed dataset can be compared easily. The following observations can be seen below:

- In our supervised learning problem, the previous time step (X) is the input, and the next time step (Y) is the output.
- The order of observations within the dataset is preserved and must remain so when training a supervised model using the dataset.
- It is clear from the previous value that we cannot predict the first value in the sequence using the previous value. It will be deleted as it is useless.
- It is also clear that we do not have a known final value that predicts what will happen at the end of the sequence. Likewise, we may want to remove this value while training our supervised model.

The sliding window method is used to predict the next time step by using information from previous time steps. This method is commonly referred to as the window method. Time series analysis and statistics refer to this as the lag method. Therefore, previous time steps are shown as the lag size or order, or as the window width. After converting times series data into labeled times series values, we train the supervised linear or non-linear machine learning algorithm by using this data.

4.5.3 Model development for LSTM

The Long Short Term Memory(LSTM) model consist of two memory block and those blocks were connected via one input layer, two drop out layer, one hidden layer, and one output layer. The univariate close price of every stock is converted into sliding window data and input as a covariate to the LSTM model. The following sections give explanations for the different hyperparameters and clarify how they have been selected,

Layers: A neural network(NN) consists of several layers. As per the theory, only one hidden layer is called a shallow neural network, that was sufficient to solve any problem (Cybenko, 1989) [9]. The model consists of three layers. Two of the layers are LSTM layers and one is a Dense layer, where the LSTM layers perform the calculations previously shown in equations 3.18 to 3.23, and the Dense layer performs the more simple linear combination equations 3.16 and 3.17.

Neurons: Based on the empirical study that the number of neurons does not affect the performance of the model. In this work, the input layer, hidden layer, and output layer consist of 200, 50, and 1 neuron. The output layer consists of one single neuron because only one output value at each time is used to forecast time series problems such as this one.

Loss function: When training an LSTM model, it is necessary to specify a loss measure. It is intuitive to minimize the loss when determining parameters in the model, such as the number of epochs. To accelerate learning, the Loss function calculates the distance between a model's output and the desired output. For validation data, the user sets the desired output.

We have chosen to set the validation data at 10% of the training data. Overfitting can be prevented by stopping the model during training because the training data output will be compared with the validation data after each epoch. If the loss on the training decreases while the validation increases, then we may be overfitting. As a loss function, we are using the MSE, which is widely used in time series forecasting (Makridakis and Hibon, 1991).

Optimizer: When the network approaches optimal values, there is an additional parameter that can be given to an optimizer that will reduce the learning rate gradually. It is an algorithm used to minimize the loss function. This is computed by a CPU that knows the actual values in the training data and iteratively tries to end up where the loss function is minimized. Here we used *Adam* optimizer which handles large datasets well in addition to being appropriate to use with non-stationary data. Adam is used with the default configuration parameters which are recommended by its creators (Kingma and Ba, 2015)[22].

Table 4.2: Hyperparameters used for training the LSTM model

Hyperparameters	**Values**
Learning rate	0.01, 0.2
Epochs	25,30,40
Batch size	32
Optimizer	Adam
Dropout layer	0.2
Loss function	MSE
LSTM layer	2

Regularization: There are numerous ways to try to mitigate overfitting, and thus improve generalization in an LSTM. This is done using regularization techniques. In the network created for this thesis, dropout is used as the regularization technique since it is a method that has been proven to produce very good results. It randomly selected the nodes with the probability of 0.2. The dropout is an important technique that reduces overfitting by randomly choosing cells in a layer according to the probability chosen and set their output to 0.

Number Epochs: A LSTM NN is trained multiple times with a batch size of 32. To minimize the error of the model and to minimize the training loss of the model 25, 30, and 40 epoch is used. Where the size of the epochs varies based on the different stock index data.

Optimal Hyperparameter: In order to get an accurate prediction when backtesting our model, we must set up the hyperparameters correctly and adjust them as needed. We built our LSTM model using default hyperparameters that fit our case the best based on several papers that we believe to be valuable. After that, each hyperparameter will be examined one by one in order to determine the optimal value. The most suitable value for a hyperparameter is found by evaluating the LSTM model by backtesting it with the test data. We will then calculate the MSE between the model's prediction and the actual closing price for that day. After all potential values for that specific hyperparameter have been evaluated and fixed.

Validation data: In neural networks, several parameters must be validated, including the optimizer, loss function, activation function, hidden layer size, input window size, as well as the number of epochs. LSTM parameters were tuned at daily scales for all the aforementioned parameters. A time series is a type of data that consists of temporal correlations. Therefore, conventional K-fold cross-validation cannot be used to exploit temporal relationships among data points without causing data leakage. Hold-out cross-validation has been our preferred approach. The dataset is divided into a training and testing set. After this, the training set can be divided into a training subset and a validation subset. Tuning the hyperparameters is performed based on the validation subset. Our training set was used to train models, and our

validation set was used to validate the models. We train the models available to determine the upper bounds. We used 10% of the training phase given to the validation subset.

4.5.4 Window size

The optimal window length was determined by an empirical test. We shouldn't restrict the window size too short, because then the model won't acquire the longer dependencies, thus ignoring critical information. Another downside to a large window size is that it will add a greater amount of redundant noise and thus will overfit the training data [15]. To determine the most appropriate window size, we conducted the empirical tests on the different window sizes (Gers et al., 2002) and find the value with the smallest RMSE on the forecast, when training the LSTM model **figure** 4.5 the optimal window length for our case would be 20, 25, and 30 training days. To do this empirical test, the dropout parameter was set to 0.2, and the epoch value to 20.

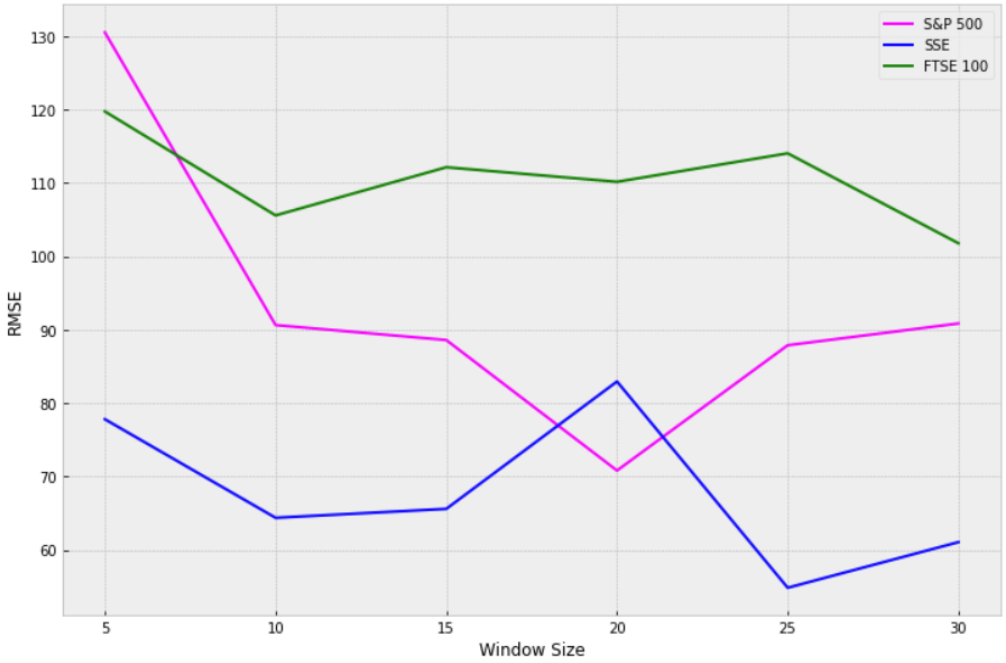


Figure 4.5: Illustrating empirical testing for finding the optimal length for the window hyper-parameter

4.6 Hybrid ARIMA-GARCH specification

Generally, *ARIMA* models are proposed for stationary time series under homoskedasticity assumptions, whereas financial time series data often violate these assumptions. It is common for stock prices to be extremely volatile during economic expansions and recessions. In such cases, the error distribution is heteroscedastic (also known as heteroskedasticity). Due to heteroskedasticity, *ARIMA* and linear regression give equal weights to all observations. This is because observations with a smaller disturbance variance contain more information than those with a bigger disturbance variance. Given that heteroskedasticity can affect the validity or power of statistical tests when using *ARIMA* models, the *ARCH* effect should be considered.

GARCH and ***ARCH*** models can therefore be used to not only capture the variance of each error term and correct the deficiencies of heteroskedasticity for least-squares analysis but also

tackle the problem of volatility clustering. We suggest that a *Hybrid ARIMA-GARCH* model can simultaneously predict both the conditional mean and heteroscedasticity of a process. In this combination, an *ARIMA* specification is applied for modeling the mean behavior, and a family of *GARCH* functions is used for predicting the variance behavior of the residuals from the *ARIMA* model. A *Hybrid ARIMA(p,d,q)-GARCH(r,s)* can be specified as:

$$\begin{aligned} y_t &= \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + \epsilon_t - \theta_1 \epsilon_{t-1} - ... - \theta_q \epsilon_{t-q} \\ \sigma_t^2 &= \alpha_0 + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2 \\ \epsilon_t &= z_t \sigma_t \ (\epsilon_t : N(0, \sigma_t^2)) \end{aligned} \tag{4.7}$$

4.6.1 Overview of Methodology and input parameters

To achieve the goals that were mentioned in the above sections, the methodology of this research is structured in the following way: Firstly, we conducted a rolling forecast based on an ARIMA model. To predict the return for the next point, we use the combination of p, d, and q that has the lowest AIC.

Table 4.3: Hyperparameters of the Hybrid model

Data	Horizon	Models	AIC
S&P 500	1-5	ARIMA(0,1,1)-GARCH(1,1)	2513.676
	1-21	ARIMA(2,1,5)-GARCH(1,2)	6567.577
	1-62	ARIMA(0,1,2)-GARCH(1,2)	10211.196
SSE	1-5	ARIMA(0,1,0)-GARCH(1,1)	2384.620
	1-21	ARIMA(2,1,2)-GARCH(1,2)	6208.894
	62	ARIMA(3,1,3)-GARCH(1,2)	9629.645
FTSE 100	1-5	ARIMA(1,1,2)-GARCH(1,1)	2735.934
	1-21	ARIMA(0,1,0)-GARCH(1,2)	7350.684
	1-62	ARIMA(0,1,0)-GARCH(1,2)	11555.985

Based on the correlograms of ACF and PACF for the first differenced series and the residuals series, there are 40 possible model combinations between ARIMA and GARCH for p = 2,1,0 , d =1, q = 2,1,0 , r = 1,0 and s = 2,1,0 . From the analysis conducted in the estimation stage, three of the ARIMA-GARCH models show significant results for each data. The results of AIC for those ARIMA models with a combination of GARCH parameters are shown in the table 4.3

As part of the diagnostic checking, a test of LjungBox Q-statistic, a heteroscedasticity test, and a normality analysis are conducted on the residuals of the model to check its appropriateness. In all the hybrid models considered, the ACF and PACF of the squared residuals are near zero, which demonstrates that the models are adequate, as indicated by an insignificant Ljung-Box Q-statistic, p-value. Ideally, we failed to reject the null hypothesis. The reason we want the p-value of the test to be greater than 0.05 is that this means the residuals of the time series model will be independent, which is often the assumption made when developing a model.

4.7 Training, Validation, Test set

We first divide the data into a training set and a testing set. The training set is used for both training and testing the models. The validation size is set at 10% of the training set size. During the training procedure, samples are never shuffled since it would reduce the temporal

correlation, which is the fundamental characteristic of a time series. The size of the testing set is arbitrarily fixed at 100-time points. It applies to all the modeling techniques. This value has been chosen since they comprise both short, medium, and long-term forecasting. On a daily level, 100 points are equal to 5 months of the forecast.

4.8 Error Measures for Evaluation

It is necessary to evaluate forecasts to determine which model makes better predictions. Here, two different error measures have been used to achieve this goal. A root means square error (RMSE) is a measure used to determine whether a model is accurate and how much error it produces. The Mean Absolute Error (MAE) is a superior measure when evaluating a model, according to critics (Willmott and Matsuura, 2005). Both RMSE and MAE are used to evaluate ARIMA and LSTM models, even though the MAE might be superior.

4.8.1 Mean Absolute Error (MAE)

The scale-dependent accuracy measure used is the mean absolute error (MAE). The MAE is an easily interpreted measurement that can be used to compare different forecasting approaches when using them on the same time series, or for time series measured on the same unit (Hyndman and Athanasopoulos, 2018). The MAE is calculated as

$$MAE = mean(|e_i|) = \frac{\sum_{i=1}^{n} |\hat{y}_i - y_i|}{n} \tag{4.8}$$

Even though MAE is restricted to the same time series for comparison, it is meaningful to use because of the easy and direct interpretation of the measurement.

4.8.2 Root Mean Squared Error (RMSE)

The Root-Mean-Square Error (RMSE) is a measure frequently used for assessing the accuracy of prediction obtained by a model. It measures the differences or residuals between actual and predicted values. The metric compares prediction errors of different models for a particular data and not between datasets. The formula for computing RMSE is as follows

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{n}} \tag{4.9}$$

Chapter 5
Result

The experiment was done on both traditional and machine learning methods. This chapter shows the results of the model. However, as the stock market is a very dynamic system, hence the pattern and dynamics presented by the model may not be the same always, stock market data is way more dynamic in a real-world scenario. This causes the machine learning model to not be able to capture the dynamic changes in data points. The stock index of three different countries has been used for forecasting the closing price.

5.1 Comparison Results.

We present a performance overview of each of the three methodologies. According to 4.7, metrics are used to evaluate the performance of forecasting models. However, to compare the models across the different datasets, RMSE and MAE appear to be the most reliable. It has been empirically observed that RMSE leads to choosing the most meaningful model. The rationale behind this choice is explained in Chapter 4. To better observe the tendency of the performance along the horizon, the metrics are computed at every point on horizon 5, every odd point on horizon 21, and every 5 points on horizon 62 is checked on the test data 100-time points.

Based on the **figure** 4.2, the PACF showed significant spikes, indicating an AR(1) model. In **figure** 4.2, the ACF showed correlations after trend differencing, thus MA(1) was selected. We also selected the trend differencing term d(1) and the seasonal differencing term D(1). The estimation of the *ARIMA(p, d, q)(P, D, Q)m* model became ARIMA(0, 1, 1) with seasonal order (0, 1, 1)[12] due to 12 business days. The grid search tested all possible combinations of variables and printed out the set that resulted in the lowest AIC, and chosen parameters are shown in the methods section. The combination of ARIMA-GARCH, with the power and flexibility it offers, and the ability to handle volatility and risk in the data series, allowed the ARIMA-GARCH model to provide a very promising approach to analyzing and forecasting. To avoid the overfitting of the model, we used the walk-forward validation method as mentioned in the method. Where the upper bound of parameters is fixed with the ACF and PACF plot. In addition to this, we used the Grid Search method to find the optimal parameters. So, all the model hyperparameters are chosen accordingly to get accurate results.

Table 5.1: The mean values of RMSE and MAE for Training and Test set in S&P 500 using horizon 5

Model	Train MAE	Train RMSE	Test MAE	Test RMSE
SARIMA	37.180	146.678	251.356	290.752
Hybrid ARIMA-GARCH	32.492	124.031	171.560	179.560
LSTM	51.675	56.736	126.295	217.155

Table 5.2: The mean values of RMSE and MAE for Training and Test set in S&P 500 using horizon 21

Model	Train MAE	Train RMSE	Test MAE	Test RMSE
SARIMA	58.584	147.584	300.633	329.922
Hybrid ARIMA-GARCH	62.674	124.493	235.190	195.908
LSTM	87.284	100.805	244.736	263.049

Table 5.3: The mean values of RMSE and MAE for Training and Test set in S&P 500 using horizon 62

Model	Train MAE	Train RMSE	Test MAE	Test RMSE
SARIMA	96.821	152.630	301.082	362.201
Hybrid ARIMA-GARCH	86.192	126.201	178.549	215.1231
LSTM	132.771	154.019	362.967	416.893

Dataset 1 (S&P 500)

In Tables 5.1, 5.2, and 5.3., we report the performance measure of each model on different forecast horizons in dataset 1. This table cleary indicates the average RMSE and MAE of training and test set of every model. We have designed SARIMA, Hybrid ARIMA-GARCH, and LSTM tested for 100-time points. For dataset 1, the Hybrid ARIMA-GARCH outperforms the SARIMA and LSTM. In general, it provides a more stable and reliable forecast along the whole forecast horizon. This S&P 500 data, shows the linear trend over the whole period and obviously, there are some sudden shocks. So all models are not performing well. Even though the classical approach, i.e., the hybrid model, performs well, the deep learning technique reach outs its stable performance over the entire horizons. From the figure, it shows that LSTM and hybrid models performed more similarly in the case of horizons 5 and 21. when it comes to the long horizon like 62 the Hybrid ARIMA-GARCH manages to have a lower error in the forecast.

To sum up, these figures 5.1 highlight how the Hybrid model outperforms SARIMA and LSTM techniques.

Dataset 2 (SSE)

In Tables 5.4, 5.5, and 5.6., we report the performance measure of each model on different forecast horizons in dataset 2. We have designed SARIMA, Hybrid ARIMA-GARCH, and LSTM tested for 100-time points. For dataset 2, the LSTM has a better performance. In general, it provides a more stable and reliable forecast along the whole forecast horizon. This SSE data shows the steady-state of variance and mean over the whole period and obviously, there are some sudden shocks. So all models are not performing well. It may be inferred that all the three models are very close to each other. One may expect similar performances also from an error point of view. On the other hand, the deep learning technique(LSTM) and Classical technique (Hybrid ARIMA-GARCH) reach outs their stable performance over the entire horizons. From the figure, it shows that LSTM and hybrid models performed more similarly in the case of horizons 5. To conclude, the figure 5.2 shows LSTM has stable performance and SARIMA and Hybrid ARIMA-GARCH had a tendency to increase in error throughout the entire horizons.

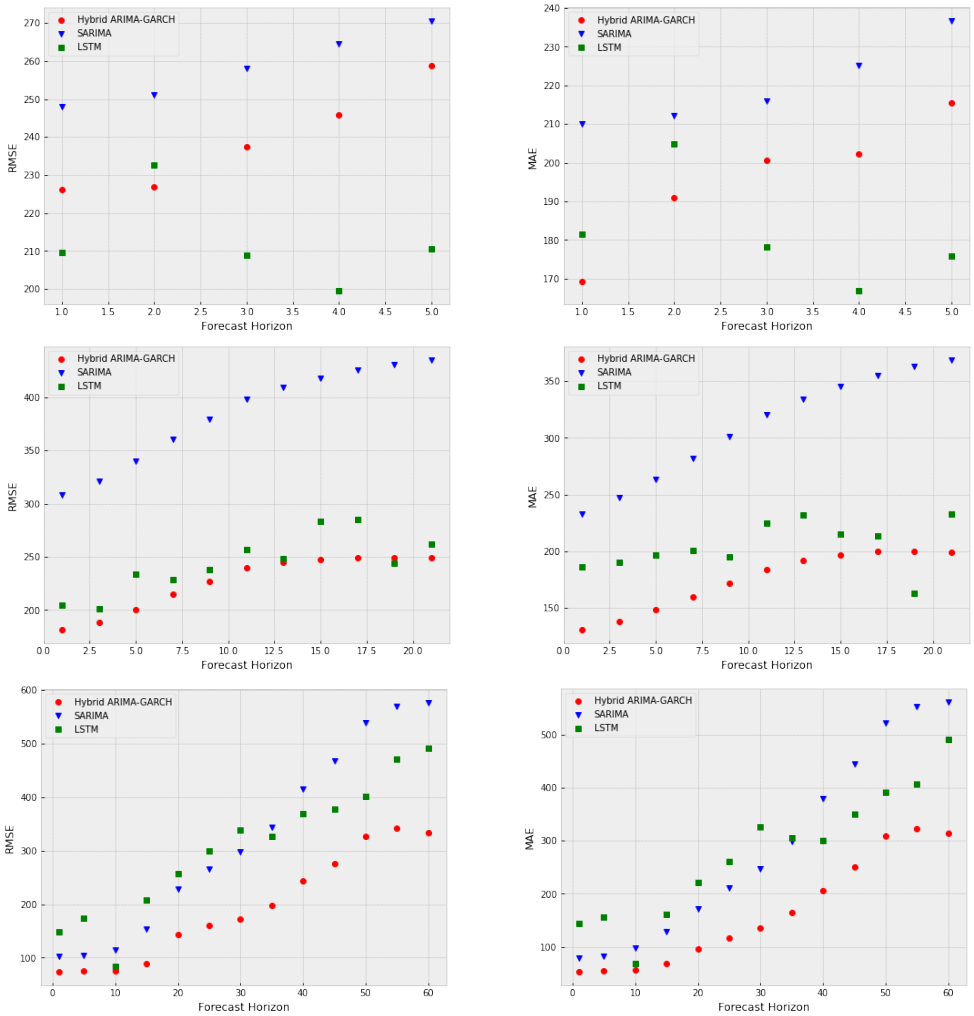


Figure 5.1: The performance of each method according to test RMSE and test MAE for the Dataset1(S&P 500)

Table 5.4: The mean values of RMSE and MAE for Training and Test set in SSE using horizon 5

Model	Train MAE	Train RMSE	Test MAE	Test RMSE
SARIMA	32.512	149.044	118.017	121.557
Hybrid ARIMA-GARCH	28.139	125.636	102.335	106.781
LSTM	46.815	55.945	104.395	109.083

Dataset 3 (FTSE 100)

In Tables 5.7, 5.8, and 5.9., we report the performance measure of each model on different forecast horizons in dataset 3. We have designed SARIMA, Hybrid ARIMA-GARCH, and LSTM tested for 100-time points. For dataset 3, the LSTM has a better performance. In general, it provides a more stable and reliable forecast along the whole forecast horizon. This FTSE 100 data, shows the steady state of variance and mean over the whole period, and obviously, there are some sudden shocks. On the other hand, the deep learning technique(LSTM) reach

Table 5.5: The mean values of RMSE and MAE for Training and Test set in SSE using horizon 21

Model	Train MAE	Train RMSE	Test MAE	Test RMSE
SARIMA	54.925	150.0594	104.194	116.863
Hybrid ARIMA-GARCH	47.621	127.419	88.921	102.351
LSTM	82.654	96.897	106.53	110.225

Table 5.6: The mean values of RMSE and MAE for Training and Test set in SSE using horizon 62

Model	Train MAE	Train RMSE	Test MAE	Test RMSE
SARIMA	82.136	172.771	96.553	117.424
Hybrid ARIMA-GARCH	73.506	128.764	91.744	110.571
LSTM	90.493	152.316	113.791	150.892

outs its stable performance over the entire horizons. From the figure, it shows that SARIMA and hybrid models performed more similarly in the case of horizons 21 and 62.

To conclude, the figure 5.3 shows LSTM has better performance, and SARIMA and Hybrid ARIMA-GARCH had a tendency to increase in error throughout the entire horizons.

Table 5.7: The mean values of RMSE and MAE for Training and Test set in FTSE 100 using horizon 5

Model	Train MAE	Train RMSE	Test MAE	Test RMSE
SARIMA	76.779	374.013	162.57	172.464
Hybrid ARIMA-GARCH	64.66	315.054	154.806	165.233
LSTM	98.755	110.582	152.977	160.470

Table 5.8: The mean values of RMSE and MAE for Training and Test set in FTSE 100 using horizon 21

Model	Train MAE	Train RMSE	Test MAE	Test RMSE
SARIMA	123.731	386.365	172.533	191.565
Hybrid ARIMA-GARCH	113.86	326.594	166.305	185.845
LSTM	211.919	234.925	126.399	159.257

Overall, time-series data exhibited significant stochastic trends as well as deterministic time series trends. A deterministic time series trend is predictable whereas a stochastic trend is unpredictable, therefore forecasting with this data did not produce accurate results. In this report, machine learning models have shown to be able, to some extent, to model and predict stock closing prices by using different forecast horizons. Classical approach - Hybrid model and machine learning approach - LSTM performed better than SARIMA. The LSTM seems to be more efficient in guessing the actual values. The evaluation metrics indicate a negative gap between the short and long-term and the inadequacy of both methods in producing a recursive forecast and learning patterns over time. Another indication that the models are quite robust. In any case, it becomes more and more difficult to predict the future using only

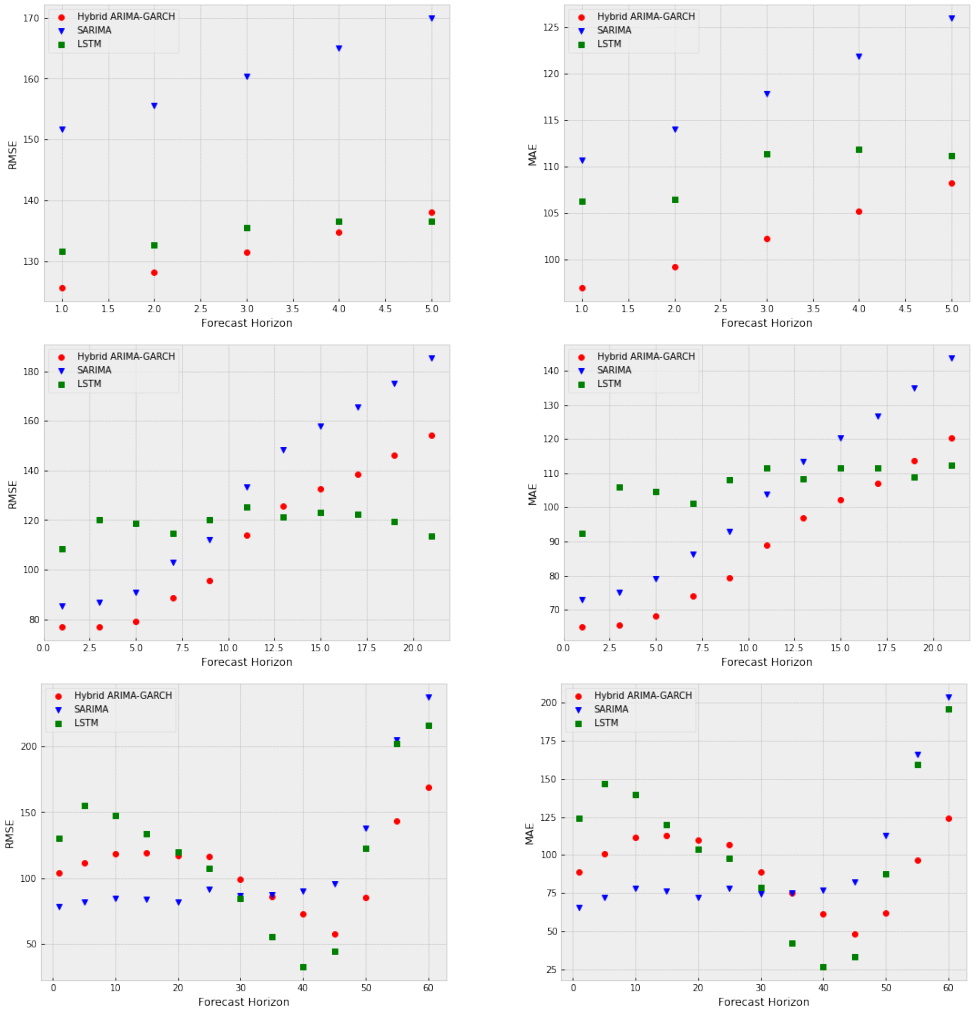


Figure 5.2: The performance of each method according to test RMSE and test MAE for the Dataset2(SSE)

Table 5.9: The mean values of RMSE and MAE for Training and Test set in FTSE 100 using horizon 62

Model	**Train MAE**	**Train RMSE**	**Test MAE**	**Test RMSE**
SARIMA	152.343	392.660	183.022	210.316
Hybrid ARIMA-GARCH	135.688	322.302	172.657	198.164
LSTM	250.668	293.646	202.523	234.434

current samples alone. As the size of the network grows, we may observe degradation in the quality of the forecast.

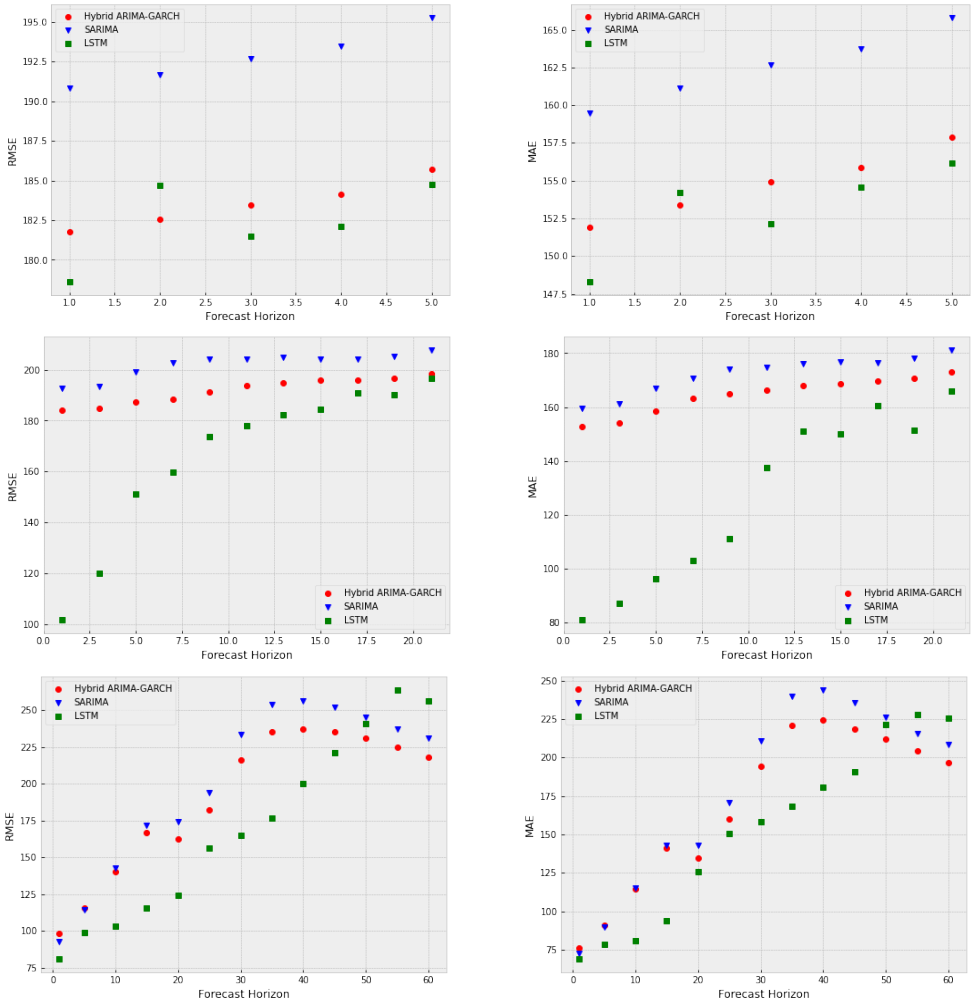


Figure 5.3: The performance of each method according to test RMSE and test MAE for the Dataset3(FTSE 100)

Chapter 6
Discussion

This chapter contains an evaluation of the achieved research result, relevant achievements, and recommendations for future work based on errors and obstacles.

6.1 Discussion

According to the results, the initial hypothesis makes sense. Stock market fluctuations could be explained by the very random nature of the market, even if it is unlikely. Even though it is not dependent on the equivalent of a coin flip for its value, past data may only have a very small impact on the present and future value of an index. Based on the AIC score, the time series process seems to be a random walk. This report shows that machine learning models can, to some extent, be used to model and forecast stock close patterns daily. We observed that both Hybrid ARIMA-GARCH and LSTM have better performance across all datasets, as shown in the results. Linearity of the patterns explains this outcome; indeed, daily data exhibit linear behavior. Besides, the limited amount of samples does not help. Neural networks are deemed to need a large amount of data to learn the patterns behind them. In dataset 1 Hybrid ARIMA-GARCH outperforms ML techniques, but the gap is minimal. However, this measure is not extremely accurate. If we look at the actual forecast we may notice that the forecast is not as good as expected. It might be due to different reasons, but in this case, RMSE is not reliable. The second and third datasets have the best performance. It is indeed, the most challenging scenario, where the data is more stable. Anyway, LSTM and Hybrid ARIMA-GARCH have an overall lower error return. If we look at the real forecast, it can be considered reasonable. To answer the first research question the LSTM and Hybrid ARIMA-GRACH are best in this case.

The SARIMA model can hardly be affected by outliers, explaining this behavior. This research does not address the handling of outliers. As a result, a grid search technique is necessary to find the most effective parameters. Grid search is time-consuming, and it has to be performed every time a model needs to be modified. Moreover, SARIMA has seven parameters, meaning the number of possible combinations could explode, making it impossible to explore fully. The grid search would require an excessive amount of time when the input size increases because it would not be able to exploit all the points. The use of machine learning techniques along with the largest input size, on the other hand, has been shown to improve their performance. As you can see in the plots, the performance follows a low error rate. Therefore, they may outperform statistical methods. In all the datasets, if we look at the graphs we notice how each model performs on different horizons. Both the hybrid model and LSTM results are quite impressive. Overall, we can affirm that the model produces a closer error along the forecast horizon. It is another piece of evidence that the models are quite robust. Perhaps, if we increased the size of the network, we would observe a drop in the quality of the forecast. Curios are the fact that the Hybrid model and SARIMA forecast seem to follow the same trend with different amplitudes.

SARIMA and *Hybrid ARIMA-GARCH*, as well as *LSTM*, were all employed to forecast stock closing prices with different forecast horizons. Using traditional methods, data used for forecasting has the same scaling (no normalization is done). However, in the case of LSTM, we used the *min-max* scaler function to normalize the data. The LSTM still outperforms the other two methods with all horizons with reliable errors on all datasets. Based on the evaluation between our two approaches prediction and the actual closing price, we can see that the error was larger when the S&P 500, SSE, and FTSE 100 closing prices tended to be more volatile. Using a univariate time series model alone will not reduce error during volatile periods of the index, since the volatility occurs during such times, for example during the financial crisis or when some negative news about the index is broadcast. When this happens, LSTMs will have a hard time predicting accurately. When building a deep learning model, it is foolish to rely solely on historical closing prices as input parameters. Stock/index price movements are complicated, as there are many factors involved, including macroeconomic events, market noise, investor sentiment, etc. These factors also contribute to stock/index price movements.

Even though the stock market is inherently complex, simply relying on the closing price of the previous days is a very narrow approach. Since this model does not include fundamental aspects such as the profitability of companies, economic policies, and some other political aspects. Market shocks induced by unforeseen events are also impossible to forecast. Adding indicators to this would likely make the model more accurate. This thesis is only limited by the fact that it models the SSE, FTSE 100, and S&P 500 indexes. As a result, if one applied the same model using different hyperparameters, one would find that the conclusions differ from the forecast to a certain degree. Initially, it would seem that a network that trains itself on past data and then makes predictions is a good idea for predicting stock market movements, but this view is quite naive. Any model that incorporates only past data in its forecasts is unlikely to be able to accurately predict market values in the future. If one tried to use a trading strategy or an investment strategy, it might not end catastrophically, but it wouldn't be as successful as if the forecasts had been accurate. That's not to say neural networks and machine learning do not have any practical use, on the contrary, in areas where forecasts can be made based on past data, the power to identify indicators ahead of time is extremely valuable. When it comes to their use in finance, given the fact that finance professionals use machine learning more and more with every passing year, they are incredibly relevant, even if they're not exactly a magical entity that predicts future values based only on past data.

We might be able to improve the market forecasting by creating some model that takes into account both historical prices and other factors such as macroeconomic indicators, the intrinsic value of the companies that compose an index, and perhaps some kind of sentiment indicator.

6.1.1 Relevant achievements

The process of building a neural network capable of forecasting time series takes a lot of time. It is a complex architecture with a lot of parameters that need to be tuned. Trial and error is typically employed to find the most appropriate solution. The most significant outcome of this research is a simplified architecture created by combining the approach described in [7] with the existing architecture, known as LSTM. In addition, we provided a way for embedding the information obtained from temporal exploration into the network. This approach, where the input and output layers of the LSTM are fixed, has been applied to three datasets, with better results compared to the SARIMA and hybrid ARIMA-GARCH. Furthermore, the constraints on the layer size make it easier to speed up the process of designing the network. A second notable outcome is the different forecast horizon used in the tests. This work has a forecast horizon of 5,21 and 62 with a test data size of 100 time points. A reasonable forecast was provided by our architecture, demonstrating its high robustness. Finally, we performed all the experiments on real-world scenario time-series, which supports our findings.

Chapter 7
Conclusion

Based on the analysis in this article, it is evident that deep learning-based algorithms and techniques have a lot of potential in the economics and financial fields. In finance and economics, several other prediction problems can be addressed using machine learning. To do this, we fitted three SARIMA models, three hybrid ARIMA-GARCH models, and three LSTM neural networks to the same data from the S&P 500, SSE, and FTSE 100 index and evaluated their performance at different horizons.

Recent advancements in developing sophisticated machine learning-based techniques, particularly deep learning algorithms, have caused these methods to gain popularity among researchers across a wide variety of disciplines. These newly introduced methods should prove to be as effective and accurate as traditional methods. A comparison of classical (SARIMA, Hybrid ARIMA-GARCH) and machine learning (LSTM) methods are presented here, as representative models for the forecasting of time-series statistics. Two of these techniques were implemented and applied to a set of financial data. The results showed that LSTM was superior to the other two methods. Furthermore, it is demonstrated that the results would be different if different datasets were used in the forecasts.

However, one conclusion to be drawn regarding the effectiveness of the two presented methods is where the LSTM neural network consistently outperforms the classical approach. Because none of the models are highly accurate, trying to make a profit trading using any of them would probably be futile.

7.1 Future work

There are several components to research in this field that should be incorporated into other research frameworks. When backtesting data, the first step would be to optimize a hyperparameter to eliminate overfitting. Second, one should always make a prediction and compare it with what is happening in real-time. Instead of just presenting backtested graphs. Another aspect would be to use sentiment analysis to forecast the volatility of the index price since we saw that the error of the prediction increased when the index price tended to be volatile. If we had considered more factors than just past index price trends, we would have made a more informed decision. A KST (Kolmogorov–Smirnov test and T statistic) method is will use for the construction of a correlation network based on the fluctuation of each time series within the multivariate time signals. So KST technique is used to analyze which day of the week is following the same distribution. Then find the distribution of every business day in a week, where a model is fitted on each cluster to forecast values based on different horizons. Combine those forecasts at the end. This type of implementation may increase the complexity of the system, but it might improve both the accuracy of prediction and forecast horizon.

Bibliography

[1] Pankratz Alan. "Forecasting With Univariate Box-Jenkins Models: Concepts and Cases." In: (1983).

[2] Hisham El-Amir and Mahmoud Hamdy. *Deep Learning Pipeline : Building a Deep Learning Model with TensorFlow.* Apress, 2020. ISBN: 9781484253489.

[3] Weiss Andrew A. "ARMA model with ARCH errors." In: *Journal of Time Series Analysis* 5 (2008), pp. 129–143. ISSN: 1467-9892.

[4] Tim Bollerslev. "Generalized autoregressive conditional heteroskedasticity". In: *Journal of Econometrics* 31.3 (1986), pp. 307–327. ISSN: 0304-4076. DOI: https://doi.org/10.1016/0304-4076(86)90063-1.

[5] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[6] Chenyi Chen, Jianming Hu, Qiang Meng, and Yi Zhang. "Short-time traffic flow prediction with ARIMA-GARCH model". In: *2011 IEEE Intelligent Vehicles Symposium (IV)* (2011), pp. 607–612.

[7] Kai Chen, Yi Zhou, and Fangyan Dai. "A LSTM-based method for stock returns prediction: A case study of China stock market." In: *2015 IEEE International Conference on Big Data (Big Data)* (2015), pp. 2823–2824. ISSN: 9781479999262.

[8] *Cross validation for time series*. Robjhyndman, December 05, 2016 [Online]. URL: https://robjhyndman.com/hyndsight/tscv/.

[9] G. Cybenko. "Approximation by superpositions of a sigmoidal function." In: *Mathematics of Control, Signals, and Systems* 5 (2005), p. 455. ISSN: 1435-568X.

[10] G. Dreyfus. *Modeling with Neural Networks: Principles and Model Design Methodology.* Springer Berlin Heidelberg, 2005. ISBN: 978-3-540-22980-3.

[11] Robert Engle, Victor K. Ng, and Michael Rothschild. *Asset pricing with a factor-arch covariance structure: Empirical estimates for treasury bills*. 1990.

[12] Robert F. Engle. "ARCH with estimates of the variance of UK inflation." In: *Econometrica* 50.4 (1982), pp. 987–1007. ISSN: 00129682.

[13] C. W. J. Granger. "Investigating Causal Relations by Econometric Models and Cross-spectral Methods". In: *Econometrica* 37.3 (1969), pp. 424–438. ISSN: 00129682, 14680262.

[14] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. "A Novel Connectionist System for Unconstrained Handwriting Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31 (2009), pp. 855–868.

[15] Alex Graves and Jürgen Schmidhuber. "Framewise phoneme classification with bidirectional LSTM and other neural network architectures". In: *Neural networks : the official journal of the International Neural Network Society* 18 5-6 (2005), pp. 602–10.

[16] Rahim H A and Ghani I M Md. "Modeling and Forecasting of Volatility using ARMA-GARCH: Case Study on Malaysia Natural Rubber Prices." In: *IOP Conference Series: Materials Science and Engineering* 548 (2019), p. 012023. ISSN: 1757-899X.

[17] Sepp Hochreiter. "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6 (Apr. 1998), pp. 107–116. DOI: 10.1142/S0218488598000094.

[18] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[19] Rob J Hyndman and Yeasmin Khandakar. "Automatic time series forecasting: the forecast package for R". In: *Journal of Statistical Software* 26.3 (2008), pp. 1–22. DOI: 10.18637/jss.v027.i03.

[20] TIMOTHY D. JOHNSON. "Time Series Analysis with Applications in R, 2nd edition by CRYER, J. D. and CHAN, K.-S." In: *Biometrics* 65.1 (2009), p. 337. ISSN: 0006341X.

[21] Wasiat Khan, Mustansar ali Ghazanfar, Muhammad Awais Azam, Amin Karami, Khaled Alyoubi, and Ahmed Alfakeeh. "Stock market prediction using machine learning classifiers and social media, news". In: *Journal of Ambient Intelligence and Humanized Computing* (Mar. 2020). DOI: 10.1007/s12652-020-01839-w.

[22] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2015).

[23] Zhang Lei and Peng Xiongwei. "Time series estimation of gas sensor baseline drift using ARMA and Kalman based models". In: *Sensor Review* 36.1 (2016), pp. 34–39. ISSN: 0260-2288.

[24] Xiaodong Li, Pangjing Wu, and Wenpeng Wang. "Incorporating stock prices and news sentiments for stock market prediction: A case of Hong Kong". In: *Information Processing Management* 57 (Feb. 2020), p. 102212. DOI: 10.1016/j.ipm.2020.102212.

[25] Heping Liu and Jing Shi. "Applying ARMA–GARCH approaches to forecasting short-term electricity prices". In: *Energy Economics* 37 (2013), pp. 152–166. ISSN: 0140-9883. DOI: https://doi.org/10.1016/j.eneco.2013.02.006.

[26] Junling Luo, Zhongliang Zhang, Yao Fu, and Feng Rao. "Time series prediction of COVID-19 transmission in America using LSTM and XGBoost algorithms". In: *Results in Physics* 27 (2021), p. 104462. ISSN: 2211-3797. DOI: https://doi.org/10.1016/j.rinp.2021.104462.

[27] Umberto Michelucci. *Applied Deep Learning : A Case-Based Approach to Understanding Deep Neural Networks.* Apress, 2018. ISBN: 9781484237892.

[28] Adil Moghar and Mhamed Hamiche. "Stock Market Prediction Using LSTM Recurrent Neural Network". In: *Procedia Computer Science* 170 (2020), pp. 1168–1173. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2020.03.049.

[29] Felipe Dias Paiva, Rodrigo Tomás Nogueira Cardoso, Gustavo Peixoto Hanaoka, and Wendel Moreira Duarte. "Decision-making for financial trading: A fusion approach of machine learning and portfolio selection". In: *Expert Systems with Applications* 115 (2019), pp. 635–655. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2018.08.003.

[30] A.J. Robinson, G.D. Cook, D.P.W. Ellis, E. Fosler-Lussier, S.J. Renals, and D.A.G. Williams. "Connectionist speech recognition of Broadcast News". In: *Speech Communication* 37.1 (2002), pp. 27–45. ISSN: 0167-6393. DOI: https://doi.org/10.1016/S0167-6393(01)00058-9.

[31] Sima Siami-Namini and Akbar Siami Namin. "Forecasting Economics and Financial Time Series: ARIMA vs. LSTM". In: *ArXiv* abs/1803.06386 (2018).

[32] Fariha Sohil, Muhammed Sohail, and Javid Shabbir. "An introduction to statistical learning with applications in R". In: *Statistical Theory and Related Fields* (Sept. 2021), pp. 1–1. DOI: 10.1080/24754269.2021.1980261.

[33] Thomas P. Trappenberg. *Neural networks and Keras.* Oxford University Press, 2019. ISBN: 978-0-19-188387-3.

[34] Tomas Vantuch and Ivan Zelinka. *Evolutionary Based ARIMA Models for Stock Price Forecasting*. Jan. 2015, pp. 239–247. ISBN: 978-3-319-10758-5. DOI: 10.1007/978-3-319-10759-2_25.

[35] Yan Wang and Yuankai Guo. "Forecasting method of stock market volatility in time series data based on mixed model of ARIMA and XGBoost". In: *China Communications* 17.3 (2020), pp. 205–221. DOI: 10.23919/JCC.2020.03.017.

[36] *Yahoo Finanace, Business Finance, Stock Market, Quotes, News*. URL: https://finance.yahoo.com/.

[37] Roslindar Yaziz Siti, Hura Ahmad Maizah, Roslinazairimah Zakaria, and Azlinna Azizan Noor. "Modelling gold price using ARIMA–TGARCH." In: *Applied Mathematical Sciences* 10 (2016), pp. 1391–1402. ISSN: 1314-7552.

[38] Chao Yin, Charles Ward, and Sotiris Tsolacos. "Motivated monitoring: The importance of the institutional investment horizon". In: *International Review of Financial Analysis* 60.C (2018), pp. 197–212. DOI: 10.1016/j.irfa.2018.08.01.

[39] Zihao Zhang, Stefan Zohren, and Stephen Roberts. "DeepLOB: Deep Convolutional Neural Networks for Limit Order Books". In: *IEEE Transactions on Signal Processing* 67.11 (2019), pp. 3001–3012. DOI: 10.1109/TSP.2019.2907260.

[40] Xuening Zhu, Weining Wang, Hansheng Wang, and Wolfgang Karl Härdle. "Network quantile autoregression". In: *Journal of Econometrics* 212.1 (2019). Big Data in Dynamic Predictive Econometric Modeling, pp. 345–358. ISSN: 0304-4076. DOI: https://doi.org/10.1016/j.jeconom.2019.04.034.

[41] Yongqiong Zhu. "Stock price prediction using the RNN model". In: *Journal of Physics: Conference Series* 1650 (Oct. 2020), p. 032103. DOI: 10.1088/1742-6596/1650/3/032103.