Stock Market Price Prediction Using Time Series Method

By

HO YUNG HARNG

FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY

TUNKU ABDUL RAHMAN UNIVERSITY OF MANAGEMENT AND TECHNOLOGY KUALA LUMPUR

ACADEMIC YEAR 2022/2023

Stock Market Price Prediction Using Time Series Method

By

HO YUNG HARNG

Supervisor: Ms Yap Saw Teng

A project report submitted to the
Faculty of Computing and Information Technology
in partial fulfillment of the requirement for the
Bachelor of Science (Hons.)
Management Mathematics with Computing
Tunku Abdul Rahman University of Management and Technology

Department of Mathematical and Data Science
Faculty of Computing and Information Technology
Tunku Abdul Rahman University of Management and Technology
Kuala Lumpur
2022/2023

DECLARATION

The project report submitted herewith is a result of my own efforts in totality and in every aspects of the works. All information that has been obtained from other sources had been fully acknowledged. I understand that any plagiarism, cheating or collusion of any sorts constitutes a breach of Tunku Abdul Rahman University of Management and Technology rules and regulations and would be subjected to disciplinary actions.

Signature	:	Yung
Name	:	Ho Yung Harng
ID No.	:	21WMR05068
Date	:	18/3/2023

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"Stock Market Price Prediction using Time Series Method"** was prepared by **HO YUNG HARNG** and has met the required standard for submission in partial fulfillment of the requirements for the award of Bachelor of Science (Hons.) Management Mathematics with Computing at Tunku Abdul Rahman University of Management and Technology.

Approved by,

Signature:	[illegible]	**Signature:**	[illegible]
Supervisor:	Ms. Yap Saw Teng	**Moderator:**	Dr. Chin Wan Yoke
Date:	11/5/2023	**Date:**	11/5/2023

ACKNOWLEDGEMENTS

I am grateful to everyone who contributed to the successful completion of this project. I would like to express my appreciation to my research supervisor, Ms. Yap Saw Teng, for providing invaluable advice, guidance, and being patient throughout the development of the research. I also want to thank my project supervisor, Ms. Yap Sew Teng, for providing constant support, guidance, and encouragement during the project's completion. Furthermore, I would like to acknowledge Ms. Yap Sew Teng for sharing her ideas and extensive knowledge in the field, and express my heartfelt appreciation.

Next, I also want to thank my project moderator, Dr. Chin Wan Yoke for some coding techniques guidance during the project's completion.

Lastly, I would like to thank Tunku Abdul Rahman University College and the Faculty of Computing and Information Technology (FOCS) for their efforts in conducting this Final Year Project which benefits the students in different aspects. Although challenging, I really enjoyed the process of doing research and completing this project. Also, I learned a lot of Time Series Analysis and Deep Learning Model.

ABSTRACT

In this project, we are studying the behaviour of stock market prices using some statistical methods and deep learning method. Use time series analysis such as ARIMA model and deep learning such as LSTM model. This report consists of five chapters. In Chapter 1, we discuss the economic crisis problem, and then explain how ARIMA models and LSTMs work. It also introduces that some investors will be facing the risk of lose money on the investment of stocks. The forecasting method can help them to reduce the risk. In Chapter 1.2, we will discuss the objective and project scope of this final year project.

In Chapter 2, we will discuss the at journals and research papers related to the project. After, we have studied all the journals and papers, and we should make some conclusions and conclusions about it. In Chapter 2.1, we applied some method listed in the journals related to ARIMA analysis. In Chapter 2.2 and 2.3, we will apply some methods found in the journals related to LSTM model.

In Chapter 3, we will study methods for ARIMA models and LSTM models. In this chapter, we will explain the mathematical equations in ARIMA, LSTM, and statistical precision formulas found from the journals and the websites.

In Chapter 4, we will present the results of what we did in Chapter 3. In Chapter 4, first we will run the ARIMA model. The first step is to run the stationary forecasting analysis. After running the analysis, we run ACF plot and PACF plot to determine the optimal ARIMA parameters. After that, we choose the best parameters therefore we can try to run the ARIMA model to form the model. Second model, we will run the LSTM model. In the first step, we slip the test and training dataset. After that, we can train the LSTM model in batch size and epochs. After everything is done, we try to run the accuracy to check which model is the best model and perform the future predictions. Finally, in Chapter 5 we summarize and suggest some areas for future improvement.

Keywords: LSTM, Time Series, ARIMA, Stock Price Prediction, Deep learning

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

Nowadays, the lack of global financial resources and a general decline in equity values caused by the world economic crisis is hardly breaking news. Stock Market also is a marketplace to allow every industry or individual personal for the interchange of corporate stock acquisitions and sales that is seamless. Besides that, predicting the future stock market price seems to be an important skill that everyone would like to have, especially when it brings some rewards and benefits. As a result, the Time Series Method might be used in the stock market to reduce the risk of loss and to increase the profit gained.

A time series is a series of data points that occur in consecutive order over a period of time. Cross-sectional data at capture time points can be used to compare this. Investing time series records data points at regular intervals and tracks changes in selected data points over a given time period (2022). Therefore, Time series analysis uses statistical techniques to examine and simulate structured observational sequences. A process model of the system that generates data at random is produced as a result of this modelling. Most frequently, but not always, observations are sorted across time, particularly at evenly spaced intervals. A time series is only an observation or measurement manifestation of a random process in more theoretical literature (Henrik Madsen. 2022.).

Next, stock market prediction is focused on creating successful strategies on forecasting models because an important success skill in stock market forecasting is to produce optimal results and reduce errors. It is undeniable that forecasting stock indices is very challenging as market indices are often subject to changes due to the rising or falling characteristics of stock values. Investors' beliefs are being affected by volatility. To help stock market investors make smarter and more accurate investment decisions, look for more effective stock market index forecasting techniques (2022).

Besides that, the purpose of studying this topic is to predict the future price or trend of a stock, as many different analytical methods can be used such as financial instrument analysis, technical analysis, etc. Based on these analyses, most investors lost a lot of inventory losses due to misjudgement. Most of the time, they lacked the knowledge and practical forecasting tools to understand market trends and the stock market. So, seeking the right forecasting tool to predict future trends as a way to become a millionaire. Therefore, time series forecasting models came into being. The problem statement is to find out which is the best autoregressive integrated moving average ARIMA (p, d, q) model that predicts the most accurately and increases the maximum profit for the investor. In this study, try to predict the stock prices and share prices in Malaysia.

Hence, investors also can use deep learning algorithms to perform various tasks including investing, forecast management, and limiting the price of the entire portfolio. As we know, deep learning as artificial intelligence is a big trend in the world today. Based on the research is to consider the framework that forecasts stock index values using long short-term memory networks (LSTM) models, which are enhanced iterations of neural network time series architectures. Then, a certain time step is used to produce the input sequence for the LSTM model. The model considers hyperparameters such as the number of epochs, batch size, and time steps (Bhandari et al., 2022). The overfitting issue has been solved using regularisation approaches. Feed the input data into an LSTM model to predict the closing price of a stock market once the hyperparameters have been tuned. The Root Mean Square Error, Mean Absolute Percentage Error, Mean Absolute Error, and Mean Squared Error test the suggested model's quality (Bhandari et al., 2022).

1.2 Objective and Project Scope

The objective for this paper and research is to use time series autoregressive integrated moving average (ARIMA) model and deep learning model Long Short-Term Memory (LSTM) to find out which model is suitable for predicting stock market prices.

For investor decision-making, improperly executed plans will result in huge losses. On this basis, we should establish an efficient evaluation model that enables the investor to select stocks such as the Time Series autoregressive integrated moving average (ARIMA) model and LSTM model. Time series models can help investors to predict the stock market trend. For example, a dataset related to the stock market is collected and studied in order to find the best autoregressive integrated moving average (ARIMA) model for prediction and to build the ideal LSTM model.

Furthermore, time series and forecasting frequently make use of the Long Short-Term Memory (LSTM) model. The most widely used deep learning models for processing and interpreting time-series data, including time-series data, are Long Short-Term Memory (LSTM). Long Short-Term Memory (LSTM) models are made to operate with data sequences, such time series data, where the timing and order of the data are crucial. They belong to a class of recurrent neural networks (RNNs) that may predict future states by remembering past states. As a result, they are excellent candidates for use in time series analysis, where models must be able to draw conclusions from the past in order to produce precise forecasts of the future values.

CHAPTER 2

LITERATURE REVIEW

2.1 ARIMA model

The publication of Time Series Analysis: Forecasting and Control is based on Box-Jenkins (1976). Time series analysis is a new generation of forecasting tools. The most popular Box-Jenkins methodology as we know such as ARIMA methodology. A time series is a series of data points that occur in consecutive order over a period of time and many datasets is appear as the time series: daily stock market price, daily rainfall, monthly road accident. Based on those example time series analysis abounds in the engineering, economics and social science.

First, Adebiyi, Adewumi and Ayo (2022) use the ARIMA model to predict stock prices. The research of the study shows that the ARIMA model is highly competitive with current stock price forecasting methods and also has high predictive potential in short-term forecasting. The research direction of this paper is to build an ARIMA model to predict short-term stock prices. Therefore, the results using real-world data and research methods show the great potential benefit of using ARIMA (2,1,0) models to provide investors with short-term stock price forecasts and help investors make correct decisions because ARIMA (2,1,0) shows that the Bayesian information criterion (BIC), Adjusted R squared and Standard error of the regression are the best models based on the experimental results. In my opinion, using Bayesian information criterion (BIC), Adjusted R squared and Standard error of Regression is not a good way to know if it is the best model or not. We can try to determine the model using diagnostic checks.

Second, ShieldSquare Captcha (2022) it has been determined that calculating forecasting error using mean absolute deviation and mean absolute percentage error is crucial for selecting the best forecasting technique. According to the study's findings, the MAD and MAPE methods compute the data set's error resulting from the least squares method of prediction. As a result, MAD and Mean Absolute Percentage Error (MAPE) can effectively test the test and prediction findings. I believe there are several approaches that can be used to select the appropriate anticipated. For example, root mean square error (RMSE).

Third, Fatai Adewole Adebayo, Ramysamy Sivasamy and Dahud Kehinde Shangodoyin (2014) based their research on forecasting the stock market using the ARIMA model. According to the study's findings, this paper aims to find the ideal ARIMA model for the Botswana and Nigeria stock markets using diagnostic checks including AIC, BIC, HQC, RMSE and MAE to find the optimal ARIMA model. Therefore, having a reliable forecasting model to predict the stock market is a very big decision for investors. In my opinion, predictive models are sometimes not good at deciding to predict stock market prices, because there are sometimes uncertain factors in the stock market that lead to stock market crashes, so predictive models cannot predict such factors.

Lastly, Rahkmawati, Sumertajaya and Nur Aidi (2019) based on this research is to find the best ARIMA model, using selection criteria to define the ideal ARIMA Model. Based on the results of the study, the analysis model selects the characteristics of the criteria to simulate several generations of data conditional models, parameter values and variances to determine the accuracy of the ARIMA model. Therefore, in this research paper found that BIC model (Bayesian Information Criterion) it is a model with higher recognition accuracy so that on the future research can use BIC as a one of the models to find the ideal ARIMA model.

2.2 LSTM model

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture that is widely used in the field of deep learning for processing sequential data. It was developed by Hochreiter and Schmidhuber in 1997 to solve the vanishing gradient issue that prevents conventional RNNs from learning long-term relationships in sequential data

First, Moghar and Hamiche (2020) studied the deep learning language LSTM with Recurrent Neural Network based on this paper. The research of this study aims to build a model using LSTM and RNN to predict future stock market prices. In this article, you can see how the researchers trained the data and performed some experiments to choose the optimal batch size and epoch. The research direction of this paper is to run experiments to find out the optimal batch size and epochs to improve predictions. In my opinion, one can try to use hyperparameters to find the batch size and epochs, which will reduce the experiment time to calculate the batch size and number of epochs.

Beside that, Hum Nath Bhandari, Binod Rimal, Nawa Raj Pokhrel, Ramchandra Rimal, Keshab R. Dahal, Rajendra K.C. Khatri (Bhandari et al., 2022) studied LSTM models based on this paper to predict stock market prices. The research direction of this paper is to study single-layer and multi-layer LSTM. Therefore, according to the paper it is shown that single-layer LSTMs have higher prediction accuracy for multi-layers, and their performance will use RMSE, MAPE, and R to determine model fit.

2.3 Compare ARIMA model and LSTM model

First, Patil and Rajat (2021) this research papers is using ARIMA model, ARIMAX model and LSTM model to predict the stock price. The research of this study is to gradually show the modelling process of ARIMA, ARIMAX, LSTM, the transformation of data to ARIMA model, the pre-processing of data such as replacing null values, etc. The research direction of this paper is to understand all model processes.

Next, Ma, Qihang (2020) based on this research paper, ANN (artificial neural network), ARIMA model and LSTM model are studied. The study's research showed that three models were compared with the predicted results. Since ARIMA and ANN are more popular in the prediction market, the research direction of this paper mainly focuses on the LSTM model. Based on research, it is proposed that LSTM has higher accuracy in terms of prediction. In my opinion, not only does LSTM deep learning have high prediction accuracy, but also other deep learning such as Neural-Prophet.

Last, Ruochen Xiao, Yingying Feng, Lei Yan, Yihan Ma (n.d. (Xiao et al., n.d.) based on this research is to study ARIMA model and LSTM for stock price prediction Model. The research in this paper is to use MAE, MSE and RMSE to analyse the performance of stock price prediction. The research direction of this paper is to use the ARIMA and LSTM model to apply to the experiment of the stock of the financial Yahoo website company. Therefore, this paper uses the data of the training set and the test set Splitting is done for forecasting. For both models, ARIMA and LSTM use 60 days to predict the next day's price. The results show that LSTM's results are more accurate than ARIMA, but in the last ARIMA is more convenient to apply because LSTM requires some time to run batch size and epoch.

CHAPTER 3

METHODOLOGY

3.1 Data Description

The company selected for this paper is Top Glove Corp Bhd (7113.KL), which is publicly available on the Yahoo Finance website. The reason to study this company for the research is because this company is a glove factory company that manufactures medical gloves or any kind of gloves. The demand for the gloves related product is huge in the recent pandemic.

The dataset consists of 6 independent variables, 1 dependent variable. In addition, Python Jupyter is also used for generating results and building an ARIMA model and LSTM model. The dataset which consists of 6 features are displayed as follows: The date is used as the independent variable, adjusted closing price is used as the dependent variable.

Keyword
• Date - Date • Open Price - Open • Higher Price - High • Low Price - Low • Closing Price - Close • Adjusted Closing Price - Adj Close • Volume - volume

3.2 Flow Chart

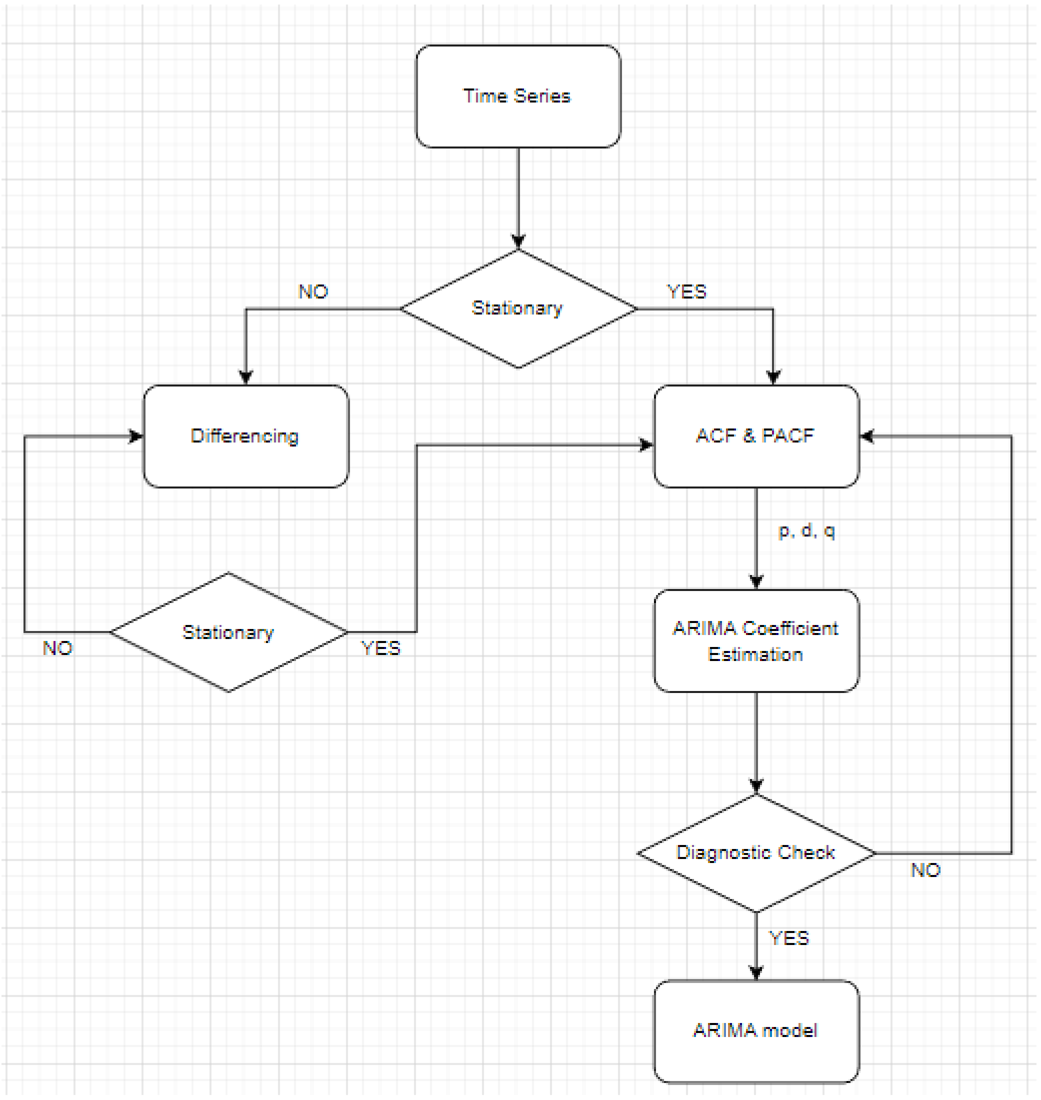


Figure 1 ARIMA Flow Chart

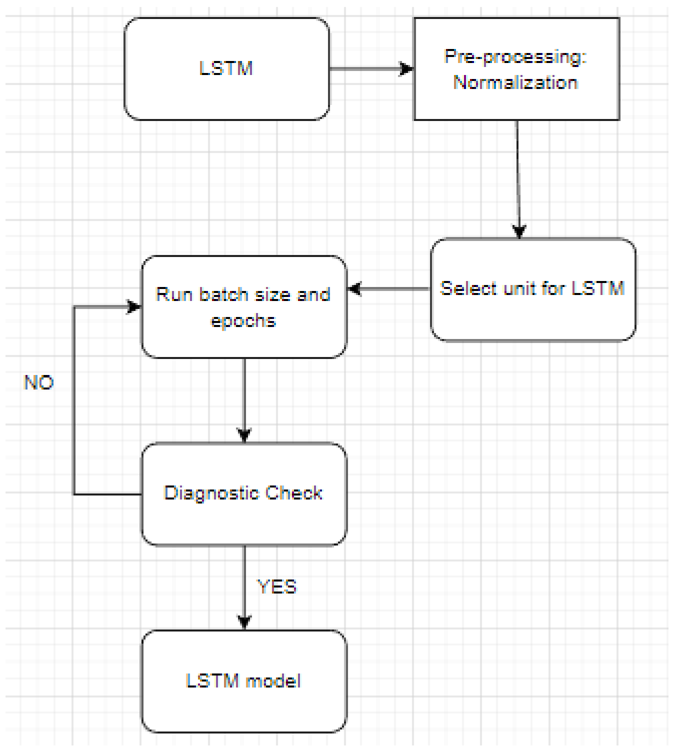


Figure 2 LSTM Flow Chart

3.3 ARIMA

ARIMA is a one statistical analysis model called an autoregressive integrated moving average time series data is used to analyse datasets and forecast trends. ARIMA is use statistical model to forecasts future values using by data from the past, it is said to be autoregressive. For instance, an ARIMA model might forecast future stock prices based on historical performance or earnings for a corporation based on historical periods. ARIMA models have three parameters, such as AR(p) process, I(d) process and MA(q) process. Finally, the model will become ARIMA (p, d, q). The problem of the world is to solve poverty. This research aims to escape poverty and learn to create suitable models to predict stock prices.

3.3.1 Stationary Test: Dickey-Fuller

First, check the dataset. It is stationary. Therefore, will use the Dickey-Fuller Test to check the stationary

$$y_t = \alpha + \beta_t + \phi y + e_t$$

which can be write as $\Delta y_t = y_t - y_{t-1} = \alpha + \beta t + \gamma y_{t-1} + e_t$

h_0: β = 0 (equivalent to φ = 1)

h_1: β < 0 (equivalent to φ < 1)

If the p-value is less than 0.05, reject null hypothesis h_0, it means data is stationary.
If the p-value is more than 0.05, fail to reject the null hypothesis h_0, it means data is not stationary.

3.3.2 Model Identification

Autocorrelation function (ACF) and partial autocorrelation function (PACF) is a tool for order of AR or MA model because the plot Autocorrelation function (ACF) and partial autocorrelation function (PACF) is the faster way to find the order of AR and MA model without to try and error. The ACF measures the correlation between a time series and its lagged values. Specifically, it computes the correlation coefficient between the series and itself, shifted by a specified number of time units (the lag). The ACF can be used to identify the presence of serial correlation, which is the tendency of a time series to exhibit dependence between adjacent observations. If the ACF shows a high degree of correlation at certain lags, it suggests that the series has some degree of persistence or memory.

Both the ACF and PACF are typically graphed as functions of the lag. A plot of the ACF typically shows the correlation coefficients as a function of the lag, while a plot of the PACF shows the partial correlation coefficients as a function of the lag. These plots can be used to identify the presence and order of different types of time series models, including AR, moving average (MA), and autoregressive integrated moving average (ARIMA) models. In research papers, the ACF and PACF can be used to analyse the serial dependence of time series data and to inform the selection of appropriate models for forecasting or modelling purposes.

ACF formula can defined as the equation:

$$\widehat{p_J} = \frac{\sum_{i=1}^{n-k}(x_i - \bar{x})(x_{i+k} - \bar{x})}{\sum_{i=1}^{n}(xi - \bar{x})^2}$$

PACF formula can defined as the equation:

$$r_{ii} = r_1 \quad if\ i = 1$$

$$r_{ii} = \frac{r_i - \sum_{k=1}^{k-1} r_i - 1, k^r i - k}{1 - \sum_{j=1}^{i-1} r_{i-1} r_j}\ , \qquad if\ k = 2,3, \ldots ..$$

Where $r_{ij} = r_{i-1,j} - r_{ii} r_{i-1,j-i}$ for $j = 1,2,3, \ldots, k-1$

The partial autocorrelations for a white noise series should all be near to zero, much as the ACF.

Model	ACF	PACF
AR (p)	Attenuation tends to zero	Cut off after lag p
MA (q)	Cut off after lag p	Attenuation tends to zero

Table 1 Theoretical for the ACF and PACF

3.3.3 Auto-Regression (AR)

A time series modelling technique called auto regression (AR) simulates the relationship between a variable and its own lagged data. In other words, an AR model forecasts a time series' future values based on its past values. The method is predicated on the idea that, with some amount of noise, the future values of a time series are a linear function of those values in the past. An AR(p) model is an Auto-regressive model where specific lagged values of yt are used as predictors. Lag is where the results of one time period affect subsequent time periods.

AR(p) model is defined as the equation:

$$y_t = c + \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + \ldots + \varphi_p y_{t-p} + \varepsilon_t$$

$$y_t = c + \sum_{i=1}^{p} \varphi_i y_{t-i} + \varepsilon_t$$

Where $y_{t-1}, y_{t-2}, y_{t-i}$ $are\ the\ past\ series\ values(Lags)$,and the white noise.
Where ε_t is the error term at the time t.
Thus, AR (1) process can be defined by $Y_t = \phi_1 x_{t-1} + \varepsilon_t$

3.3.4 Integration(I)

If the time series needs to be differentiated once in order to become stationary, it is of the first order (i.e. I (1)).
I(d) model is defined as the equation:

$$\nabla Xt = X_t - X_{t-1} \quad (1)$$

Thus, will become a new time series ∇X

$$\nabla 2Xt = \nabla(\nabla X_t) = (X_t - X_{t-1}) - (X_{t-1} - X_{t-2}) = X_t - 2X_{t-1} + X_{t-2} \quad (2)$$

If the original time series is not stationery. Therefore, we can look at the first order (1), or second order (2). After that, if the time series are stationary we can look at the ARIMA model in a different process and usually d =1 or d =2, is sufficient.

3.3.5 Moving-Average (MA)

A time series modelling technique called the moving average (MA) determines the average of a set of observations made over a predetermined amount of time. In order to smooth out short-term fluctuations and expose longer-term trends, the MA model, as its name suggests, takes a rolling average of previous values of a time series. The MA (moving-average) model is ordered by the q such as MA(q). Model the current value as a linear combination of the first q error terms.

MA(q) model is defined as the equation:

$$y_t = \theta_1\varepsilon_{t-1} + \theta_2\varepsilon_{t-2} + \ldots + \theta_q\varepsilon_{t-q} + \varepsilon_t$$

$$y_t = \sum_{i=1}^{q} \theta_i\varepsilon_{t-i} + \varepsilon_t$$

Where $\varepsilon_t \sim wn(0, {\sigma_{vv}}^2)$, and $\theta_1, \theta_2, \ldots, \theta_q (\theta_q \neq 0)$ are parameters. Therefore, the normal distribution with mean 0 and variance ${\sigma_{vv}}^2$. Thus, MA (1) process can be defined by
$Y_t = \theta_1\varepsilon_{t-1} + \varepsilon_t$.

3.3.6 Autoregressive integrated moving average Model (ARIMA)

ARIMA (Autoregressive Integrated Moving Average) is a time series modelling technique that combines autoregression (AR) and moving average (MA) models with differencing to account for non-stationarity in a time series. Similar to an autoregressive (AR) model, the AR component of an ARIMA model uses the past values of a time series to forecast its future values. The MA component, on the other hand, simulates a moving average (MA) model for the relationship between the time series current value and its historical error terms. By differencing the series until it becomes stationary, the integral (I) component of ARIMA is utilised to make a time series stationary.

Finally, an ARIMA (p, d, q) has the combined form:

$$\Delta^D y_t = c + \phi_1\Delta^D Y_{t-1} + \ldots + \phi_p\Delta^D y_{t-p} + \varepsilon_t + \theta_1\varepsilon_{t-1} + \ldots + \theta_q\varepsilon_{t-q}$$

where $\Delta^D y_t$ represent a Dth differenced by the time series, and the t is an uncorrelated innovation process with the mean zero. Therefore, we can write as

$$X_t = \Delta^D y_t \ = c + \varepsilon_t + \sum_{i=1}^{p} \phi_i \Delta^D Y_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}$$

There are three terms in the equation:

AR: Auto Regression: The regression is done on the time series using its prior value such as $y(t-1)\,, y(t-2)$ etc. The order of the lag is called as **p**.

I: Integration: The time series is made stationary using differencing. The order of the difference is denoted as **d**.

MA: Regression on a time series using residuals from past observation $error\ \varepsilon(t-1), error\ \varepsilon(t-2)$. The order of the error lag is called **q**.

3.4 Diagnostic Check

Diagnostic checks are important in time series modelling to ensure that the model assumptions are met and that the model is performing well in predicting future values. There are several diagnostic checks that can be performed on a time series model

3.4.1 Root Mean Square Error

Root Mean Square Error (RMSE) is a measure of the accuracy of a time series forecasting model. It is calculated as the square root of the average of the squared differences between the predicted and actual values of the time series over a given period of time. RMSE smaller values indicate the better fit.

$$RMSE \ = \sqrt{\frac{\varepsilon_1^2 + \varepsilon_2^2 + \ldots + \varepsilon_n^2}{N}} \ , \varepsilon_i \ = y_i - y_i$$

where y_i is the actual value of the time series at time n, Yi is the predicted value of the time series at time n, and N is the total number of observations in the test set.

3.4.2 Mean Absolute Percentage Error

MAPE stands for Mean Absolute Percentage Error, and it is a measure of the accuracy of a time series forecasting model. MAPE measures the average absolute percentage difference between the predicted and actual values of a time series over a given period of time. MAPE smaller values indicate the better fit.

$$MAPE \ = \ (\frac{1}{n}\sum_{i=1}^{n}\frac{|Actual_i - Forecast_i|}{|Actual_I|}) * 100$$

3.4.3 Mean Absolute Error

Mean absolute error (MAE) is used to evaluate the measurement of the performance of the regression model. It measures the absolute difference between the predictive value and the actual value.

$$MAE \ = \ (\frac{1}{n}\sum_{i=1}^{n}|Actual_i - Forecast_i|)$$

The lower the MAE, the better the model's performance. MAE is relatively easy to explain, because it represents the average amplitude of the model's mistakes.

3.4.4 Ljung-Box Statistic

The Ljung-Box statistic is a statistical test used to determine whether a time series is autocorrelated. Autocorrelation is the degree to which a time series is correlated with its own past values at different lags. If a time series is highly autocorrelated, it means that the past values of the time series can help predict its future values.

Hypotheses：

$$H_0: The\ residuals\ are\ independently\ distributed.$$
$$H_A: The\ residuals\ are\ not\ independently\ distributed.$$

Test statistic:

$$Q = n(n+2)\frac{\Sigma p_k^2}{(n-k)}$$

Where:

$n\ =\ sample\ size$

$\Sigma\ = sum\ of\ 1\ to\ h, where\ h\ is\ the\ number\ of\ lags$

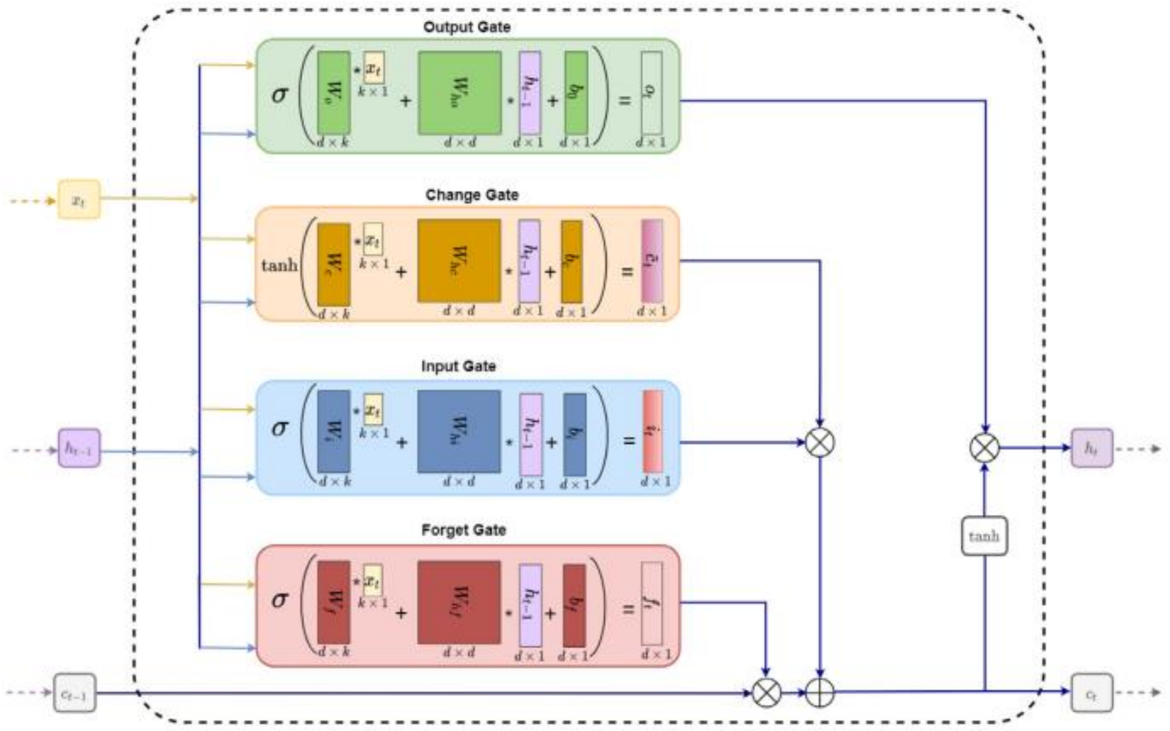
$P_K\ =\ sample\ autocorrelation\ at\ lag\ k$

Rejection Region:

Test statistic Q is following a chi-square distribution with h, called as $Q \sim \chi^2(h)$. Reject the null hypothesis H_A: The residuals are not independently distributed if $Q > \chi^2_{1-\alpha,h}$

3.5 LSTM (Long Short-Term Memory)

Recurrent neural networks (RNNs) of the LSTM (Long Short-Term Memory) type are frequently employed in applications for time series analysis. The ability of LSTMs to learn and remember long-term dependencies and patterns in time-series or sequential data is their primary advantage over conventional RNNs. LSTM models are particularly good at modelling long-term dependencies in time series data because of their distinctive architecture, which enables them to selectively forget or remember information at each time step. Input gates, forget gates, change gate, and output gates are the four types of gates used in LSTM models (Moghar and Hamiche, 2020). The input gate selects the data that is pertinent for the current time step and permits it to pass through. The forget gate decides which past information should be forgotten.



For given by the input sequence $\{x_1, x_2, \dots, x_n\}, x_t \in R^{k\times 1}$ is to represent the sequence by time t. Afterwards, for memory cell c_t is used to update the information it uses through by three gates, e.g. input gate i_t, forget gate f_t, and change gate $\hat{c}_t$. Hidden state h_t is updated using by the output gate o_t and memory cell c_t(Bhandari et al., 2022). Respective gates and layer will present as:

$$i_t = \sigma(W_i x_t + W_{hi} h_{t-1} + b_i)$$

$$h_t = \sigma(W_f x_t + W_{hf} h_{t-1} + b_f)$$

$$o_t = \sigma(W_o x_t + W_{ho} h_{t-1} + b_o)$$

$$\hat{c}_t = tanh(W_c x_t + W_{hc} h_{t-1} + b_c)$$

$$c_t = f_t \, x \, c_{t-1} + i_t \, x \, \hat{c}_t$$

$$h_t = o_t \, x \tanh(c_t)$$

Min – max Normalization:

$$\text{z} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Where $\sigma = \frac{1}{1+e^{-t}}$ and tanh is to stand for sigmoid and hyperbolic tangent function, the $\otimes$ is to represent product element-wise, therefore $W \in R^{d\times k}$, $W_h \in R^{d\times d}$ is matrices for the weight, and $b \in R^{d\times 1}$ are the vectors bias (Bhandari et al., 2022).

CHAPTER 4

RESULTS AND DISCUSSION

Based on the approach presented in previous Chapter, this chapter will present the final results step by step.

4.1 Visualise the Top Glove Stock Market Price

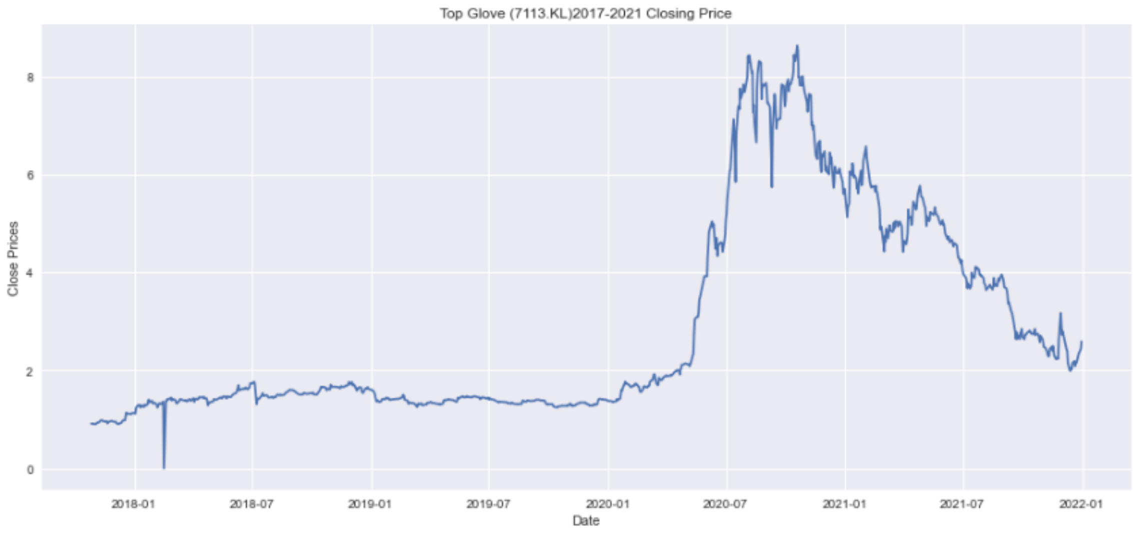
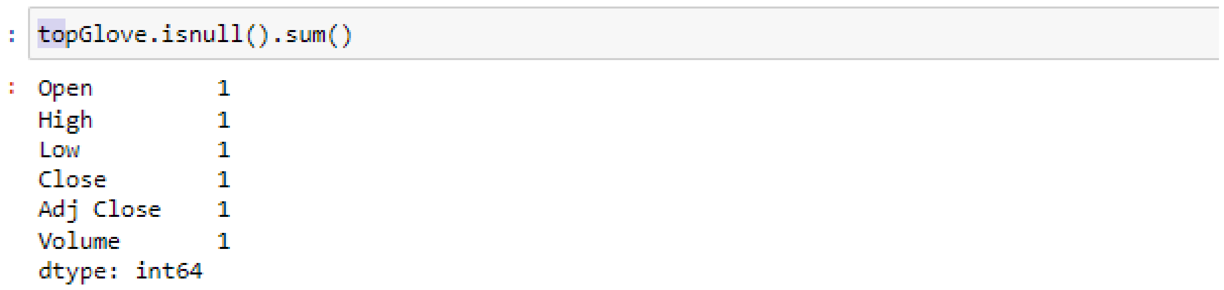


Figure 3 Top Glove Adj Close Price (2017/01/01 to 2021/12/31)

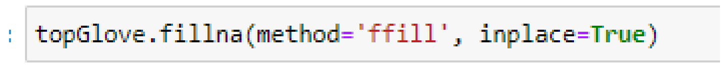
Figure 4 shows the original chart of the daily return adjusted closing price of Top Glove Corporation Berhad (7113.KL). Based on this graph, found that the movement of the price in this graph is not stationary. To verify the stationary, we need to use the adjusted dickey fuller statistical method to test whether it is stationary.

4.2 Data Cleaning



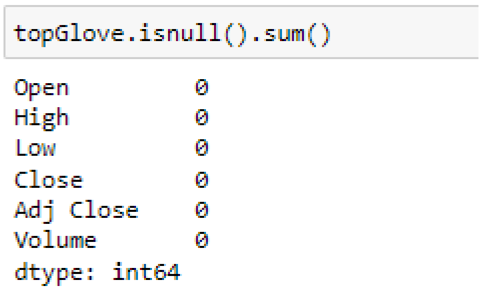
```
: topGlove.isnull().sum()

: Open         1
  High         1
  Low          1
  Close        1
  Adj Close    1
  Volume       1
  dtype: int64
```

Figure 4 Check Missing Value



```
: topGlove.fillna(method='ffill', inplace=True)
```

Figure 5 Replace Missing Value



```
topGlove.isnull().sum()

Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

Figure 6 Check Missing Value After Replace

I will be using Jupyter. isnull(). sum() checks the dataset, it has missing values on the dataset or not missing values. Figure 4 shows that we have one missing value on the dataset. Therefore, we must remove the missing values because when having a missing value on the dataset are not removed. It will affect our future predictions on the model. In this section figure 5, I'll use "ffill" to replace the missing values, because the dataset is daily share price reports often have missing values due to weekends, holidays, or data collection errors. When analysing stock price data for precise calculations such as calculating returns or forecasting future prices, it is crucial to have a complete time series with no missing values. After we do the data cleaning we can start to analyse stock price forecasting.

4.3 Stock Price Forecasting

4.3.1 Decompose

In this section, decomposition is used to view trend and seasonality in time series data. Decomposition is to consider level component, trend component, seasonal component and residual component. Figure 7 shows an exploded view in below.

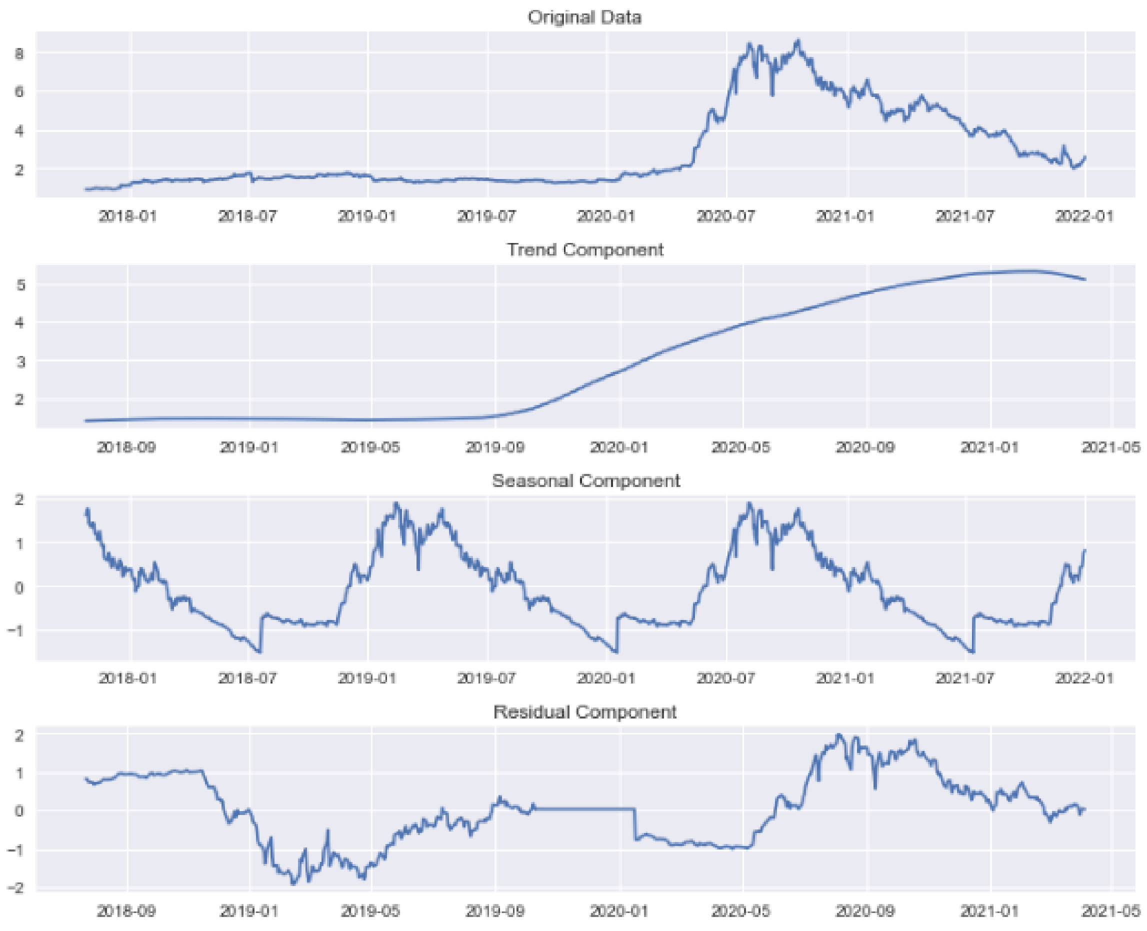
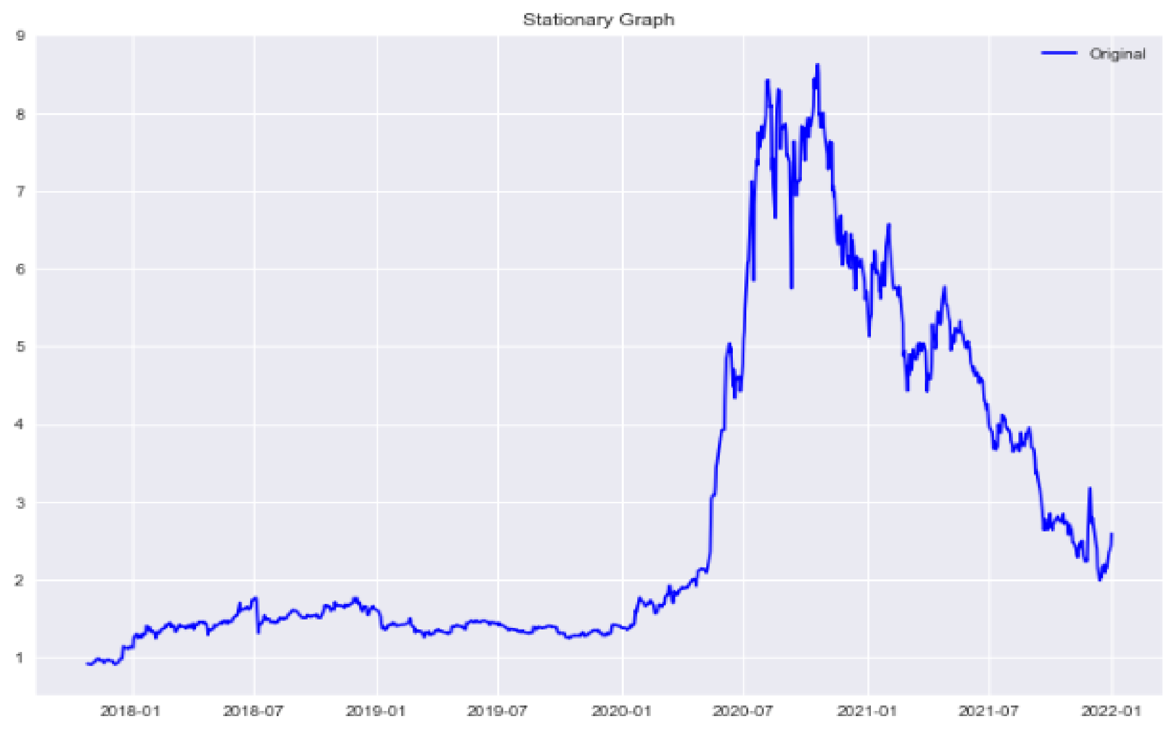


Figure 7 Analysis Seasonal Decompose

Figure 7 shows that the raw data is not a seasonal dataset, the trend component of the data is incremental, and the seasonal component is additive.

4.3.2 Stationary Test: Augmented Dickey-Fuller Test (ADF Test)



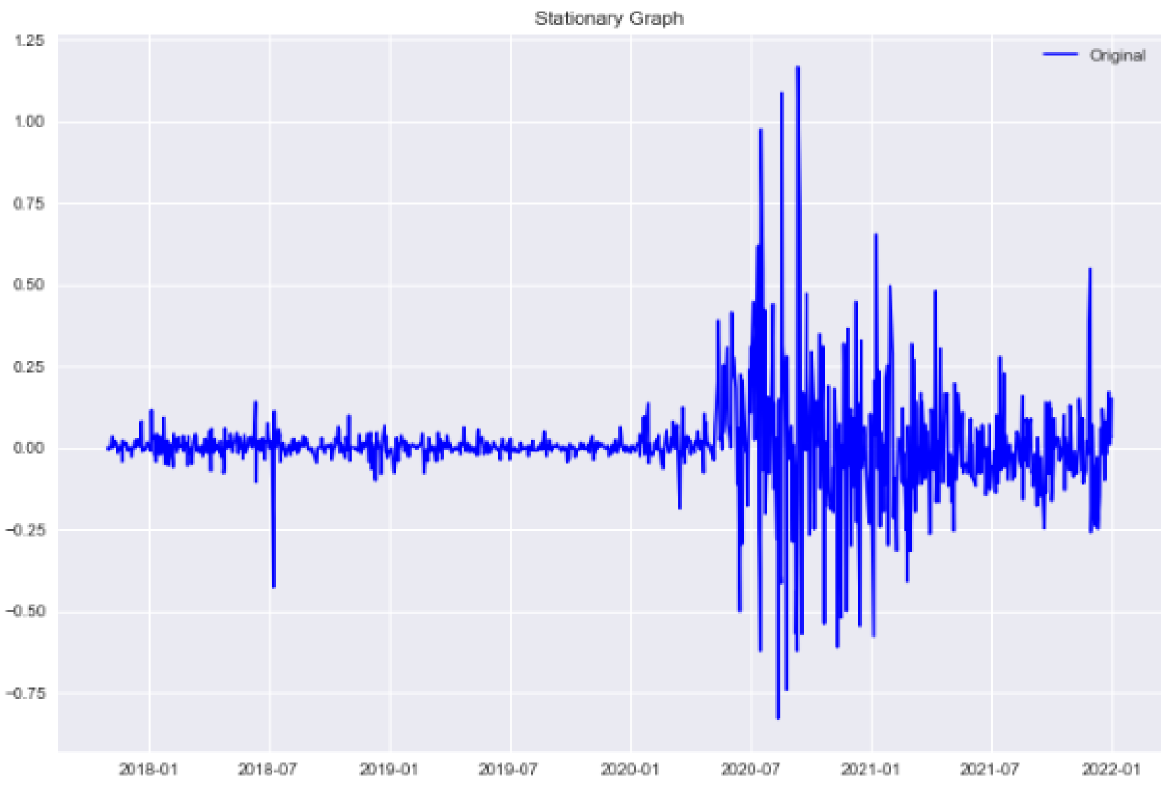
```
Results of Dickey-Fuller Test:
p-value = 0.5898. The series is likely non-stationary.
Test Statistic                   -1.384358
p-value                           0.589752
#Lags Used                       19.000000
Number of Observations Used    1008.000000
Critical Value (1%)              -3.436854
Critical Value (5%)              -2.864412
Critical Value (10%)             -2.568299
```

Figure 8 Dickey-Fuller Test

Hypothesis Testing: $\alpha = 0.05$

$$H_0: data\ is\ not\ stationary.$$
$$H_1: data\ is\ stationary\ .$$

The Figure 8 is showing the Results of the Dickey-Fuller Test. The result shows that p-value is 0.5898 that’s mean the p-value is more than 0.05, so failed to reject H_0 and conclude that the series is non-stationary. Since, the data is not a stationary, therefore differencing method has to be carried out. First, the trial is to use first differencing.



```
Results of Dickey-Fuller Test:
p-value = 0.0000. The series is likely stationary.
Test Statistic                 -6.582493e+00
p-value                         7.453808e-09
#Lags Used                      1.800000e+01
Number of Observations Used     1.008000e+03
Critical Value (1%)            -3.436854e+00
Critical Value (5%)            -2.864412e+00
Critical Value (10%)           -2.568299e+00
```

Figure 9 First Difference

Figure 9 shows the results after our first differencing. The results show a p-value of 7.453808 e-09 which is less than 0.05. Therefore, H_0 is rejected and conclude that the series is stationar y. Since the series is a stationary, therefore can start to define the model identification. Howev er, the variance is not stationary as the senses is fluctuate. Figure 4 shows that the variance is not stable from 2020/06 to 2021/07. This is because the covid-19 pandemic has a great impact on stock prices. Because the market is demanding, during the MCO and EMCO period, the sh are price rose from RM2 per share to RM8 per share.

4.4 ARIMA model

4.4.1 Model Identification- ACF Plot and PACF Plot

When data is stationary, the ARIMA parameter for p, d, q can be identified using the steps in ACF and PACF.

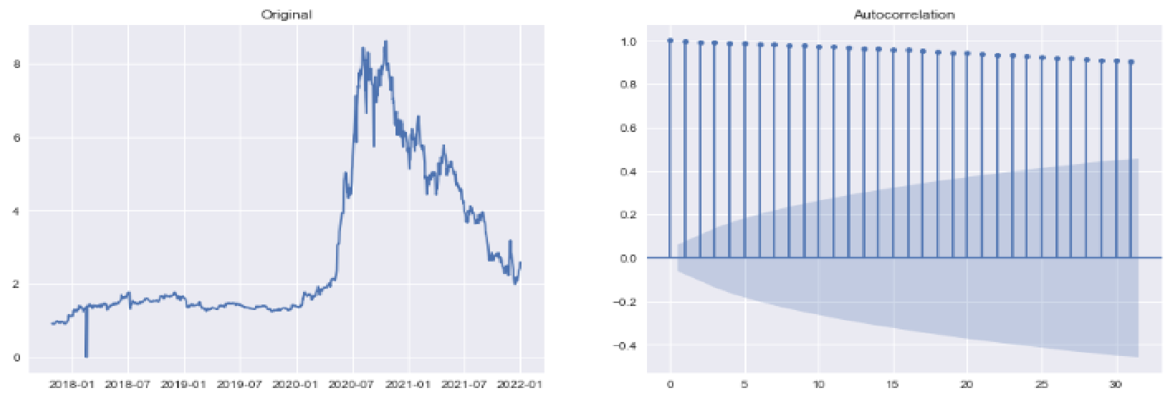


Figure 10 Original ACF Series

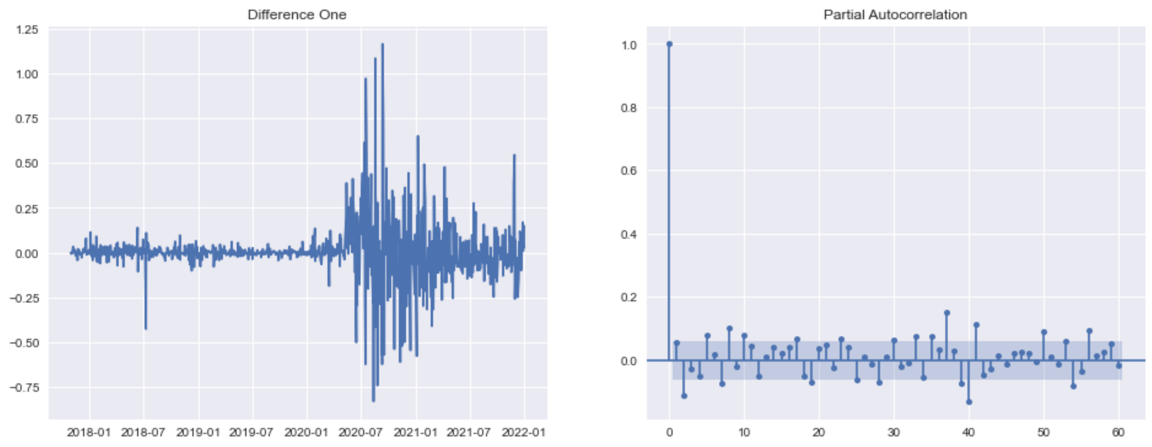


Figure 11 First Difference PACF Plot

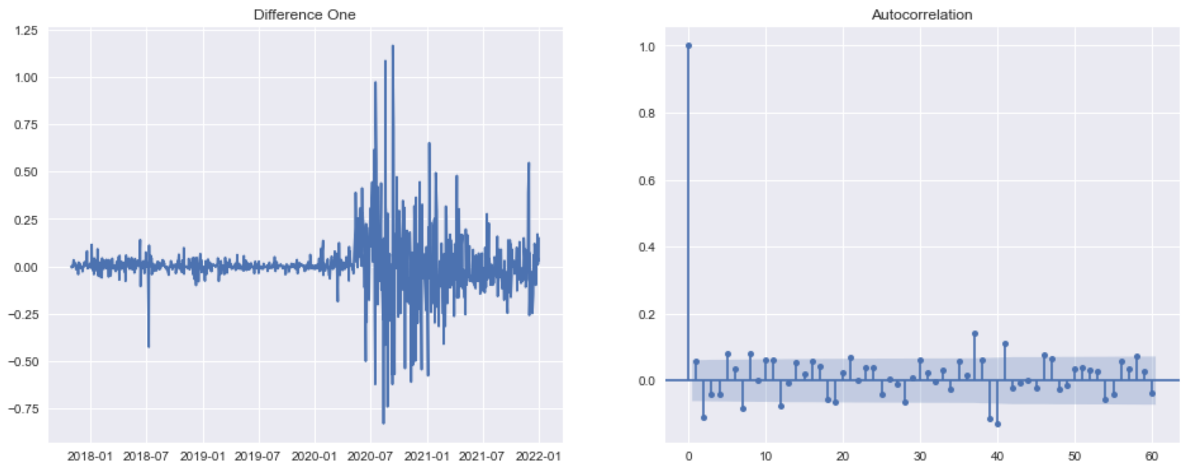


Figure 12 First Difference ACF Plot

Figure 11 and Figure 12 showing the autocorrelation and partial autocorrelation function for the first differencing.

From Figure 11 PACF plot represents the AR (p) based on the theoretical table there is cut off after lag p on lag = 2 which mean the lag 2 is well below the significant line. Whereas Figure 12 ACF plot represents the MA(q) based on the theoretical table there is cut off after lag p on lag = 2 and 5. Since, the stationary test involved only one differencing that means d will be selected as 1. Finally, we can form the ARIMA model as ARIMA (2, 1, 2) and ARIMA (2, 1, 5). ARIMA models stop at ARIMA (2, 1, 5) because total p, d and q sum up not suggest more than 8.

4.4.2 Estimation of Parameter

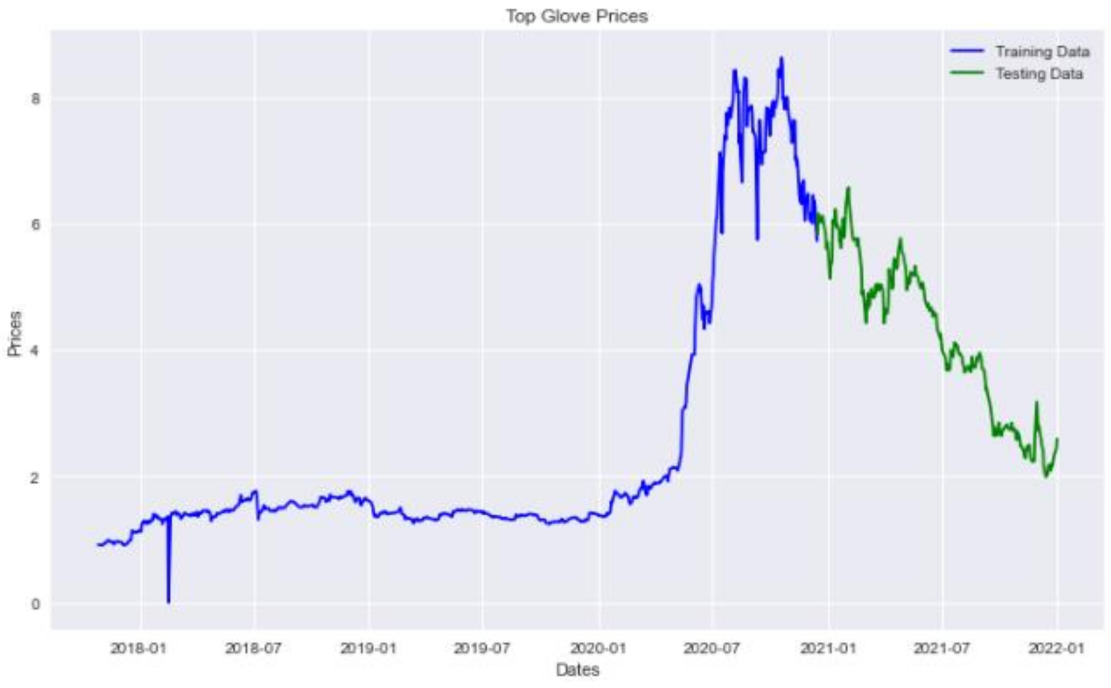


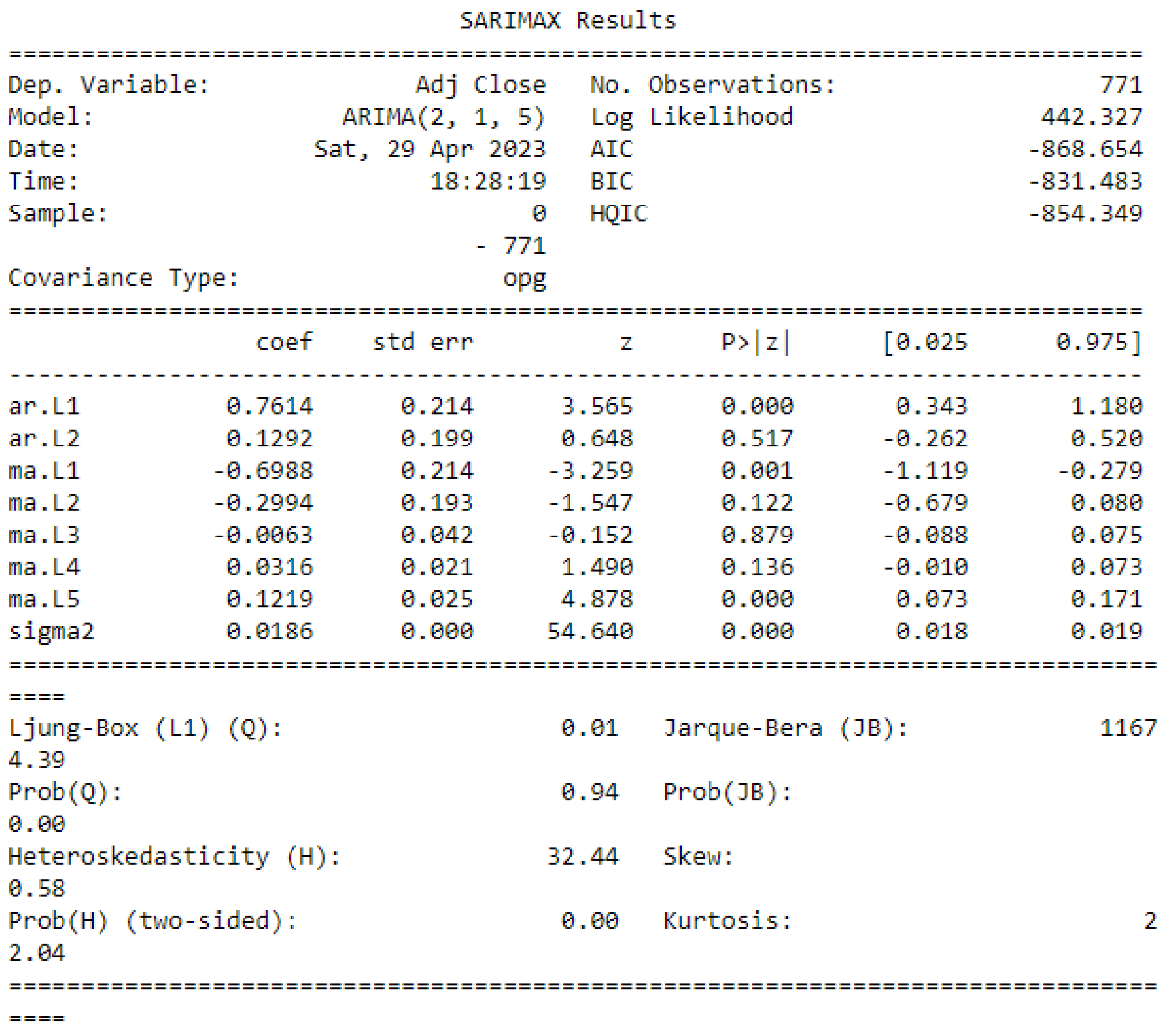
Figure 13 Graph Test and Validation

Figure 13 showing the training dataset and testing dataset. On this part, 0.75 dataset will be split into the training set observation and 0.25 will be split in test set observation for the ARIMA (2, 1, 2) and ARIMA (2, 1, 5). Next step is to try select the model using mean squared error (MSE) and Akaike information criterion (AIC).

ARIMA models	Accuracy: MSE	AIC
ARIMA (2, 1, 2)	4.17	-861.510
ARIMA (2, 1, 5)	3.56	-868.654

Table 2 Model Identification

In Table 2 showing the ARIMA (2, 1, 5) having a lower MSE and AIC with ARIMA (2, 1, 5) models. Therefore, we can estimate ARIMA (2, 1, 5) is the model using the PACF plot and ACF plot. Next step, use Auto-ARIMA to find another ARIMA model.

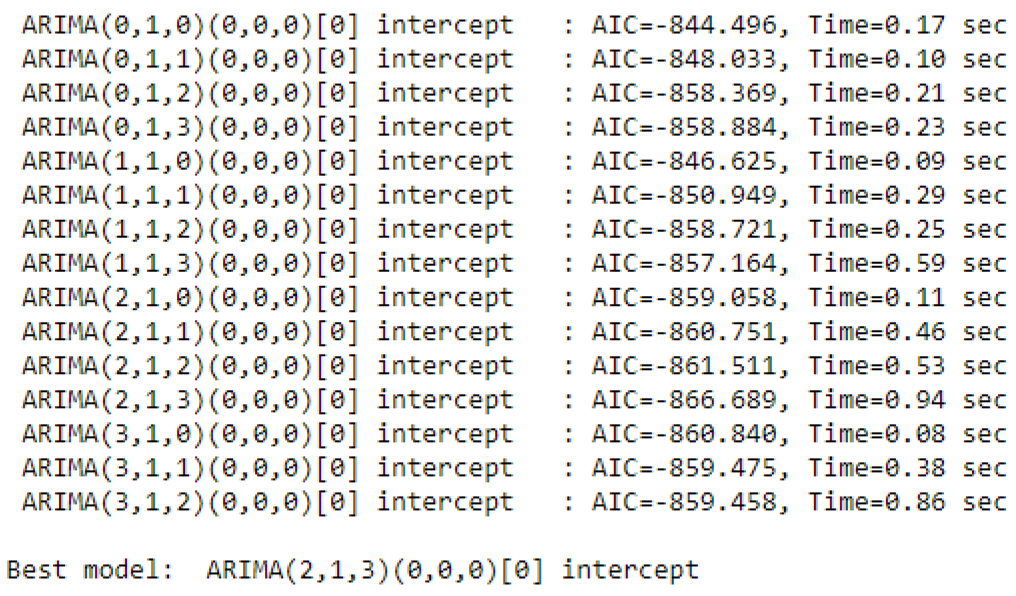


```
                               SARIMAX Results
==============================================================================
Dep. Variable:              Adj Close   No. Observations:                  771
Model:                 ARIMA(2, 1, 5)   Log Likelihood                 442.327
Date:                Sat, 29 Apr 2023   AIC                           -868.654
Time:                        18:28:19   BIC                           -831.483
Sample:                             0   HQIC                          -854.349
                                - 771
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.7614      0.214      3.565      0.000       0.343       1.180
ar.L2          0.1292      0.199      0.648      0.517      -0.262       0.520
ma.L1         -0.6988      0.214     -3.259      0.001      -1.119      -0.279
ma.L2         -0.2994      0.193     -1.547      0.122      -0.679       0.080
ma.L3         -0.0063      0.042     -0.152      0.879      -0.088       0.075
ma.L4          0.0316      0.021      1.490      0.136      -0.010       0.073
ma.L5          0.1219      0.025      4.878      0.000       0.073       0.171
sigma2         0.0186      0.000     54.640      0.000       0.018       0.019
================================================================================
====
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):              1167
4.39
Prob(Q):                              0.94   Prob(JB):
0.00
Heteroskedasticity (H):              32.44   Skew:
0.58
Prob(H) (two-sided):                  0.00   Kurtosis:                         2
2.04
================================================================================
====
```

Figure 14 Summary ARIMA (2,1,5) model

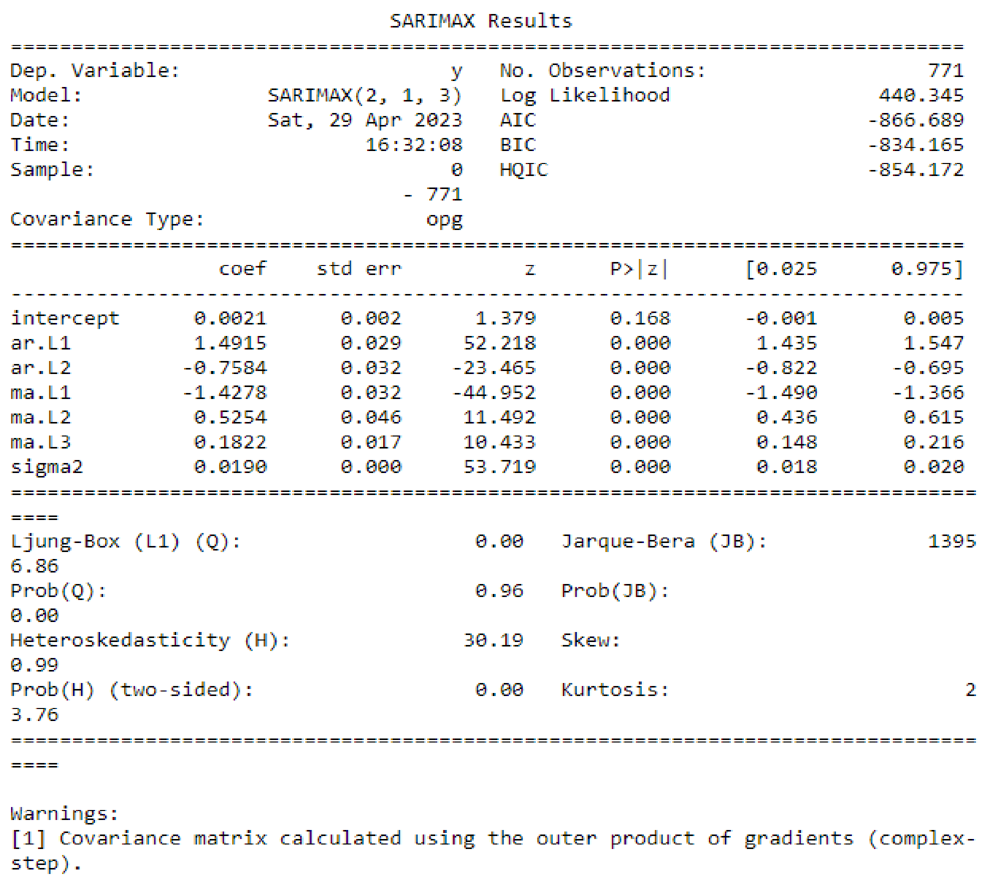
4.5 Auto-ARIMA Model

4.5.1 Estimation of Parameter on Auto-ARIMA



```
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=-844.496, Time=0.17 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=-848.033, Time=0.10 sec
 ARIMA(0,1,2)(0,0,0)[0] intercept   : AIC=-858.369, Time=0.21 sec
 ARIMA(0,1,3)(0,0,0)[0] intercept   : AIC=-858.884, Time=0.23 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=-846.625, Time=0.09 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=-850.949, Time=0.29 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=-858.721, Time=0.25 sec
 ARIMA(1,1,3)(0,0,0)[0] intercept   : AIC=-857.164, Time=0.59 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=-859.058, Time=0.11 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=-860.751, Time=0.46 sec
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=-861.511, Time=0.53 sec
 ARIMA(2,1,3)(0,0,0)[0] intercept   : AIC=-866.689, Time=0.94 sec
 ARIMA(3,1,0)(0,0,0)[0] intercept   : AIC=-860.840, Time=0.08 sec
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AIC=-859.475, Time=0.38 sec
 ARIMA(3,1,2)(0,0,0)[0] intercept   : AIC=-859.458, Time=0.86 sec

Best model:  ARIMA(2,1,3)(0,0,0)[0] intercept
```

Figure 15 Model Suggestion Using Auto-ARIMA



```
                               SARIMAX Results                                
==============================================================================
Dep. Variable:                      y   No. Observations:                  771
Model:               SARIMAX(2, 1, 3)   Log Likelihood                 440.345
Date:                Sat, 29 Apr 2023   AIC                           -866.689
Time:                        16:32:08   BIC                           -834.165
Sample:                             0   HQIC                          -854.172
                                - 771                                         
Covariance Type:                  opg                                         
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      0.0021      0.002      1.379      0.168      -0.001       0.005
ar.L1          1.4915      0.029     52.218      0.000       1.435       1.547
ar.L2         -0.7584      0.032    -23.465      0.000      -0.822      -0.695
ma.L1         -1.4278      0.032    -44.952      0.000      -1.490      -1.366
ma.L2          0.5254      0.046     11.492      0.000       0.436       0.615
ma.L3          0.1822      0.017     10.433      0.000       0.148       0.216
sigma2         0.0190      0.000     53.719      0.000       0.018       0.020
===============================================================================
====
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):               1395
6.86
Prob(Q):                              0.96   Prob(JB):                           
0.00
Heteroskedasticity (H):              30.19   Skew:                               
0.99
Prob(H) (two-sided):                  0.00   Kurtosis:                          2
3.76
===============================================================================
====

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
```

Figure 16 Summary Model using Auto-ARIMA

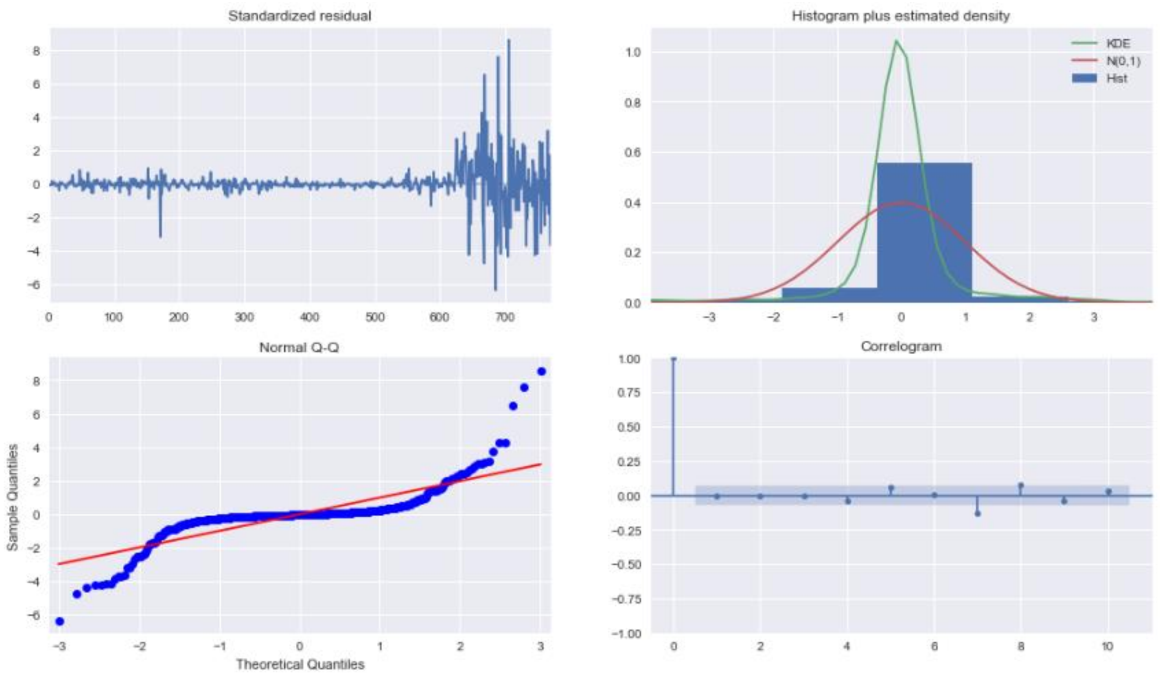


Figure 17 Plot for residual ARIMA (2, 1, 3)

Based on the Figure 15 showing above, we will use Auto-ARIMA to find out which ARIMA model is more suitable model. In figure 15 is using AIC to determine which model is the suitable model. The result shows that the best order on ARIMA is ARIMA (2, 1, 3) and we can see that the AIC value of the ARIMA (2, 1, 3) is -866.689.

4.5.2 Model Diagnostic Check - ARIMA (2,1,5)

Models	Auto-ARIMA (2,1,3)	ARIMA (2,1,5)
MSE	10.35	3.56
MAE	2.67	1.51
RMSE	3.21	1.88
MAPE	0.87	0.50

Ljung-Box		
Lag	Statistic	P-value
10	15.30641739	0.12

Table 3 Model Statistics

The Diagnostic check is to test how well for the model. First and foremost, Ljung-Box Statistics is to test the residuals whether independently or not independently. Table 3 Ljung-Box test shows that the statistic value is 15.30641739 and p-value is 0.12. Since the p-value is more than 0.05, thus to failed to reject the null hypothesis. It indicates that the model has correctly modelled the time series correlation.

Also, smaller values of MSE, MAE, RMSE, and MAPE fit the model well. In Table 3, MAPE (mean absolute percentage error) is commonly interpreted and explained in the Time series diagnostic check. In Table 3 showing that, ARIMA (2, 1, 5) has smaller accuracy value compared with Auto-ARIMA (2, 1, 3). Therefore, the ARIMA (2, 1, 5) model can be conducted in this paper.

Table 3 shows that the ARIMA (2, 1,5) MAPE is approximately 50%, indicating that the model is approximately 50% accurate in predicting the test set observations. ARIMA (2, 1,5) MSE (mean squared error) around 3.54 (average squared) between the actual value and predicted value. According by the (Lewis, C.D. (1982)), MAPE less than 10 % can be considered a good accurate forecast. Since the MAPE is more than 10% that’s mean is considered not a good accurate forecast. After completing all the diagnostic checks, we can plot the ARIMA (2, 1,5) forecasting graph to identify whether the actual price paired well with the prediction price in Figure 15 and 16.

4.5.3 Forecasting ARIMA (2,1,5)

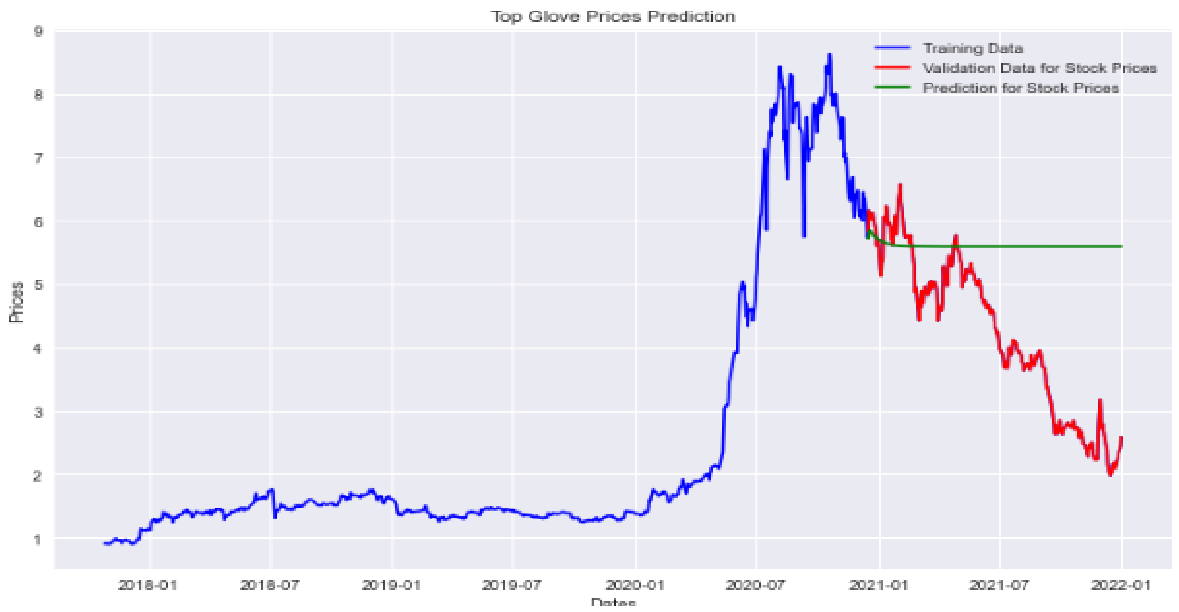


Figure 18 Forecasting ARIMA (2,1,5)

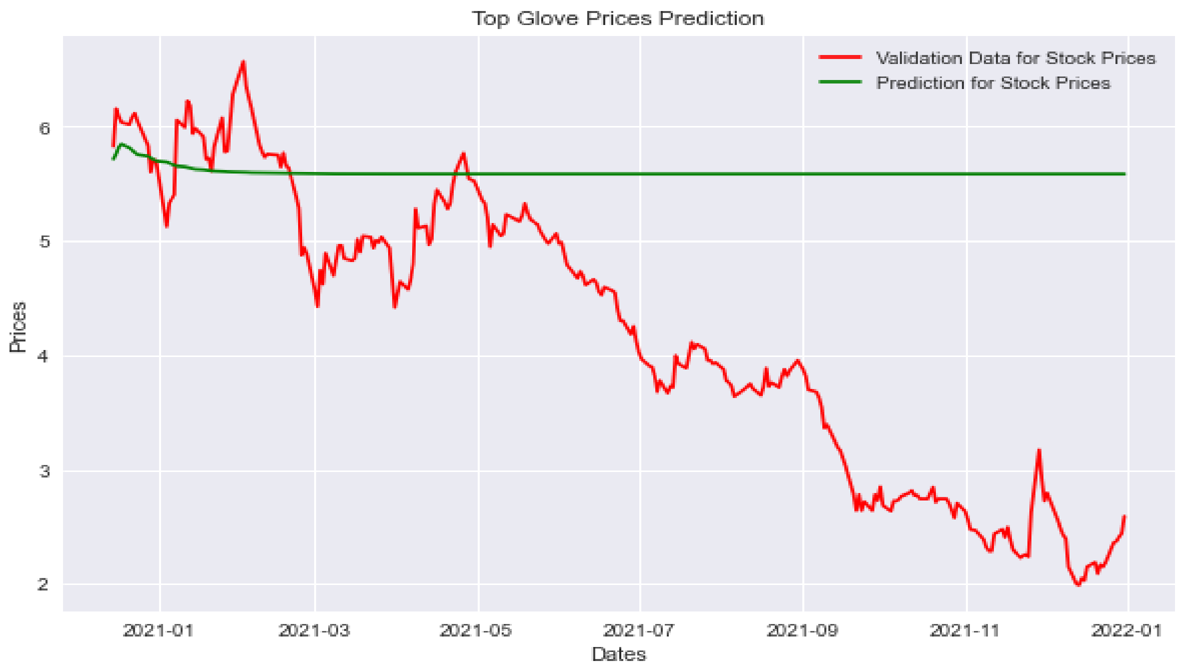


Figure 19 Forecasting ARIMA (2,1,5)- 2020/12-2021/12

Since the MAPE is more than 10% and Figure 18 and 19 show that the forecasting does not paired well with actual price. We can observe that the predicted price does not fit with the actual price. Therefore, we should try other models to improve the prediction. In this part, I will use the deep learning method in chapter 4.6.

4.6 LSTM Model

4.6.1 Data Splitting

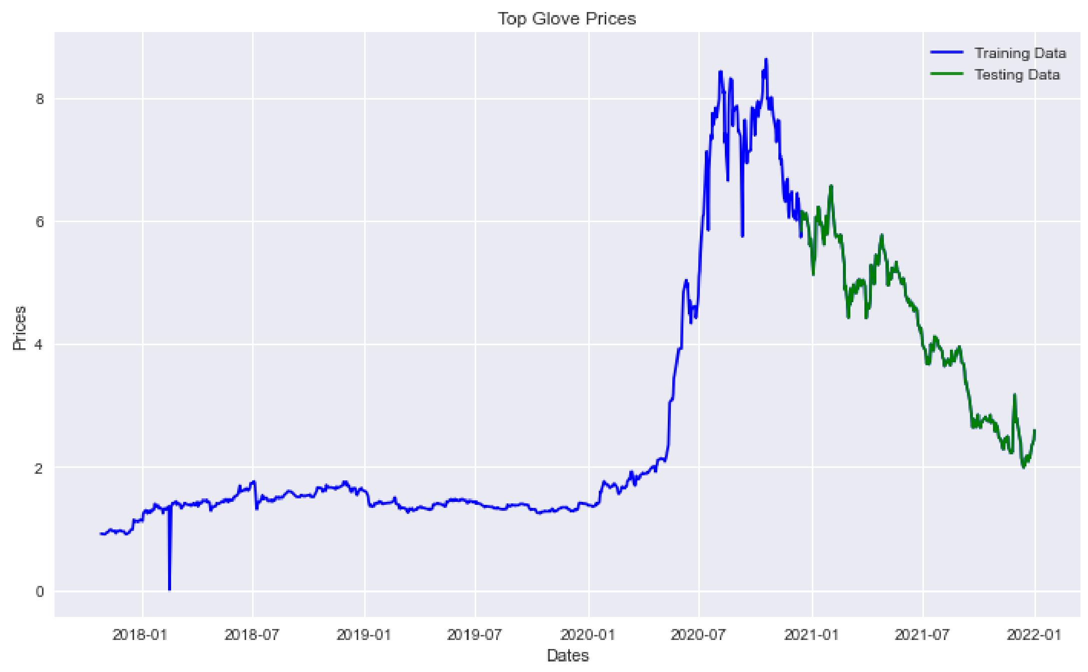
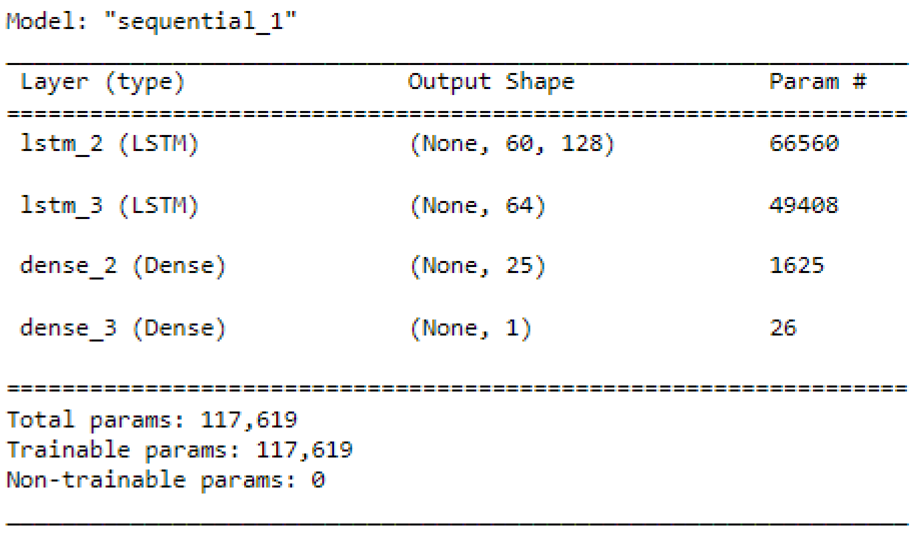


Figure 20 LSTM Graph Test and Validation

Figure 20 showing the training dataset and testing dataset. In this part, we will split the 0.75 dataset for training data to train the LSTM model. The LSTM model is used by the ‘Keras’ package in Juypter. After that, the batch size is 3 and the run epochs is 40, in this case, a batch size of 3 means that the model updates its weights after processing 3 samples at a time. Next, will run the 40 epochs, which means that it processes the entire training dataset 40 times during training.

4.6.2 Build Model Data (batch size =3, epochs =40)



```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_2 (LSTM)               (None, 60, 128)           66560

 lstm_3 (LSTM)               (None, 64)                49408

 dense_2 (Dense)             (None, 25)                1625

 dense_3 (Dense)             (None, 1)                 26

=================================================================
Total params: 117,619
Trainable params: 117,619
Non-trainable params: 0
_________________________________________________________________
```

Figure 21 LSTM model Summery

In Figure 21, The first LSTM layer, lstm_2, takes in sequences of length 60 and has 128 neurons. Therefore, the output shape of this layer is (None, 60, 128), meaning that it returns a sequence of vectors of length 128 for each input time step. This layer has 66,560 trainable parameters.

The output of the first LSTM layer is then fed into the second LSTM layer, lstm_3, which has 64 neurons and does not return sequences. Therefore, the output shape of this layer is (None, 64), which means that it outputs a single vector of length 64 that summarizes the input sequence. This layer has 49,408 trainable parameters. The output of the second LSTM layer is then fed into the first Dense layer, dense_2, which has 25 neurons. This layer has 1,625 trainable parameters.

Finally, the output of dense_2 is fed into the last layer, dense_3, which has a single output neuron that represents the predicted value for the next time step. This layer has 26 trainable parameters. Overall, the model has 117,619 trainable parameters, and the output of the model has shape (None, 1), meaning that it outputs a single value for each input sequence. After completing the model training, we will run a diagnostic check to evaluate the model's performance.

4.6.3 Diagnostic Check

Model Fit Statistics			
MSE	MAE	RMSE	MAPE
0.024	0.12	0.15	0.031

Table 4 Model Statistics

In Table 4, MAPE (mean absolute percentage error) is commonly to interpret and explain in the time series diagnostic check therefore MAPE around 3.1% implies the model is about 96.9% accurate in the predicting the testing set observation. MSE (mean squared error) around 0.024 (average squared) between the actual value and predicted value. According the (Lewis, C.D. (1982)), MAPE less than 10 % can consider a good accurate forecast. After conclude all the diagnostic check can plot out the forecasting graph to see whether the whether it is observed that actual price paired well with prediction price.

4.6.4 Forecasting Graph

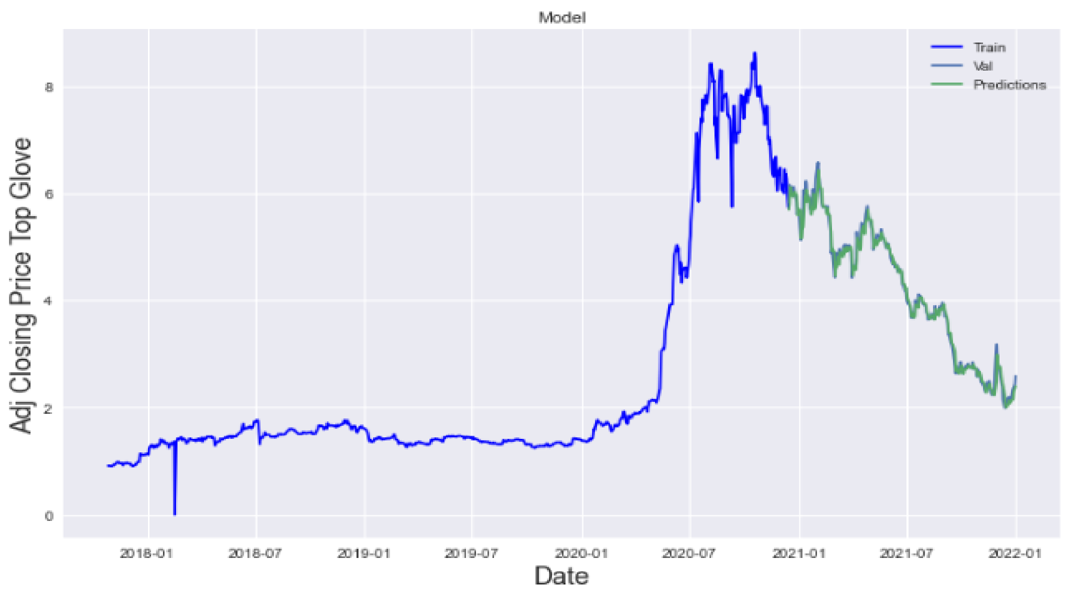


Figure 22 Forecasting LSTM

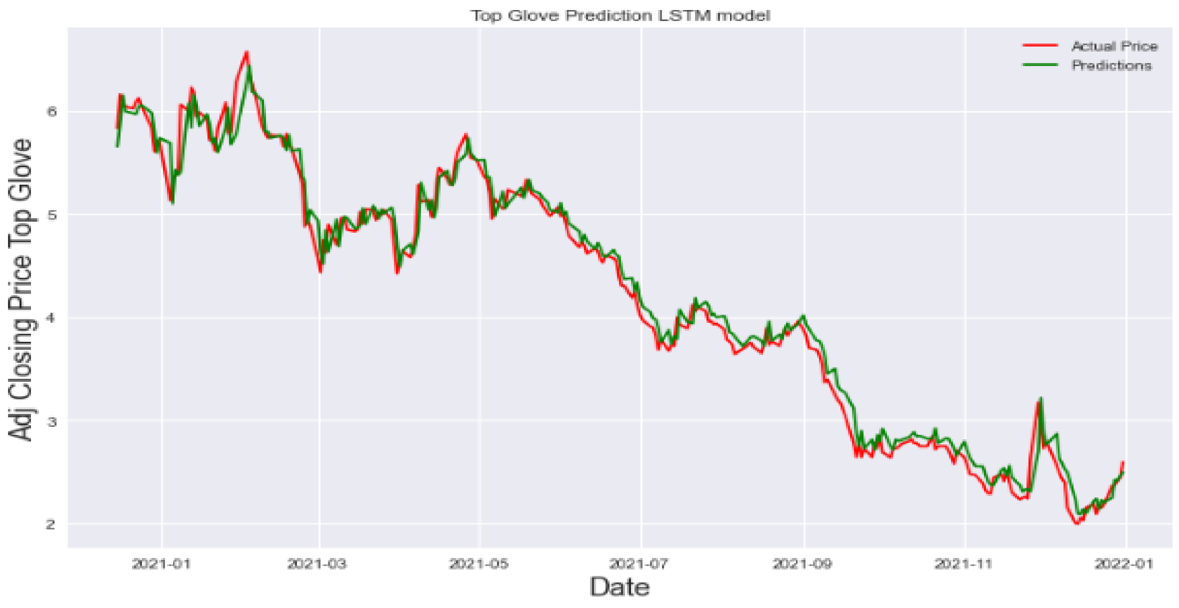


Figure 23 Forecasting LSTM from 2020/12 – 2021/12

In Figure 22, blue line is the train dataset, light blue line is the test dataset and green line is the prediction values using the LSTM model. In Figure 23, can clearly see the actual price compared with the predicted price. Based on this graph you can see that the prediction price is fit with the actual price. Completing all steps means that future value predictions can be obtained by this model.

4.7 Model Evaluation

Accuracy metrics are obtained directly from 2020-12-15 to 2021-12-31. MAPE (mean absolute percentage error) is commonly to interpret and explain in the Time series diagnostic check. Table 5 shows all the accuracy value for ARIMA model, and LSTM model.

Models	MSE	MAE	RMSE	MAPE
LSTM	0.0247096	0.1213672	0.1571928	0.0318378
ARIMA (2,1, 5)	3.5613131	1.5105113	1.8871441	0.5049844

Table 5 Result of the model evaluation

From table 5, we can see that LSTM model having a similar accuracy value. Therefore, LSTM model is the best model on this project for prediction stock market price compare with ARIMA model.

4.8 LSTM Future 5 days Price

Day	Actual Price	Prediction Price Future 5 days - LSTM
2022-01-01	0	2.49
2022-01-02	0	2.54
2022-01-03	2.39	2.59
2022-01-04	2.33	2.65
2022-01-05	2.44	2.70

Table 6 Comparison between the actual price and future price predicted by LSTM and ARIMA model

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

To sum up, this project mainly focuses on studying the movement of Top Glove (7113.KL) stock prices. Using statistical methods and deep learning methods to find a suitable model to predict stock market prices. In this study, the two models found are Autoregressive Integrated Moving Average (ARIMA) and Long Short-Term Memory (LSTM). These models are used to predict the Top Glove (7113.KL) stock prices. The LSTM model forecasting result is found to be closed to the Top Glove (7113.KL) daily stock prices from 2020/12/15-2022/12/31.

Between the Autoregressive Integrated Moving Average (ARIMA) and Long Short-Term Memory (LSTM) models, it is found that Long Short-Term Memory (LSTM) appears to be more suitable in predicting prices in the stock market. From the result shown in Table 5, we found that the LSTM model for predicting stock prices is the better option for prediction. Furthermore, LSTM is simpler than mathematical statistical methods. Since stock prices often exhibit complex non-linear behaviour, LSTMs are a better choice.

Conclusion, the LSTM model can be used to predict the future stock price. However, LSTM is not 100% predictive of future prices, because there are some uncertain factors in the stock market that lead to stock market crashes, where the predictive model does not consider the existence of these factors.

.

5.2 Future Work

The stock market exerts significant influence on the economy and can also impact the development of a country. Hence, it is crucial for investors to be adept at predicting future stock prices. By accurately predicting stock market prices, they can potentially reap significant financial benefits and earn their first big profit in the stock market.

For future work, the ARIMA and LSTM models have potential utility in other time series domains such as rainfall prediction, traffic accident analysis, forecasting palm oil prices, disease rate, etc. In addition, the potential future research could apply alternative deep learning methods for forecasting, such as the popular Hybrid Prophet algorithm in the research community.

The future research could also consider combining ARIMA and LSTM models with other statistical and deep learning methods to improve prediction accuracy in addition to their applicability in other time series domains. Overall, more investigation into predictive modelling in numerous fields has the potential to advance informed decision-making and benefit society.

REFERENCES

[1] Investopedia, 2022, *Understanding Time Series*, viewed 9 August 2022, https://www.investopedia.com/terms/t/timeseries.asp#:~:text=A%20time%20series%20is%20a%20data%20set%20that%20tracks%20a,economic%20variable%20changes%20over%20time..

[2] Henrik Madsen, 2022, *Time_Series_Analysis*, viewed 9 August 2022, <https://www.researchgate.net/publication/228068920_Time_Series_Analysis.

[3] Adebayo, Fatai & R, Sivasamy & Shangodoyin, Kehinde. (2014). "Statistical and Econometric Methods", *Forecasting Stock Market Series with ARIMA Model*. Vol 3.pp. 65 - 77.

[4] Khair1, U., Fahmi1, H., Hakim1, S.A. and Rahim2, R., A 2017, 'Journal of Physics: Conference Series', Forecasting Error Calculation with Mean Absolute Deviation and Mean Absolute Percentage Error, vol 930, 012002, pp. 1-6.

[5] Ayodele A. Adebiyi., 2Aderemi O. Adewumi, Charles K. Ayo, M 2014, '2014 UKSim-AMSS 16th International Conference on Computer Modeling and Simulation At: Cambridge University, United Kingdom', Stock price prediction using the ARIMA model, doi: 10.1109/UKSim.2014.67., pp. 106-112.

[6] Ma, Qihang. (2020). *Comparison of ARIMA, ANN and LSTM for Stock Price Prediction*. Vol 218, 01026. 10.1051/e3sconf/202021801026, pp 1-5.

[7] Patil, Rajat. (2021). *Time Series Analysis and Stock Price Forecasting using Machine Learning Techniques*. Vol 19, 10.1994/Rajat/AI, pp.1-19.

[8] Rahkmawati, Y., Sumertajaya, I.M. and Nur Aidi, M. (2019). Evaluation of Accuracy in Identification of ARIMA Models Based on Model Selection Criteria for Inflation Forecasting with the TSClust Approach. *International Journal of Scientific and Research Publications (IJSRP)*, 9(9), p.p9355. doi: https://doi.org/10.29322/ijsrp.9.09.2019.p9355.

[9] Xiao, R., Feng, Y., Yan, L. and Ma, Y. (n.d.). *PREDICT STOCK PRICES WITH ARIMA AND LSTM*. [online] Available at: https://arxiv.org/pdf/2209.02407.pdf [Accessed 24 Jan. 2023].

[10] Moghar, A. and Hamiche, M. (2020). Stock Market Prediction Using LSTM Recurrent Neural Network. *Procedia Computer Science*, 170, pp.1168–1173. doi: https://doi.org/10.1016/j.procs.2020.03.049.

[11] Bhandari, H.N., Rimal, B., Pokhrel, N.R., Rimal, R., Dahal, K.R. and Khatri, R.K.C. (2022). Predicting stock market index using LSTM. *Machine Learning with Applications*, p.100320. doi: https://doi.org/10.1016/j.mlwa.2022.100320.

[12] Van Houdt, G., Mosquera, C. and Nápoles, G. (2020). A review on the long short-term memory model. *Artificial Intelligence Review*. doi: https://doi.org/10.1007/s10462-020-09838-1.

[13] Zach (2020). *How to Perform a Ljung-Box Test in Python*. [online] Statology. Available at: https://www.statology.org/ljung-box-test-python/#:~:text=The%20Ljung%2DBox%20test%20is [Accessed 13 Mar. 2023].

[14] Selva Prabhakaran (2019). *ARIMA Model - Complete Guide to Time Series Forecasting in Python | ML+*. [online] Machine Learning Plus. Available at: https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/.

[15] Analytics Vidhya. (2021). *Stock market forecasting using Time Series analysis With ARIMA model*. [online] Available at: https://www.analyticsvidhya.com/blog/2021/07/stock-market-forecasting-using-time-series-analysis-with-arima-model/.

[16] Lewis, C.D. (1982), Industrial and business forecasting methods: A practical guide to exponential smoothing and curve ting, London; Boston: Butterworth Scientist.

[17] Keshvani, A. (2013). *Using AIC to Test ARIMA Models*. [online] CoolStatsBlog. Available at: https://coolstatsblog.com/2013/08/14/using-aic-to-test-arima-models-2/.

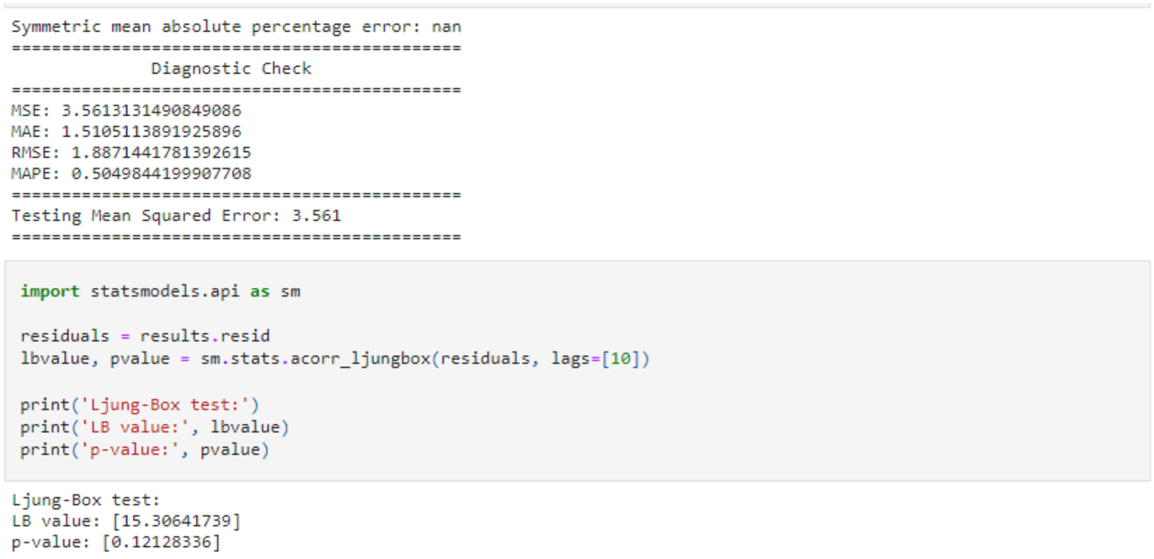
[18] Börjesson, L. and Singull, M. (2020). Forecasting Financial Time Series through Causal and Dilated Convolutional Neural Networks. *Entropy*, 22(10), p.1094. doi:https://doi.org/10.3390/e22101094.

[19] Anderson, O.D. (1976). *Time series analysis and forecasting : the Box-Jenkins approach*. London ; Boston: Butterworth.

APPENDICES

APPENDIX A: Graphs

ARIMA (2,1,5)

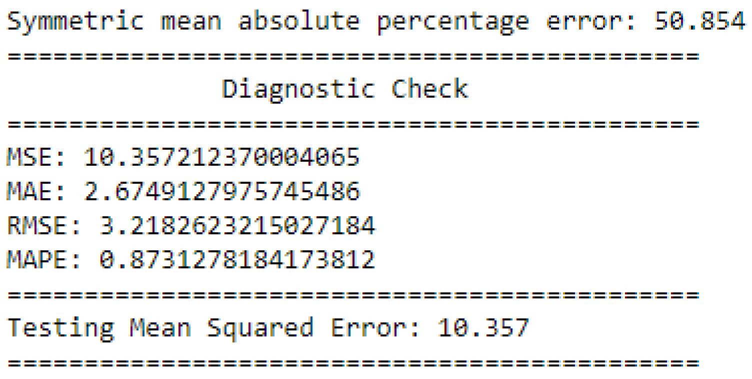


```
Symmetric mean absolute percentage error: nan
============================================
              Diagnostic Check
============================================
MSE: 3.5613131490849086
MAE: 1.5105113891925896
RMSE: 1.8871441781392615
MAPE: 0.5049844199907708
============================================
Testing Mean Squared Error: 3.561
============================================
```

```python
import statsmodels.api as sm

residuals = results.resid
lbvalue, pvalue = sm.stats.acorr_ljungbox(residuals, lags=[10])

print('Ljung-Box test:')
print('LB value:', lbvalue)
print('p-value:', pvalue)
```

```
Ljung-Box test:
LB value: [15.30641739]
p-value: [0.12128336]
```

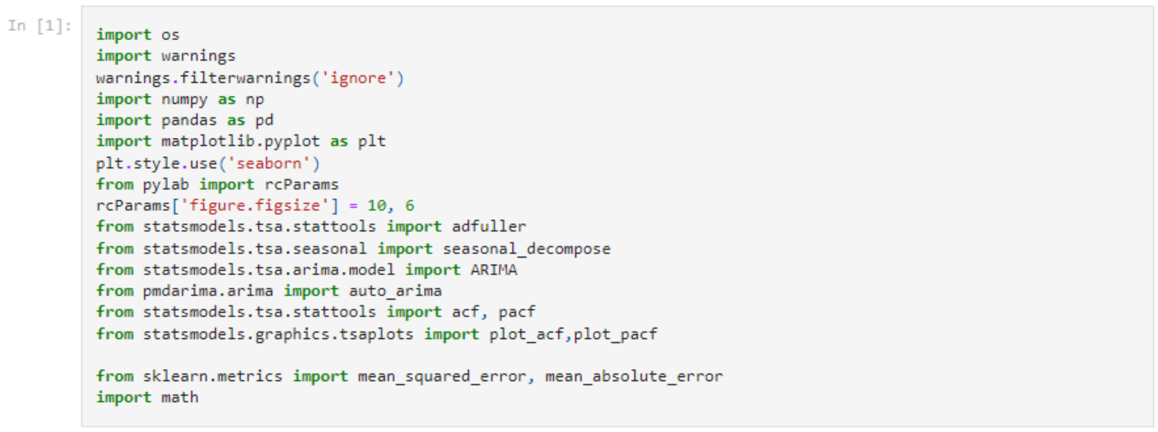
Auto-ARIMA (2, 1, 3)



```
Symmetric mean absolute percentage error: 50.854
============================================
              Diagnostic Check
============================================
MSE: 10.357212370004065
MAE: 2.6749127975745486
RMSE: 3.2182623215027184
MAPE: 0.8731278184173812
============================================
Testing Mean Squared Error: 10.357
============================================
```

APPENDIX B : Computer Programming

ARIMA model



```
In [1]: import os
        import warnings
        warnings.filterwarnings('ignore')
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        plt.style.use('seaborn')
        from pylab import rcParams
        rcParams['figure.figsize'] = 10, 6
        from statsmodels.tsa.stattools import adfuller
        from statsmodels.tsa.seasonal import seasonal_decompose
        from statsmodels.tsa.arima.model import ARIMA
        from pmdarima.arima import auto_arima
        from statsmodels.tsa.stattools import acf, pacf
        from statsmodels.graphics.tsaplots import plot_acf,plot_pacf

        from sklearn.metrics import mean_squared_error, mean_absolute_error
        import math
```

Import Dataset

In [2]:
```
dateparse = lambda dates: pd.datetime.strptime(dates, '%Y-%m-%d')
path  = "C:/Users/User/Downloads/7113.KL(2017-2022).csv"
topGlove = pd.read_csv(path, index_col='Date', parse_dates=['Date'], date_parser=dateparse)
```

In [3]:
```
topGlove
```

Out[3]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2017-10-27	1.073333	1.103333	1.071666	1.075000	0.912288	27835200.0
2017-10-30	1.080000	1.091666	1.073333	1.075000	0.912288	7378200.0
2017-10-31	1.083333	1.083333	1.053333	1.066666	0.905216	9283200.0
2017-11-01	1.066666	1.083333	1.065000	1.066666	0.905216	5490600.0
2017-11-02	1.066666	1.080000	1.063333	1.066666	0.905216	10039800.0
...	...	...	...	...	...	...
2021-12-27	2.250000	2.450000	2.210000	2.360000	2.360000	38963400.0
2021-12-28	2.370000	2.410000	2.300000	2.370000	2.370000	18604700.0
2021-12-29	2.380000	2.480000	2.360000	2.410000	2.410000	22588800.0
2021-12-30	2.420000	2.450000	2.380000	2.440000	2.440000	20901200.0
2021-12-31	2.440000	2.590000	2.340000	2.590000	2.590000	29980300.0

1028 rows × 6 columns

1028 rows × 6 columns

In [4]:
```
topGlove.isnull().sum()
```

Out[4]:
```
Open         1
High         1
Low          1
Close        1
Adj Close    1
Volume       1
dtype: int64
```

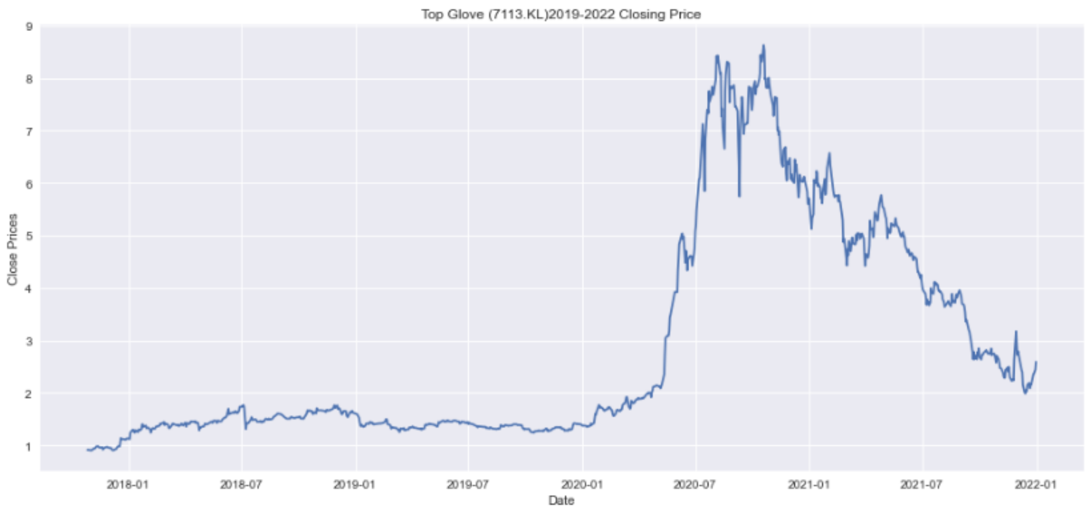
In [5]:
```
topGlove.fillna(method='ffill', inplace=True)
```

In [6]:
```
topGlove.isnull().sum()
```

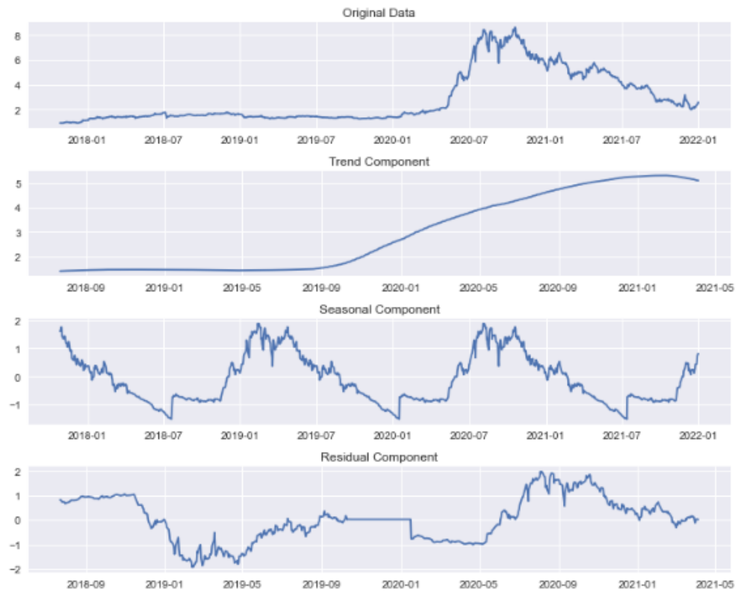
Out[6]:
```
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

Plot Top Glove Share Price Graph

In [7]:
```
plt.figure(figsize=(16,7))
plt.grid(True)
plt.xlabel('Date')
plt.ylabel('Close Prices')
plt.plot(topGlove['Adj Close'])
plt.title('Top Glove (7113.KL)2019-2022 Closing Price')
plt.show()
```



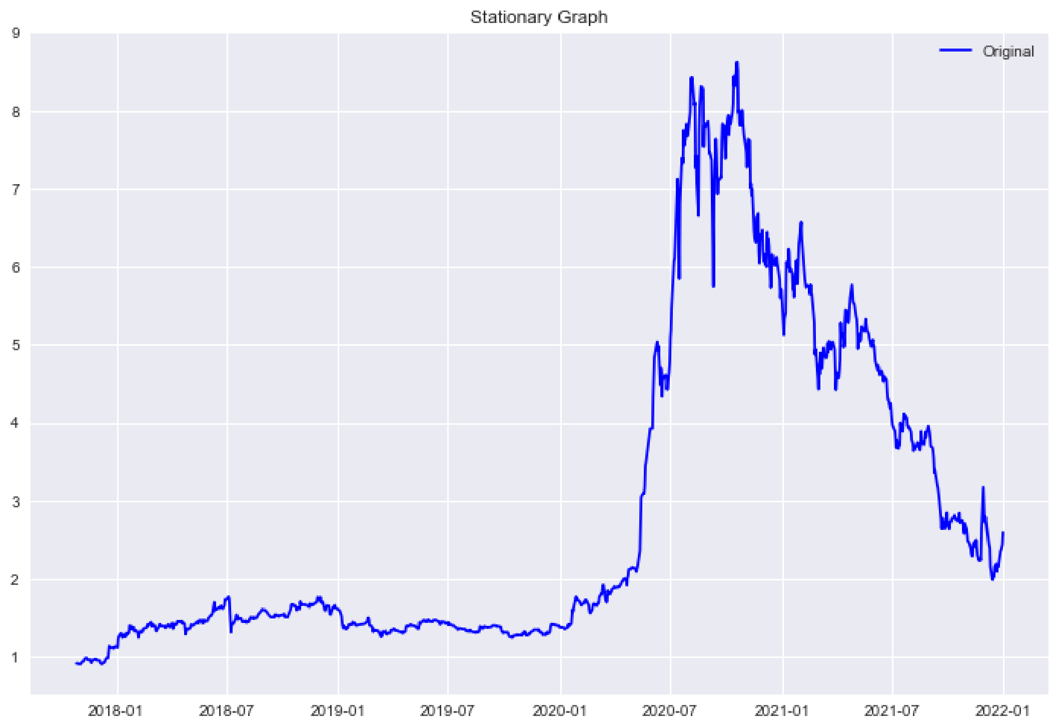
```
In [20]: result = seasonal_decompose(topGlove['Adj Close'], model='additive', period=365)

         fig, (ax1, ax2, ax3, ax4) = plt.subplots(4, 1, figsize=(10, 8))
         ax1.plot(topGlove['Adj Close'])
         ax1.set_title('Original Data')
         ax2.plot(result.trend)
         ax2.set_title('Trend Component')
         ax3.plot(result.seasonal)
         ax3.set_title('Seasonal Component')
         ax4.plot(result.resid)
         ax4.set_title('Residual Component')
         plt.tight_layout()
         plt.show()
```



Dickey-Fuller Test

```
In [8]:  def adfuller_test(stocks):
             result=adfuller(stocks)
             labels = ['ADF Test Statistic','p-value','#Lags Used','Number of Observations']
             for value,label in zip(result,labels):
                 print(label+' : '+str(value) )
```

```
In [9]:  from statsmodels.tsa.stattools import adfuller
         def test_stationarity(timeseries, window = 12, cutoff = 0.01):

             #Determing rolling statistics
             rolmean = timeseries.rolling(window).mean()
             rolstd = timeseries.rolling(window).std()

             #Plot rolling statistics:
             fig = plt.figure(figsize=(12, 8))
             orig = plt.plot(timeseries, color='blue',label='Original')
             #mean = plt.plot(rolmean, color='red', label='Rolling Mean')
             #std = plt.plot(rolstd, color='black', label = 'Rolling Std')
             plt.legend(loc='best')
             plt.title('Stationary Graph')
             plt.show()

             #Perform Dickey-Fuller test:
             print('Results of Dickey-Fuller Test:')
             dftest = adfuller(timeseries, autolag='AIC', maxlag = 20 )
             dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','Number of Observations Used
             for key,value in dftest[4].items():
                 dfoutput['Critical Value (%s)'%key] = value
             pvalue = dftest[1]
             if pvalue < cutoff:
                 print('p-value = %.4f. The series is likely stationary.' % pvalue)
             else:
                 print('p-value = %.4f. The series is likely non-stationary.' % pvalue)

             print(dfoutput)

         test_stationarity(topGlove['Adj Close'])
```

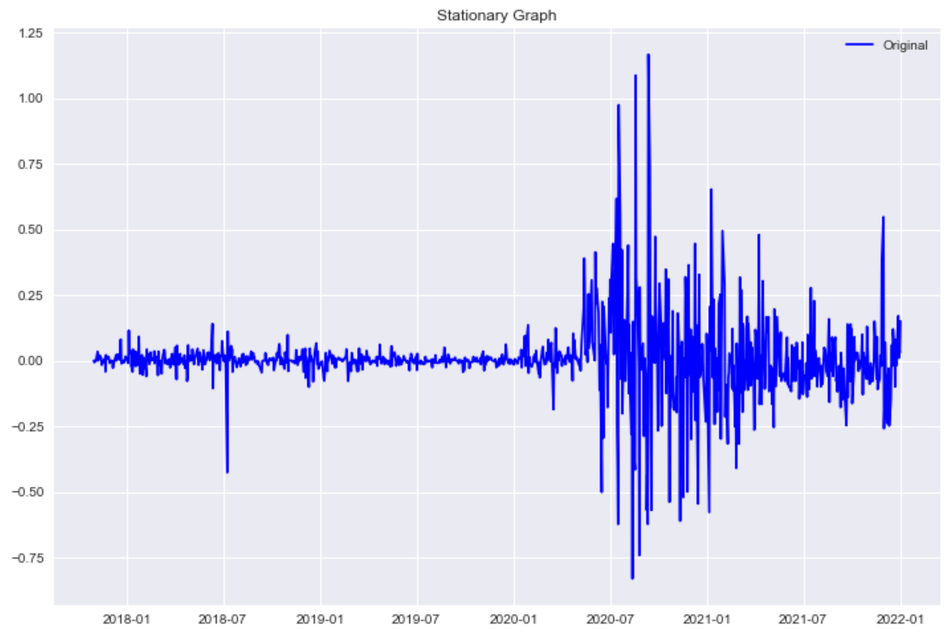


```
Results of Dickey-Fuller Test:
p-value = 0.5898. The series is likely non-stationary.
Test Statistic                   -1.384358
p-value                           0.589752
#Lags Used                       19.000000
Number of Observations Used    1008.000000
Critical Value (1%)              -3.436854
Critical Value (5%)              -2.864412
Critical Value (10%)             -2.568299
dtype: float64
```

```
Number of Observations : 1008
```

In [11]:
```
first_diff =topGlove['Adj Close']-topGlove['Adj Close'].shift(1)
first_diff = first_diff.dropna(inplace = False)
```

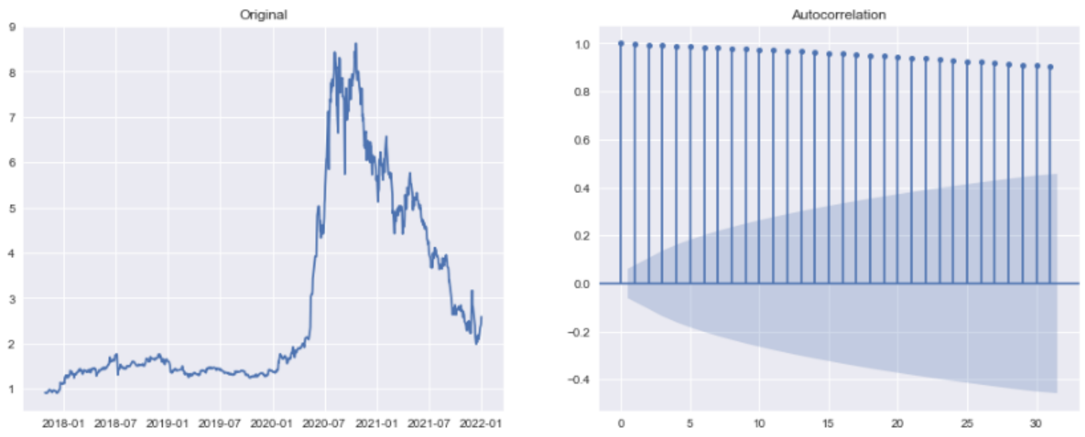
In [12]:
```
test_stationarity(first_diff, window = 12)
```



```
Results of Dickey-Fuller Test:
p-value = 0.0000. The series is likely stationary.
Test Statistic                -6.582493e+00
p-value                        7.453808e-09
#Lags Used                     1.800000e+01
Number of Observations Used    1.008000e+03
Critical Value (1%)           -3.436854e+00
Critical Value (5%)           -2.864412e+00
Critical Value (10%)          -2.568299e+00
dtype: float64
```

Find Parameter p, d, q

In [15]:
```
fig, (ax1,ax2) = plt.subplots(1,2 ,figsize=(16,6))

ax1.plot(topGlove['Adj Close'])
ax1.set_title("Original")
plot_acf(topGlove['Adj Close'] , ax=ax2);
```

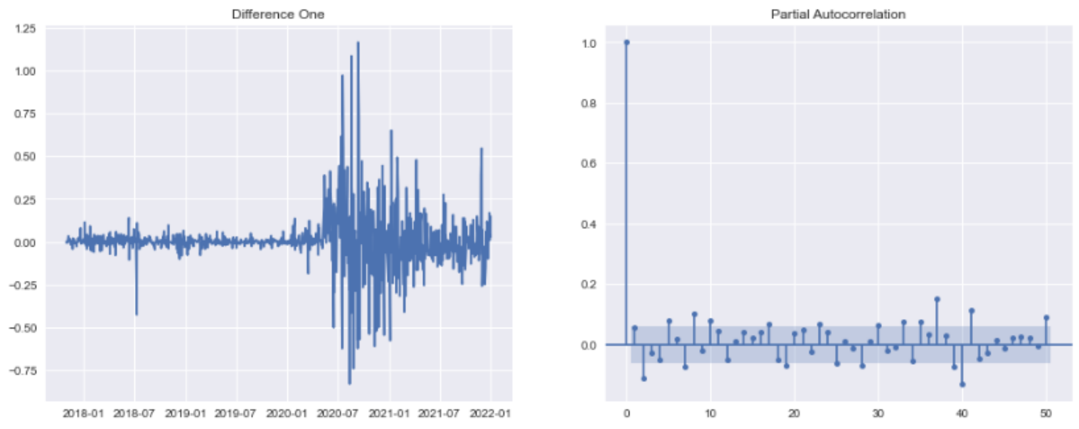


In [16]:
```
#can use other method is pmdarima
from pmdarima.arima.utils import ndiffs
ndiffs(topGlove['Adj Close'], test="adf")
```

Out[16]:
```
1
```

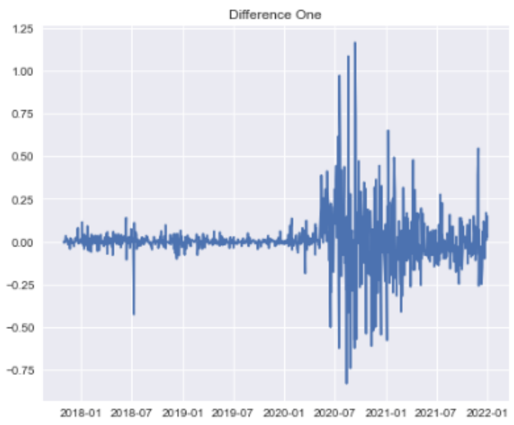
AR(p)

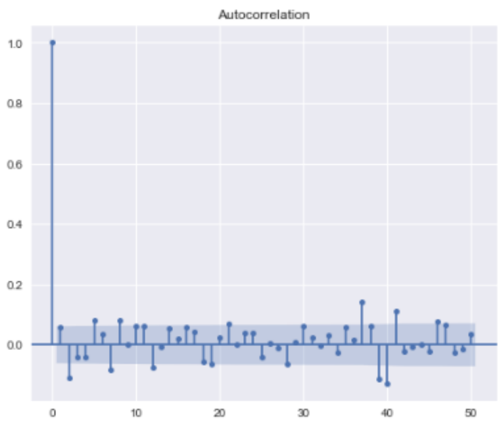
In [17]:
```
diffOne = topGlove['Adj Close'].diff().dropna()

fig, (ax1,ax2) = plt.subplots(1,2 ,figsize=(16,6))
ax1.plot(diffOne)
ax1.set_title("Difference One")
#ax2.set_ylim(0,1)
plot_pacf(diffOne,lags=50,ax=ax2 );
#AR2
```



MA(q)

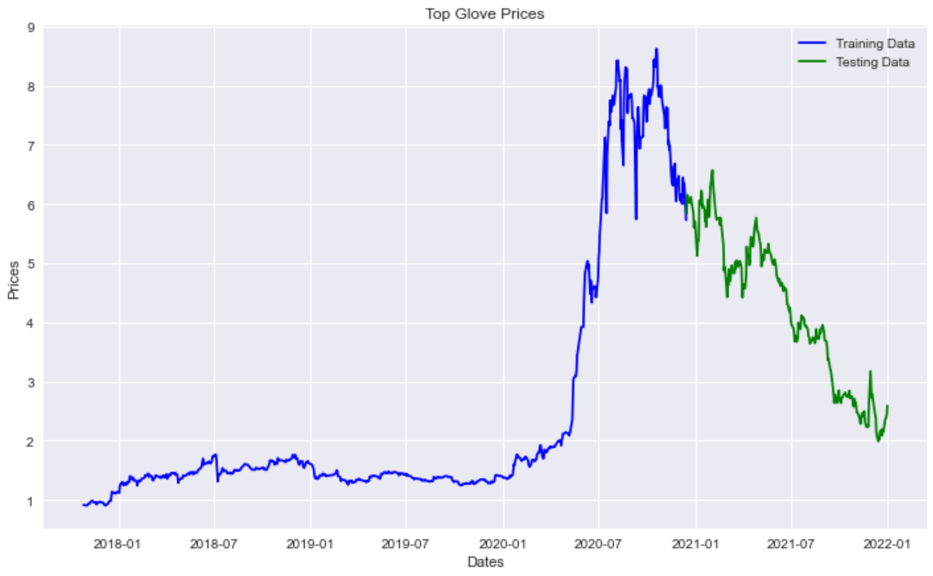
```
In [18]:  diffOne = topGlove['Adj Close'].diff().dropna()

          fig, (ax1,ax2) = plt.subplots(1,2 ,figsize=(16,6))
          ax1.plot(diffOne)
          ax1.set_title("Difference One")
          #ax2.set_ylim(0,1)
          plot_acf(diffOne,lags=50,ax=ax2 );
          #MA 2 5 7 8
```





Split Data

```
In [19]:  train_data, test_data = topGlove[0:int(len(topGlove)*0.75)], topGlove[int(len(topGlove)*0.75):]


          plt.figure(figsize=(12,7))
          plt.title('Top Glove Prices')
          plt.xlabel('Dates')
          plt.ylabel('Prices')
          plt.plot(train_data['Adj Close'], 'blue', label='Training Data')
          plt.plot(test_data['Adj Close'], 'green', label='Testing Data')
          plt.legend()

Out[19]:  <matplotlib.legend.Legend at 0x19a9a021520>
```



```
In [22]:  model = ARIMA(train_data['Adj Close'], order=(2, 1, 2))
          results = model.fit()
```

```
C:\Users\User\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:581: ValueWarning: A date index has b
een provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
C:\Users\User\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:581: ValueWarning: A date index has b
een provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
C:\Users\User\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:581: ValueWarning: A date index has b
een provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
```

```
In [25]:  print(results.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:              Adj Close   No. Observations:                  771
Model:                 ARIMA(2, 1, 2)   Log Likelihood                 435.755
Date:                Sat, 29 Apr 2023   AIC                           -861.510
Time:                        18:26:18   BIC                           -838.278
Sample:                             0   HQIC                          -852.569
                                - 771
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.5743      0.079      7.225      0.000       0.418       0.730
ar.L2         -0.5191      0.085     -6.133      0.000      -0.685      -0.353
ma.L1         -0.5004      0.086     -5.795      0.000      -0.670      -0.331
ma.L2          0.3648      0.092      3.977      0.000       0.185       0.545
sigma2         0.0189      0.000     57.952      0.000       0.018       0.020
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):             13022.13
Prob(Q):                              0.91   Prob(JB):                         0.00
Heteroskedasticity (H):              33.39   Skew:                             0.90
Prob(H) (two-sided):                  0.00   Kurtosis:                        23.07
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

```
In [21]:  model = ARIMA(train_data['Adj Close'], order=(2, 1, 5))
          results = model.fit()
```

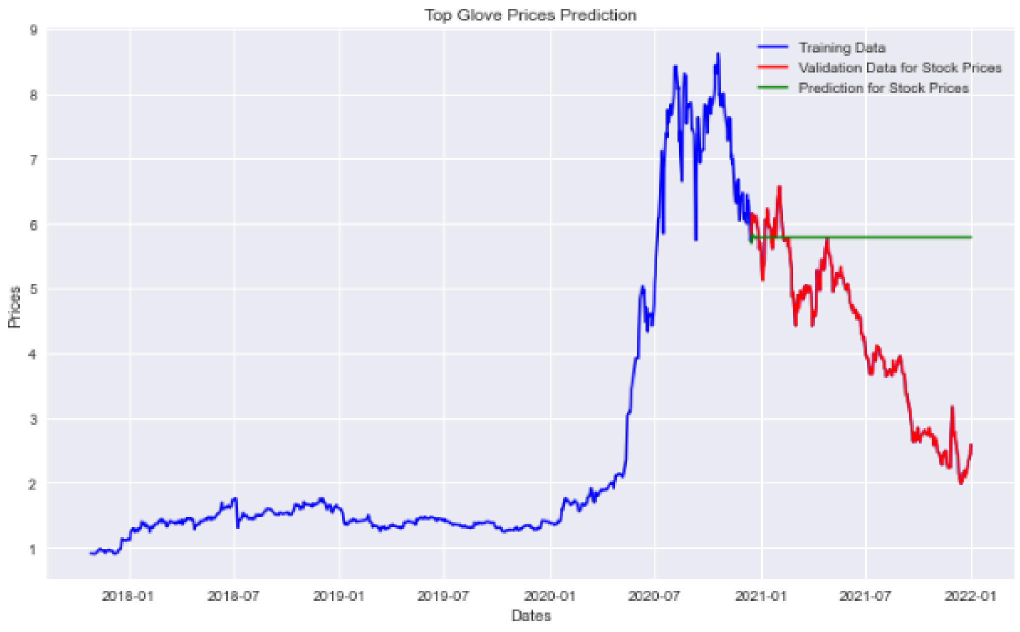
```
C:\Users\User\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:581: ValueWarning: A date index has b
een provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
C:\Users\User\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:581: ValueWarning: A date index has b
een provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
C:\Users\User\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:581: ValueWarning: A date index has b
een provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
C:\Users\User\anaconda3\lib\site-packages\statsmodels\base\model.py:566: ConvergenceWarning: Maximum Likelihood o
ptimization failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```

```
In [22]:  print(results.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:              Adj Close   No. Observations:                  771
Model:                 ARIMA(2, 1, 5)   Log Likelihood                 442.327
Date:                Sat, 29 Apr 2023   AIC                           -868.654
Time:                        18:28:19   BIC                           -831.483
Sample:                             0   HQIC                          -854.349
                                - 771
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.7614      0.214      3.565      0.000       0.343       1.180
ar.L2          0.1292      0.199      0.648      0.517      -0.262       0.520
ma.L1         -0.6988      0.214     -3.259      0.001      -1.119      -0.279
ma.L2         -0.2994      0.193     -1.547      0.122      -0.679       0.080
ma.L3         -0.0063      0.042     -0.152      0.879      -0.088       0.075
ma.L4          0.0316      0.021      1.490      0.136      -0.010       0.073
ma.L5          0.1219      0.025      4.878      0.000       0.073       0.171
sigma2         0.0186      0.000     54.640      0.000       0.018       0.019
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):             11674.39
Prob(Q):                              0.94   Prob(JB):                         0.00
Heteroskedasticity (H):              32.44   Skew:                             0.58
Prob(H) (two-sided):                  0.00   Kurtosis:                        22.04
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

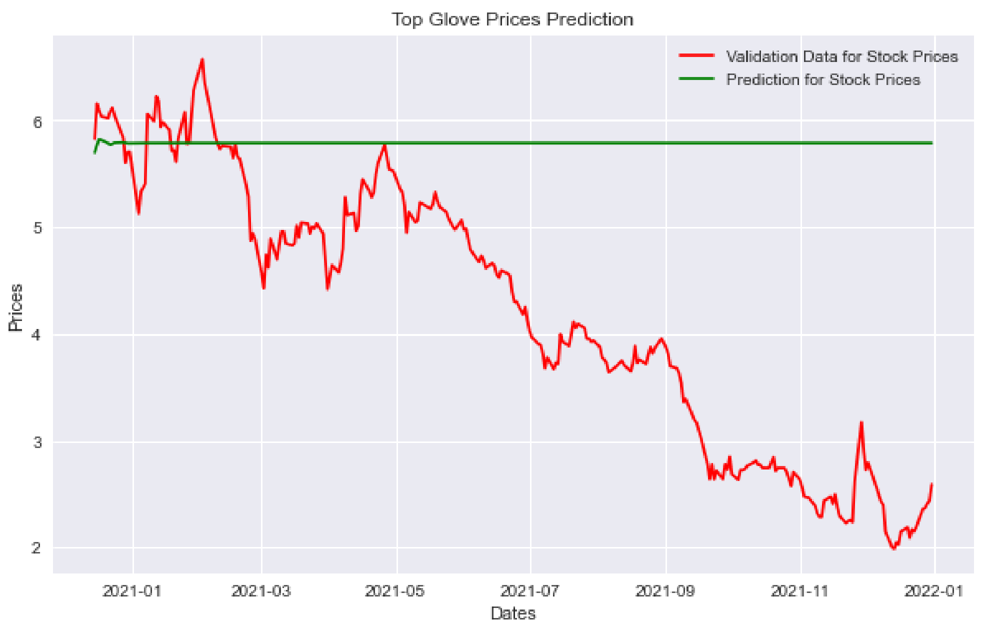
```
In [35]: plt.figure(figsize=(12,7))
         plt.plot(topGlove['Adj Close'], 'green', color='blue', label='Training Data')
         plt.plot(test_data.index,test_data["Adj Close"],'red',label="Validation Data for
         plt.plot(test_data.index, forecast,'green',label="Prediction for Stock Prices")
         plt.title('Top Glove Prices Prediction')
         plt.xlabel('Dates')
         plt.ylabel('Prices')
         plt.legend()
```

```
Out[35]: <matplotlib.legend.Legend at 0x19a862fdeb0>
```



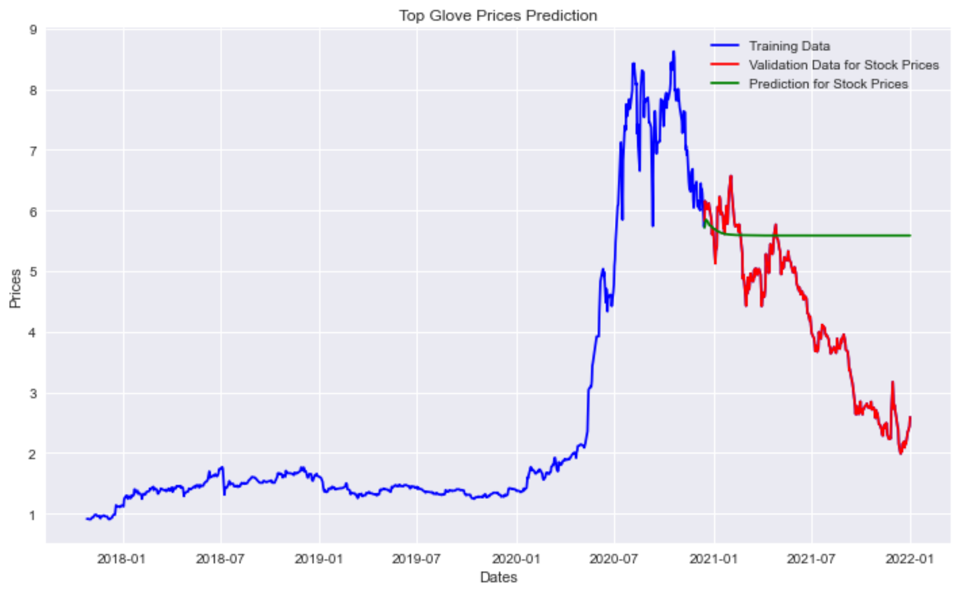
In [36]:
```
plt.plot(test_data.index,test_data["Adj Close"],'red',label="Validation Data for
plt.plot(test_data.index, forecast,'green',label="Prediction for Stock Prices")
plt.title('Top Glove Prices Prediction')
plt.xlabel('Dates')
plt.ylabel('Prices')
plt.legend()
```

Out[36]:
```
<matplotlib.legend.Legend at 0x19a9b0be430>
```

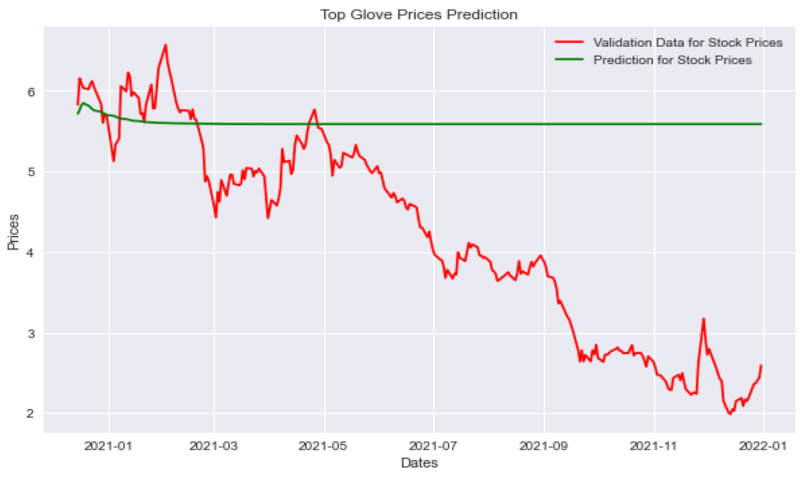


In [26]:
```
plt.figure(figsize=(12,7))
plt.plot(topGlove['Adj Close'], 'green', color='blue', label='Training Data')
plt.plot(test_data.index,test_data["Adj Close"],'red',label="Validation Data for Stock Prices")
plt.plot(test_data.index, forecast,'green',label="Prediction for Stock Prices")
plt.title('Top Glove Prices Prediction')
plt.xlabel('Dates')
plt.ylabel('Prices')
plt.legend()
```

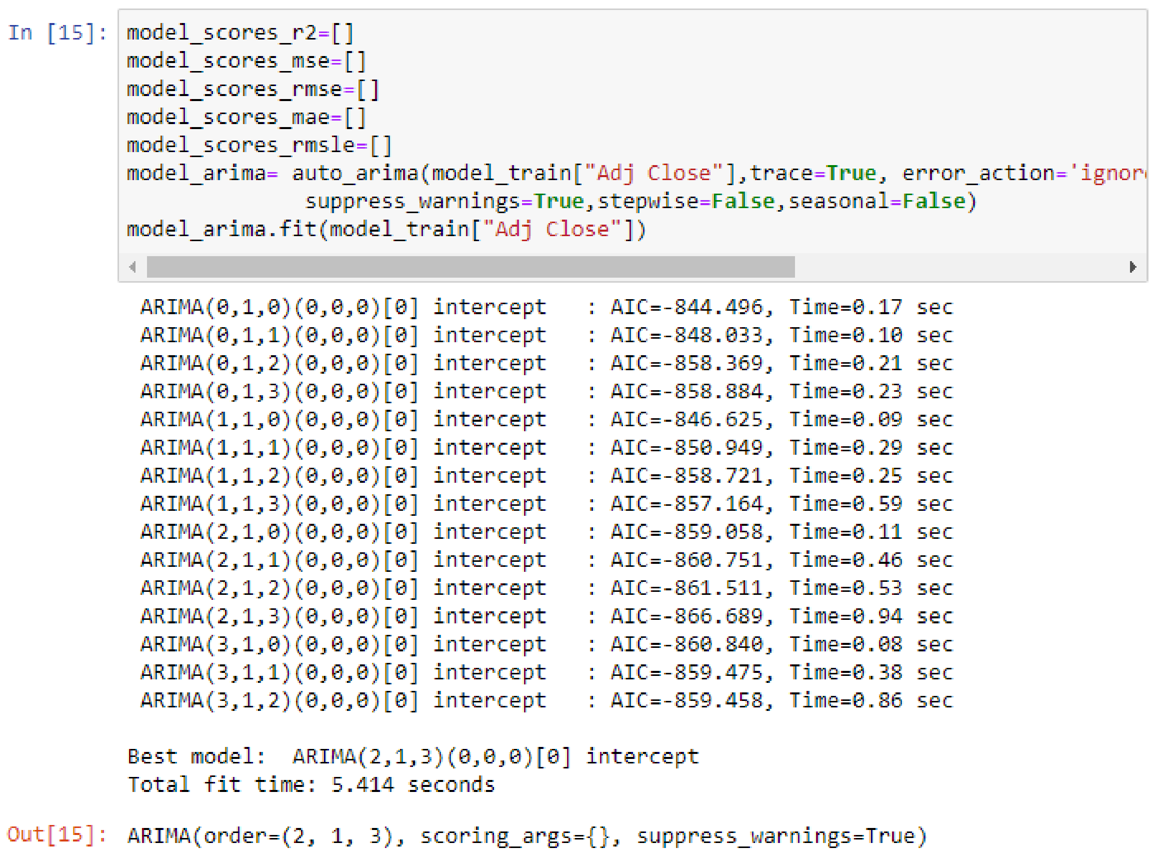
Out[26]:
```
<matplotlib.legend.Legend at 0x1c697368bb0>
```



```
In [27]: plt.plot(test_data.index,test_data["Adj Close"],'red',label="Validation Data for Stock Prices")
         plt.plot(test_data.index, forecast,'green',label="Prediction for Stock Prices")
         plt.title('Top Glove Prices Prediction')
         plt.xlabel('Dates')
         plt.ylabel('Prices')
         plt.legend()

Out[27]: <matplotlib.legend.Legend at 0x1c6973317f0>
```

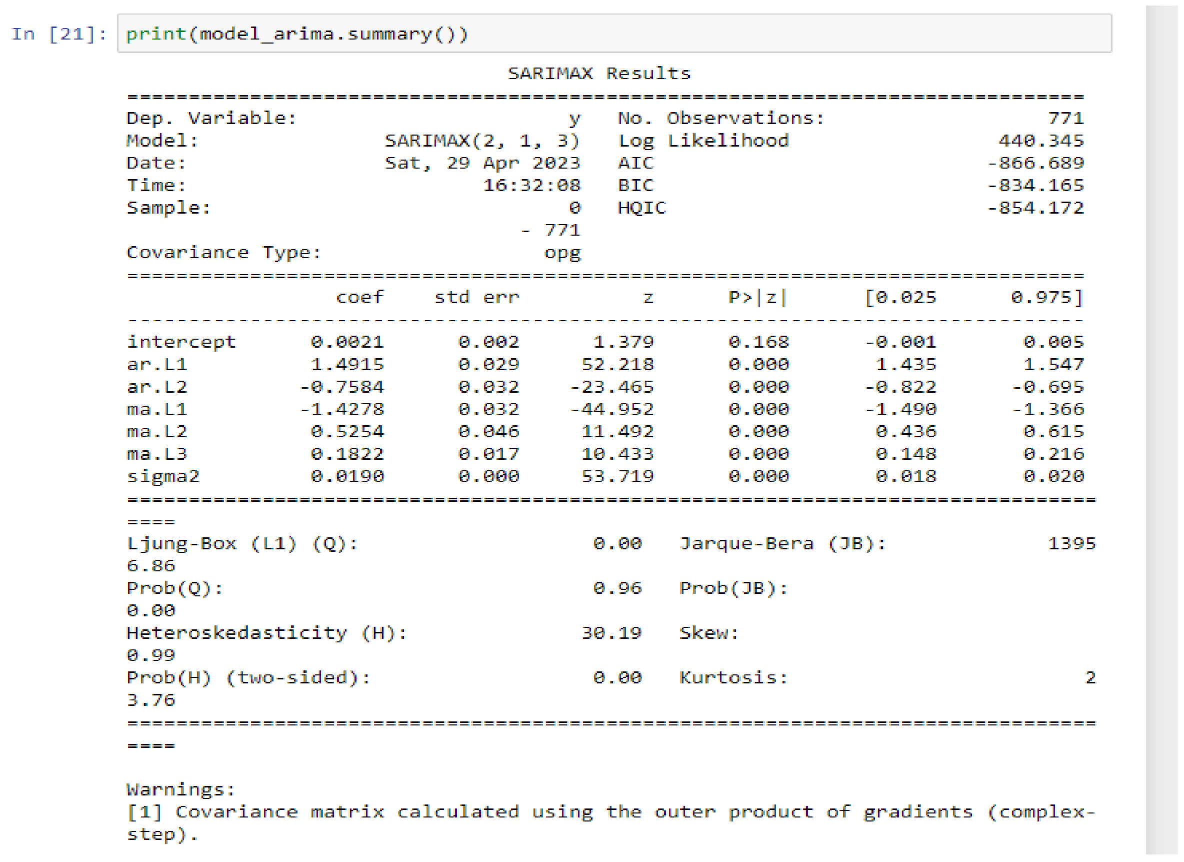


Auto-ARIMA



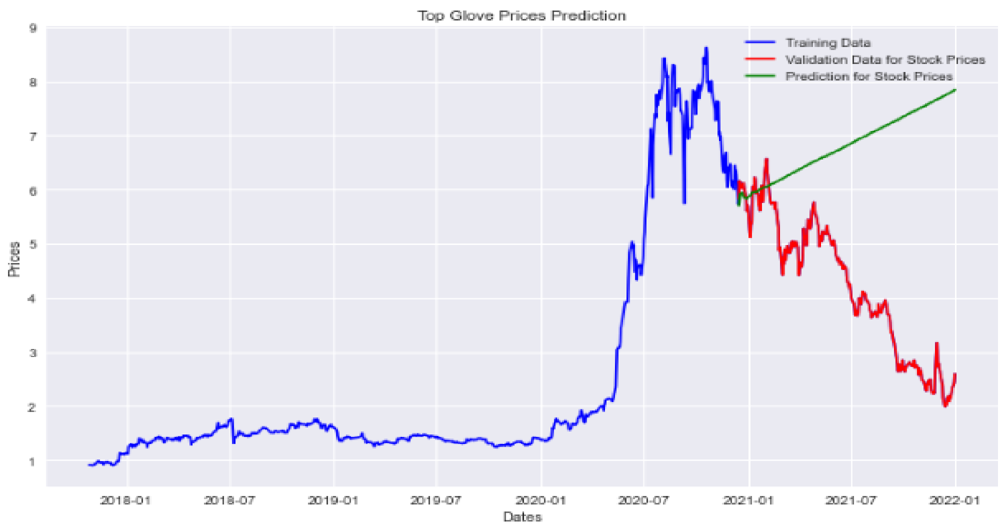
```
In [15]: model_scores_r2=[]
         model_scores_mse=[]
         model_scores_rmse=[]
         model_scores_mae=[]
         model_scores_rmsle=[]
         model_arima= auto_arima(model_train["Adj Close"],trace=True, error_action='ignor
                       suppress_warnings=True,stepwise=False,seasonal=False)
         model_arima.fit(model_train["Adj Close"])
```

```
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=-844.496, Time=0.17 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=-848.033, Time=0.10 sec
 ARIMA(0,1,2)(0,0,0)[0] intercept   : AIC=-858.369, Time=0.21 sec
 ARIMA(0,1,3)(0,0,0)[0] intercept   : AIC=-858.884, Time=0.23 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=-846.625, Time=0.09 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=-850.949, Time=0.29 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=-858.721, Time=0.25 sec
 ARIMA(1,1,3)(0,0,0)[0] intercept   : AIC=-857.164, Time=0.59 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=-859.058, Time=0.11 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=-860.751, Time=0.46 sec
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=-861.511, Time=0.53 sec
 ARIMA(2,1,3)(0,0,0)[0] intercept   : AIC=-866.689, Time=0.94 sec
 ARIMA(3,1,0)(0,0,0)[0] intercept   : AIC=-860.840, Time=0.08 sec
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AIC=-859.475, Time=0.38 sec
 ARIMA(3,1,2)(0,0,0)[0] intercept   : AIC=-859.458, Time=0.86 sec

Best model:  ARIMA(2,1,3)(0,0,0)[0] intercept
Total fit time: 5.414 seconds

Out[15]: ARIMA(order=(2, 1, 3), scoring_args={}, suppress_warnings=True)
```

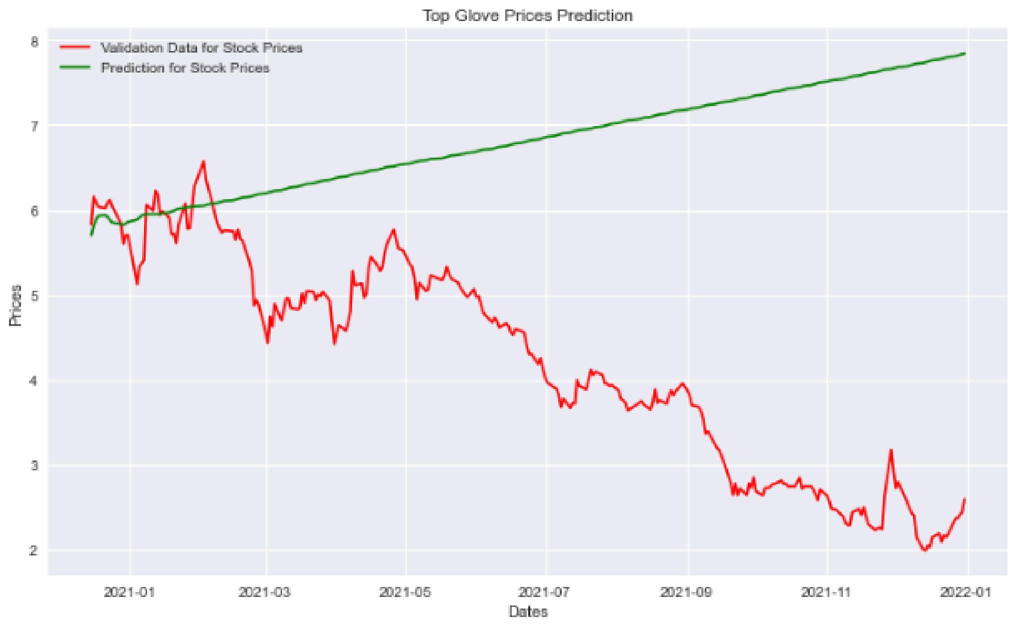


```
In [21]: print(model_arima.summary())

                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  771
Model:               SARIMAX(2, 1, 3)   Log Likelihood                 440.345
Date:                Sat, 29 Apr 2023   AIC                           -866.689
Time:                        16:32:08   BIC                           -834.165
Sample:                             0   HQIC                          -854.172
                                - 771
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      0.0021      0.002      1.379      0.168      -0.001       0.005
ar.L1          1.4915      0.029     52.218      0.000       1.435       1.547
ar.L2         -0.7584      0.032    -23.465      0.000      -0.822      -0.695
ma.L1         -1.4278      0.032    -44.952      0.000      -1.490      -1.366
ma.L2          0.5254      0.046     11.492      0.000       0.436       0.615
ma.L3          0.1822      0.017     10.433      0.000       0.148       0.216
sigma2         0.0190      0.000     53.719      0.000       0.018       0.020
===============================================================================
====
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):              1395
6.86
Prob(Q):                              0.96   Prob(JB):
0.00
Heteroskedasticity (H):              30.19   Skew:
0.99
Prob(H) (two-sided):                  0.00   Kurtosis:                         2
3.76
===============================================================================
====

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
```

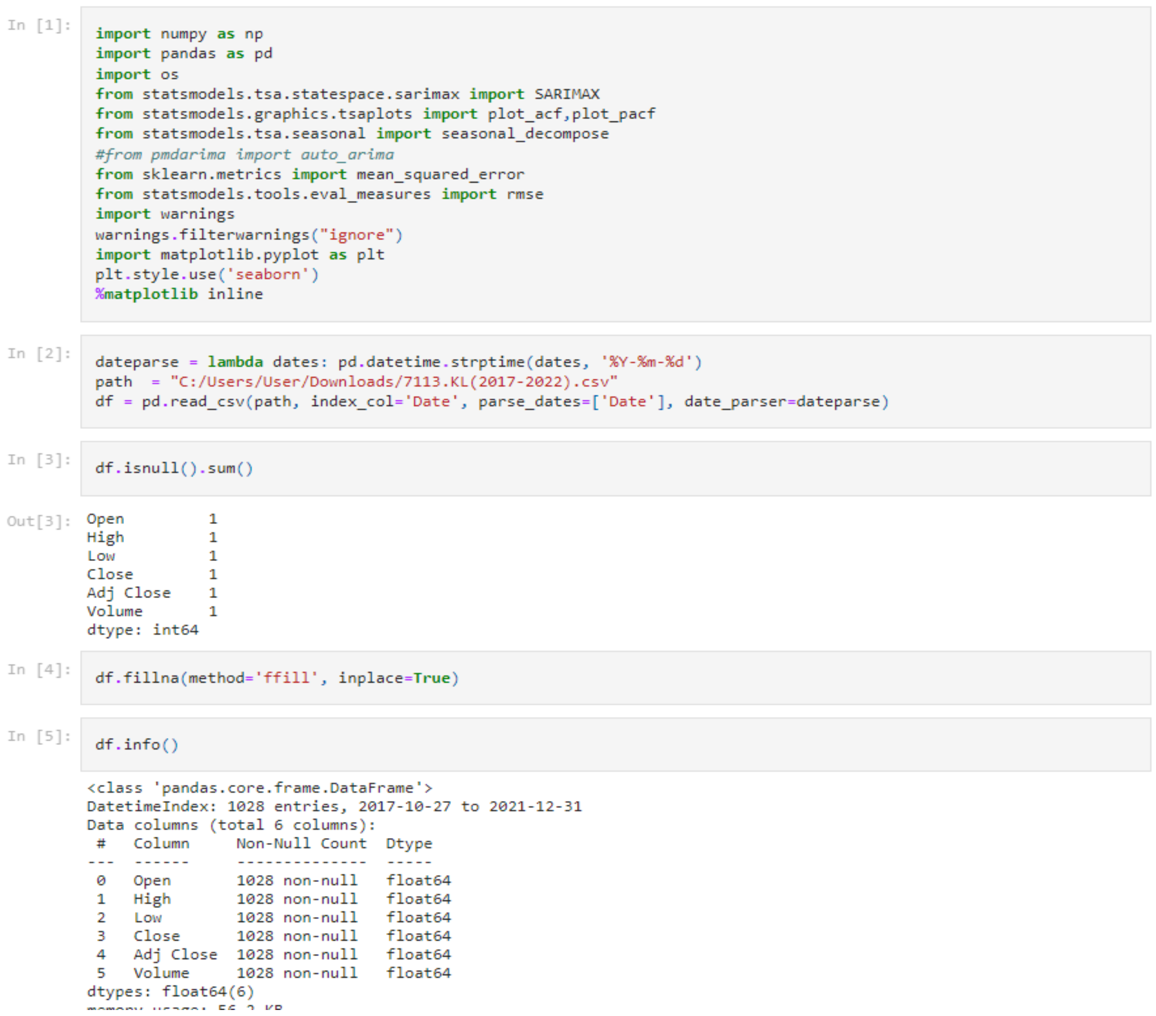
```
In [17]: plt.figure(figsize=(12,7))
         plt.plot(df['Adj Close'], 'green', color='blue', label='Training Data')
         plt.plot(valid.index,valid["Adj Close"],'red',label="Validation Data for Stock P
         plt.plot(valid.index, y_pred["ARIMA Model Prediction"],'green',label="Prediction
         plt.title('Top Glove Prices Prediction')
         plt.xlabel('Dates')
         plt.ylabel('Prices')
         plt.legend()

Out[17]: <matplotlib.legend.Legend at 0x195e79fa3a0>
```



```
In [18]: plt.figure(figsize=(12,7))
         plt.plot(valid.index,valid["Adj Close"],'red',label="Validation Data for Stock P
         plt.plot(valid.index, y_pred["ARIMA Model Prediction"],'green',label="Prediction
         plt.title('Top Glove Prices Prediction')
         plt.xlabel('Dates')
         plt.ylabel('Prices')
         plt.legend()

Out[18]: <matplotlib.legend.Legend at 0x195e7493ee0>
```



LSTM Model



```
In [1]:
import numpy as np
import pandas as pd
import os
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
from statsmodels.tsa.seasonal import seasonal_decompose
#from pmdarima import auto_arima
from sklearn.metrics import mean_squared_error
from statsmodels.tools.eval_measures import rmse
import warnings
warnings.filterwarnings("ignore")
import matplotlib.pyplot as plt
plt.style.use('seaborn')
%matplotlib inline
```

```
In [2]:
dateparse = lambda dates: pd.datetime.strptime(dates, '%Y-%m-%d')
path  = "C:/Users/User/Downloads/7113.KL(2017-2022).csv"
df = pd.read_csv(path, index_col='Date', parse_dates=['Date'], date_parser=dateparse)
```

```
In [3]:
df.isnull().sum()
```

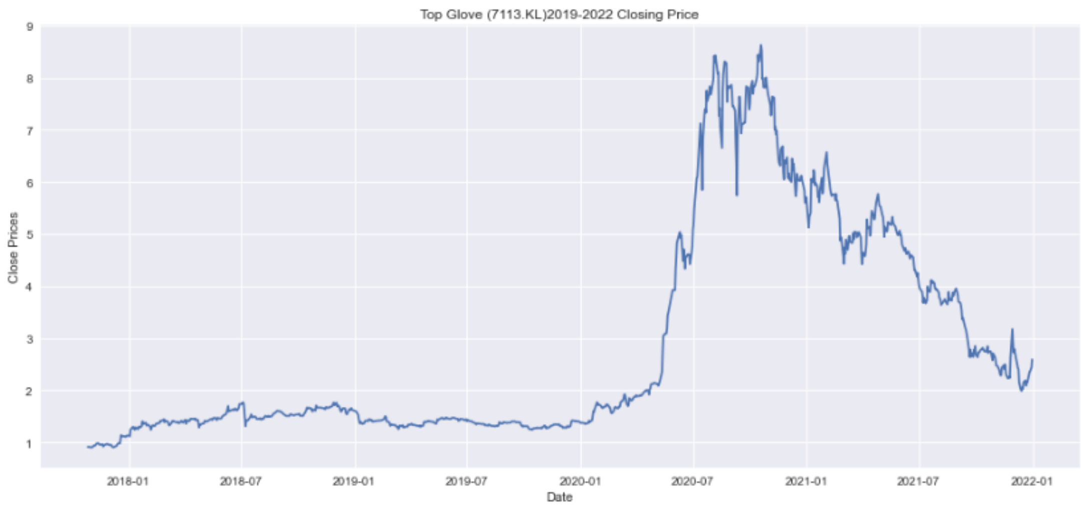
```
Out[3]:
Open         1
High         1
Low          1
Close        1
Adj Close    1
Volume       1
dtype: int64
```

```
In [4]:
df.fillna(method='ffill', inplace=True)
```

```
In [5]:
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1028 entries, 2017-10-27 to 2021-12-31
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Open       1028 non-null   float64
 1   High       1028 non-null   float64
 2   Low        1028 non-null   float64
 3   Close      1028 non-null   float64
 4   Adj Close  1028 non-null   float64
 5   Volume     1028 non-null   float64
dtypes: float64(6)
```

In [6]:
```
plt.figure(figsize=(16,7))
plt.grid(True)
plt.xlabel('Date')
plt.ylabel('Close Prices')
plt.plot(df['Adj Close'])
plt.title('Top Glove (7113.KL)2019-2022 Closing Price')
plt.show()
```



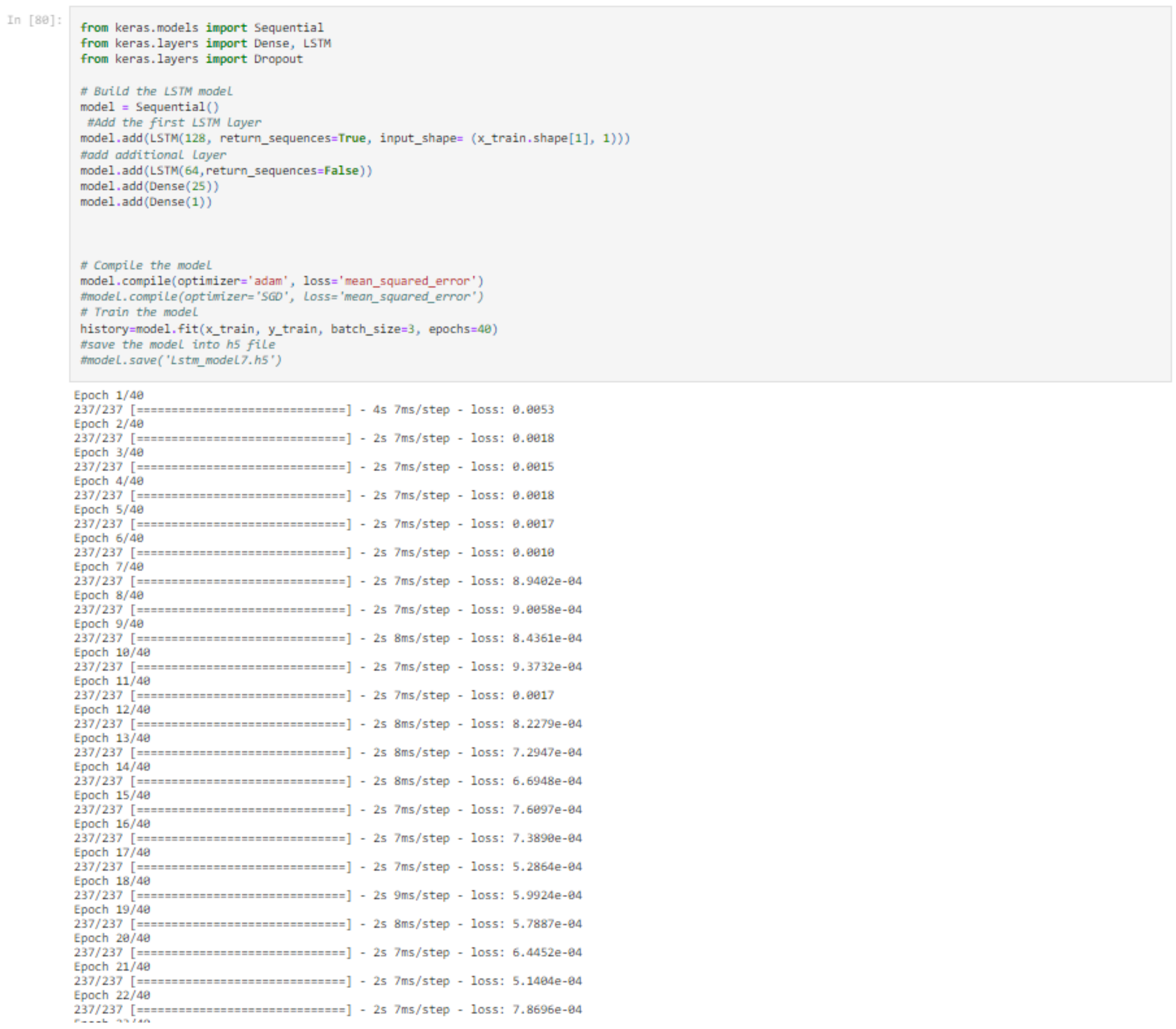
In [7]:
```
data = df.filter(['Adj Close'])
# Convert the dataframe to a numpy array
dataset = data.values
# Get the number of rows to train the model on

training_data_len = int(np.ceil( len(dataset) * .75 ))

training_data_len
```

Out[7]:
```
771
```

In [8]:
```
# Scale the data
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(dataset)

scaled_data
```

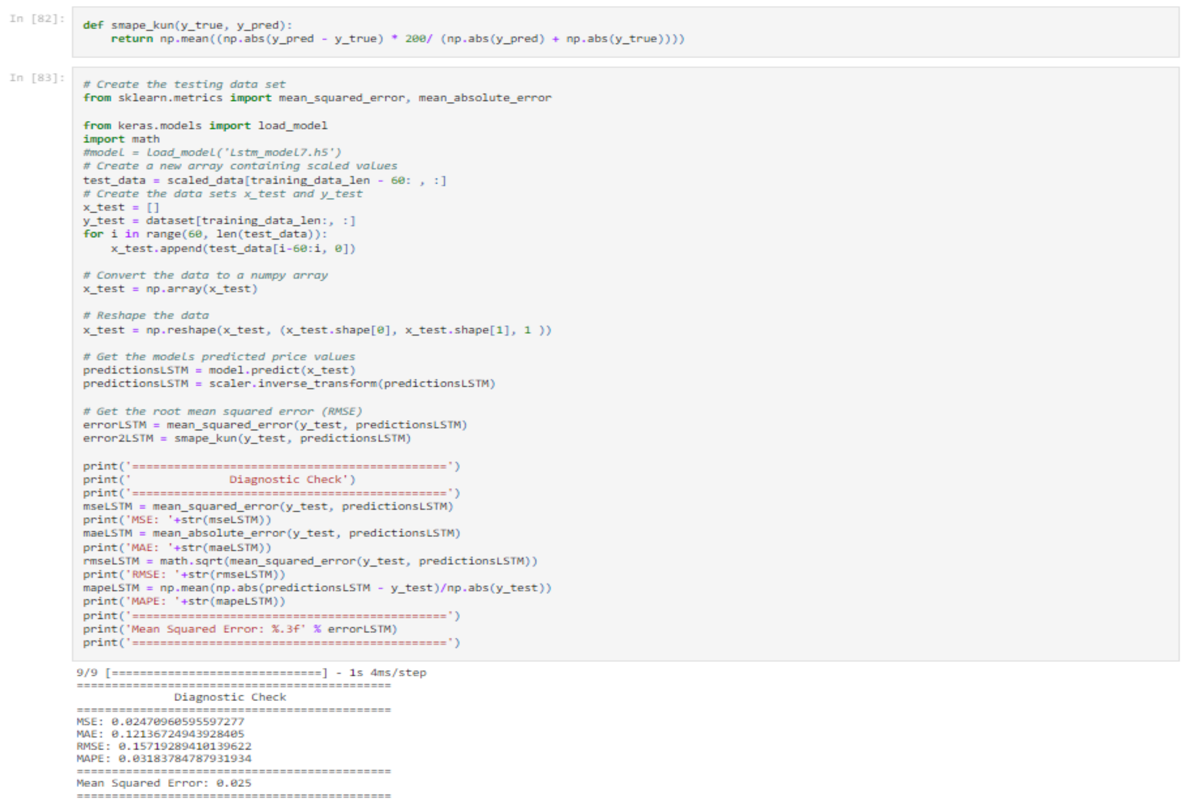
Out[8]:
```
array([[1.09904755e-03],
       [1.09904755e-03],
       [1.83131420e-04],
       ...,
       [1.95072262e-01],
       [1.98957653e-01],
       [2.18384607e-01]])
```

In [9]:
```
# Create the training data set
# Create the scaled training data set
train_data = scaled_data[0:int(training_data_len), :]
# Split the data into x_train and y_train data sets
x_train = []
y_train = []

for i in range(60, len(train_data)):
    x_train.append(train_data[i-60:i, 0])
    y_train.append(train_data[i, 0])
    if i<= 61:
        print(x_train)
        print(y_train)
        print()

# Convert the x_train and y_train to numpy arrays
x_train, y_train = np.array(x_train), np.array(y_train)

# Reshape the data
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
# x_train.shape
```



```
In [80]:
from keras.models import Sequential
from keras.layers import Dense, LSTM
from keras.layers import Dropout

# Build the LSTM model
model = Sequential()
 #Add the first LSTM layer
model.add(LSTM(128, return_sequences=True, input_shape= (x_train.shape[1], 1)))
#add additional layer
model.add(LSTM(64,return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))



# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')
#model.compile(optimizer='SGD', loss='mean_squared_error')
# Train the model
history=model.fit(x_train, y_train, batch_size=3, epochs=40)
#save the model into h5 file
#model.save('Lstm_model7.h5')
```

```
Epoch 1/40
237/237 [==============================] - 4s 7ms/step - loss: 0.0053
Epoch 2/40
237/237 [==============================] - 2s 7ms/step - loss: 0.0018
Epoch 3/40
237/237 [==============================] - 2s 7ms/step - loss: 0.0015
Epoch 4/40
237/237 [==============================] - 2s 7ms/step - loss: 0.0018
Epoch 5/40
237/237 [==============================] - 2s 7ms/step - loss: 0.0017
Epoch 6/40
237/237 [==============================] - 2s 7ms/step - loss: 0.0010
Epoch 7/40
237/237 [==============================] - 2s 7ms/step - loss: 8.9402e-04
Epoch 8/40
237/237 [==============================] - 2s 7ms/step - loss: 9.0058e-04
Epoch 9/40
237/237 [==============================] - 2s 8ms/step - loss: 8.4361e-04
Epoch 10/40
237/237 [==============================] - 2s 7ms/step - loss: 9.3732e-04
Epoch 11/40
237/237 [==============================] - 2s 7ms/step - loss: 0.0017
Epoch 12/40
237/237 [==============================] - 2s 8ms/step - loss: 8.2279e-04
Epoch 13/40
237/237 [==============================] - 2s 8ms/step - loss: 7.2947e-04
Epoch 14/40
237/237 [==============================] - 2s 8ms/step - loss: 6.6948e-04
Epoch 15/40
237/237 [==============================] - 2s 7ms/step - loss: 7.6097e-04
Epoch 16/40
237/237 [==============================] - 2s 7ms/step - loss: 7.3890e-04
Epoch 17/40
237/237 [==============================] - 2s 7ms/step - loss: 5.2864e-04
Epoch 18/40
237/237 [==============================] - 2s 9ms/step - loss: 5.9924e-04
Epoch 19/40
237/237 [==============================] - 2s 8ms/step - loss: 5.7887e-04
Epoch 20/40
237/237 [==============================] - 2s 7ms/step - loss: 6.4452e-04
Epoch 21/40
237/237 [==============================] - 2s 7ms/step - loss: 5.1404e-04
Epoch 22/40
237/237 [==============================] - 2s 7ms/step - loss: 7.8696e-04
```



```
In [82]:
def smape_kun(y_true, y_pred):
    return np.mean((np.abs(y_pred - y_true) * 200/ (np.abs(y_pred) + np.abs(y_true))))
```

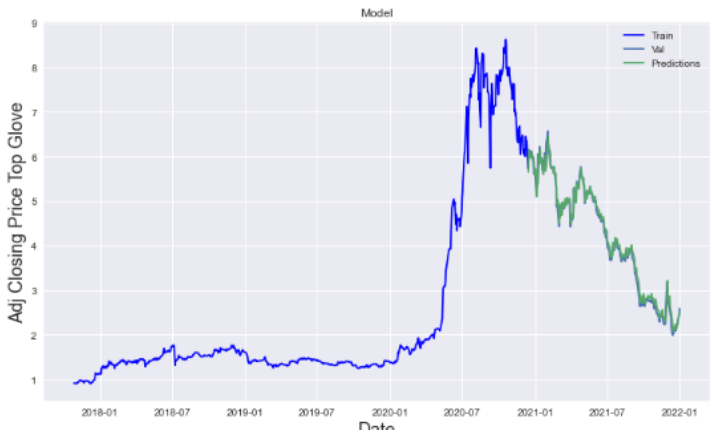
```
In [83]:
# Create the testing data set
from sklearn.metrics import mean_squared_error, mean_absolute_error

from keras.models import load_model
import math
#model = load_model('Lstm_model7.h5')
# Create a new array containing scaled values
test_data = scaled_data[training_data_len - 60: , :]
# Create the data sets x_test and y_test
x_test = []
y_test = dataset[training_data_len:, :]
for i in range(60, len(test_data)):
    x_test.append(test_data[i-60:i, 0])

# Convert the data to a numpy array
x_test = np.array(x_test)

# Reshape the data
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1 ))

# Get the models predicted price values
predictionsLSTM = model.predict(x_test)
predictionsLSTM = scaler.inverse_transform(predictionsLSTM)

# Get the root mean squared error (RMSE)
errorLSTM = mean_squared_error(y_test, predictionsLSTM)
error2LSTM = smape_kun(y_test, predictionsLSTM)

print('=============================================')
print('               Diagnostic Check')
print('=============================================')
mseLSTM = mean_squared_error(y_test, predictionsLSTM)
print('MSE: '+str(mseLSTM))
maeLSTM = mean_absolute_error(y_test, predictionsLSTM)
print('MAE: '+str(maeLSTM))
rmseLSTM = math.sqrt(mean_squared_error(y_test, predictionsLSTM))
print('RMSE: '+str(rmseLSTM))
mapeLSTM = np.mean(np.abs(predictionsLSTM - y_test)/np.abs(y_test))
print('MAPE: '+str(mapeLSTM))
print('=============================================')
print('Mean Squared Error: %.3f' % errorLSTM)
print('=============================================')
```

```
9/9 [==============================] - 1s 4ms/step
=============================================
              Diagnostic Check
=============================================
MSE: 0.02470960595597277
MAE: 0.12136724943928405
RMSE: 0.15719289410139622
MAPE: 0.03183784787931934
=============================================
Mean Squared Error: 0.025
=============================================
```

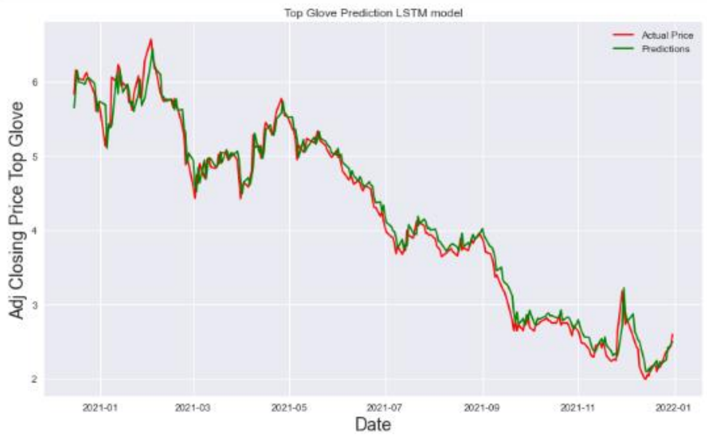
In [84]:
```
model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_10 (LSTM)              (None, 60, 128)           66560

 lstm_11 (LSTM)              (None, 64)                49408

 dense_10 (Dense)            (None, 25)                1625

 dense_11 (Dense)            (None, 1)                 26

=================================================================
Total params: 117,619
Trainable params: 117,619
Non-trainable params: 0
_________________________________________________________________
```

In [85]:
```
# Plot the data
train = data[:training_data_len]
valid = data[training_data_len:]
valid['Predictions'] = predictionsLSTM
# Visualize the data
plt.figure(figsize=(12,7))
plt.title('Model')
plt.xlabel('Date', fontsize=18)
plt.ylabel('Adj Closing Price Top Glove', fontsize=18)
plt.plot(train['Adj Close'],'green', color='blue', label='Training Data')
plt.plot(valid[['Adj Close', 'Predictions']])
plt.legend(['Train', 'Val', 'Predictions'], loc='upper right')
plt.show()
```



In [86]:
```
# Plot the data
train = data[:training_data_len]
valid = data[training_data_len:]
valid['Predictions'] = predictionsLSTM
# Visualize the data
plt.figure(figsize=(12,7))
plt.title('Top Glove Prediction LSTM model')
plt.xlabel('Date', fontsize=18)
plt.ylabel('Adj Closing Price Top Glove', fontsize=18)
plt.plot(valid['Adj Close'],color='red', label='Actual Price')
plt.plot(valid['Predictions'],color='green')
plt.legend(['Actual Price', 'Predictions'], loc='upper right')
plt.show()
```



In [87]:
```
valid[['Adj Close', 'Predictions']]
```

Out[87]:

	Adj Close	Predictions
Date		
2020-12-15	5.827806	5.648844
2020-12-16	6.155108	5.876186
2020-12-17	6.091466	6.147215
2020-12-18	6.036916	5.992418
2020-12-21	6.018733	5.973341
...	...	...
2021-12-27	2.360000	2.245670
2021-12-28	2.370000	2.416062
2021-12-29	2.410000	2.413110
2021-12-30	2.440000	2.456844
2021-12-31	2.590000	2.490339

257 rows × 2 columns

In [88]:
```
valid = valid.append(pd.DataFrame(columns=valid.columns,index=pd.date_range(start=valid.index[-1], periods=6, freq='D', closed='right')))
```

In [89]:
```
valid['2021-12-28':'2022-01-05']
```

Out[89]:

	Adj Close	Predictions
2021-12-28	2.37	2.416062
2021-12-29	2.41	2.413110
2021-12-30	2.44	2.456844
2021-12-31	2.59	2.490339
2022-01-01	NaN	NaN
2022-01-02	NaN	NaN
2022-01-03	NaN	NaN
2022-01-04	NaN	NaN
2022-01-05	NaN	NaN

In [90]:
```
upcoming_prediction = pd.DataFrame(columns=['Adj Close'],index=valid.index)
upcoming_prediction.index=pd.to_datetime(upcoming_prediction.index)
```

In [91]:
```
curr_seq = x_test[-1:]

for i in range(-5,0):
    up_pred = model.predict(curr_seq)
    upcoming_prediction.iloc[i] = up_pred
    curr_seq = np.append(curr_seq[0][1:],up_pred,axis=0)
    curr_seq = curr_seq.reshape(x_test[-1:].shape)
```

```
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
```

In [92]:
```
upcoming_prediction[['Adj Close']] = scaler.inverse_transform(upcoming_prediction[['Adj Close']])
```

In [93]:
```
upcoming_prediction['2022-01-01':'2022-01-05']
```

Out[93]:

	Adj Close
2022-01-01	2.490339
2022-01-02	2.544307
2022-01-03	2.599296
2022-01-04	2.654413
2022-01-05	2.709592

In [94]:
```
upcoming_prediction['2022-01-01':'2022-01-05']
```

Out[94]:

	Adj Close
2022-01-01	2.490339
2022-01-02	2.544307
2022-01-03	2.599296
2022-01-04	2.654413
2022-01-05	2.709592