FORECASTING CLOSING PRICE OF GOOGLE STOCKS USING AUTO ARIMA: A TIME SERIES ANALYSIS CASE STUDY

by

Angelo M. Salas
Ayn Vencent D. Delegero
Kervin Carl E. Maraviles
Veronica Ann T. Balaba

A Case Study Submitted to the Department of Mathematics and Statistics
in Partial Fulfillment of the Requirements for

MATH QEC2 (Time Series Analysis)

Polytechnic University of the Philippines
July 2023

TABLE OF CONTENTS

LIST OF TABLES

Chapter 3

LIST OF FIGURES

Chapter 3

ABSTRACT

The stock market is renowned for its unpredictability, complexity, and ever-changing nature, making accurate stock price forecasting a challenging task. This case study explores the application of the Autoregressive Integrated Moving Average (ARIMA) model to predict the closing price of Google's stocks, shedding light on the strengths and weaknesses of this methodology. The study aims to identify seasonality in historical data, determine the optimal ARIMA model, and forecast the closing price for the next six months. The analysis utilizes a dataset spanning from 2013 to 2023, provided by Alphabet Inc. The ARIMA(1,1,1)(0,0,0)[0] model is identified as the best model. While the study provides valuable insights and enhances our understanding of stock price dynamics, it should be acknowledged that unforeseen events and market changes can affect the accuracy of predictions. The study's findings can benefit investors, financial analysts, and researchers seeking to employ time series analysis techniques for stock price prediction, while also emphasizing the potential of ARIMA models in understanding the dynamic nature of the stock market. The study concludes with evaluation metric results, demonstrating the model's performance and indicating its limitations.

Keywords: stock price prediction, ARIMA model, time series analysis, Google stock, forecasting

Chapter 1

INTRODUCTION

The stock market is known for being unexpected, non-linear, and dynamic. The art of predicting stock values is a difficult undertaking because it depends on a variety of circumstances, including the political climate, the global economy, reports on the company's performance and finances, etc. The ability to accurately forecast stock prices has long been a subject of intense interest and relevance in financial markets. Investors, analysts, and researchers continually seek effective methodologies to predict the future movements of stock prices, aiming to make informed investment decisions and manage risk effectively. In pursuit of reliable forecasting techniques, time series analysis has emerged as a powerful tool to capture and model the intricate patterns inherent in stock price data.

In the paper of Mehar Vijh et al. (2020), it was stated that most of the previous forecasting in this area used classical algorithms like linear regression. Some of them are Random Walk Theory (RWT) and Moving Average Convergence/Divergence (MACD). However, in this study, the researchers will employ the ARIMA model to predict the closing price of Google's stocks, Alphabet Inc.'s largest subsidiary. The use of ARIMA models for stock price prediction has been well studied in the literature, and this case study plans to present a concrete example of the weaknesses and strengths of the said model.

In conclusion, this case study provides a thorough examination of the ARIMA model's use in predicting Google stock prices, demonstrating its potential as a useful tool for financial analysts. The study intends to offer useful insights, create a deeper understanding of stock price dynamics, and equip stakeholders with strong forecasting abilities in the

constantly changing context of financial markets by studying the complex nature of time series analysis.

Statement of the Problem

The purpose of the study was to analyze the Closing price of Google stock historical data using time series ARIMA model and to forecast its future values. Specifically, the study aimed to:

1. Check for seasonality in the historical data based on graph observation.
2. Determine the best ARIMA model for forecasting the Closing price, and use this model to predict the Closing price for the next 6 months.
3. Derive key insights from the analysis.

Hypothesis

H_0**:** The data is non-stationary.

To determine whether the data is stationary (i.e., reject the null hypothesis), the p-value should be less than or equal to the significance level of 0.05 after the Augmented Dickey-Fuller (ADF) test.

Conversely, if the p-value appears to be greater than the significance level, then it suggests that there is sufficient evidence that the data is likely non-stationary (i.e., failed to reject the null hypothesis).

Scope and Limitations of the Study

This study is focused on predicting the Closing price of the Google stock's historical data for the next 6 months. This study will use the dataset obtained by Alphabet Inc. from 2013 to 2023 and apply the Autoregressive Integrated Moving Average (ARIMA).

Any errors or inconsistencies in the dataset may affect the accuracy of the forecasted values in this study. Unforeseen events or changes within the market dynamics could also impact the accuracy of the predicted values. The predictions are solely based on historical patterns and statistical models, so there is always a degree of uncertainty associated with the future forecasts. Thus, this study should only be viewed as a tool to assist in the decision-making rather than a definitive prediction of future stock prices.

Significance of the Study

This case study can be a great help to the investors, financial analysts, and researchers that are trying to employ time series analysis techniques for predicting stock prices. It can also provide insights on the effectiveness of ARIMA models in determining the dynamic nature of the stock market and how well they can produce reliable forecasts in the context of Google stocks.

Chapter 2

METHODOLOGY

This chapter presents the process and methods used by the researchers to obtain their desired result. This includes the preparation of the data and the experiment design, which tackles the software details, the experiment model used, the experiment procedure, and the performance criteria.

Data Preparation

The researchers will be working with the Google 10 years stock price dataset from Kaggle. The dataset provides historical stock data for Alphabet Inc. (GOOG), such as the opening, high, low, and close prices of stock from January 2013 to January 2023. The data is available on a daily basis and is presented in US dollars.

Experiment Design

This section discusses the entire process of the study that will be carried out by the researchers. This includes the discussion of the software to compute the forecasted close price of Google stocks for the next 6 months and the statistical analysis model that will be utilized during the procedure.

Software Details

The researchers will mainly use the Python 3.11-based Jupyter Notebook software. The main packages needed for this study are Pandas, which is crucial for handling data; NumPy, which

provides the mathematical modules needed for Python computation; SciPy, which provides features for this study's optimization; and Statsmodels and Matplotlib, which provide statistical modeling and result visualization.

Experiment Model

An ARIMA model is a class of statistical analysis models that use time series data to either better understand the dataset or predict future trends. It specifically addresses a number of common time series data types and, as a result, offers a straightforward but effective technique for producing accurate time series forecasts.

ARIMA is an acronym that stands for Autoregressive Integrated Moving Average. It is the combination of autoregression (AR), integration (I), and the moving average (MA).

- **Autoregression (AR).** A model where a variable is changing and regresses on its own lagged, or prior, values.
- **Integrated (I).** The use of differencing of raw observations (i.e., data values are replaced by the difference between the data values and the previous values) in order to make the time series stationary.
- **Moving Average (MA).** A model that incorporates the dependency between an observation and a residual error from a moving average model applied to lagged observations.

Experiment Procedure

This section presents the implementation of ARIMA using Python 3.11-based Jupyter Notebook in the stock prices of Google.

Libraries

The following are the necessary libraries that have to be installed and imported in the Jupyter Notebook for ARIMA to function properly for this study:

- **Warning.** Filters possible warnings that may appear when you run the data in Python.
- **Matplotlib.pyplot.** Imports collections of command style functions for data visualization in various formats. Using pyplot functions, Python can create figures as well as plot areas and add figures with labels, lines, and axes, which is needed for this study to visualize the outcome.
- **Numpy**. Provides data structures to Python for calculations of arrays and matrices and even complex functions that will be needed to calculate the given data in this study.
- **Pandas.** A Python library for data wrangling, data manipulation, and data analysis. It is very important for the entire process of collecting the data for this study.
- **Statsmodels.** For statistical data exploration and statistical modeling.
- **Pmdarima**. A statistical library composed of auto – ARIMA functions, tests for data stationarity and seasonality, and other time series analysis tools.
- **Auto–ARIMA**. From pmdarima; used to test data stationarity and seasonality, as well as seasonal time series decompositions.
- **Sklearn.** A toolkit for science Python that is used for statistical modeling, machine learning, and deep learning.
- **Metrics.** Module from Sklearn that implements functions that will assess prediction errors.

- **Seasonal decompose.** Need to import for the SARIMA that the researchers will be doing.
- **Adfuller**. Used to test the stationarity of the data using p – values and critical values at given confidence intervals. This test is needed for the study to reject or not reject the hypothesis.

After the researchers have imported the necessary libraries for the study, the data must be imported and loaded into Python in order to run the time series analysis on the data. Since the data was stored in a CSV file, Python was able to read the CSV file and input the data's location from your computer's files. The information is arranged based on its time-frequency distribution. To check whether the correct data was successfully imported for this study, the researchers can display the first ten rows of the data in tables.

Exploratory Data Analysis

After importing and loading the data, the researchers then visualized the utilized time series data of the study. In order to know and understand whether the data contained a pattern, they devised a graphical representation for exploratory data analysis. Allowing the researchers to see the trend of their data through the graphical depiction and evaluate whether it contains seasonal, cyclical, or irregular components.

Kernel Density Estimation

The probability density function (PDF) of a random variable can be estimated in a non-parametric manner using kernel density estimation (KDE). By using KDE for PDF

estimation, one can ascertain the underlying distribution of the data, spot patterns, identify peaks or modes, and estimate probabilities at various values of the variable. Line graphs and histograms are frequently used in Python to illustrate this.

Seasonal-Trend Decomposition

Researchers may dissect the trend and seasonal components of the data in the JUPYTER Notebook in order to comprehend the underlying issues in the time series and forecasting processes, which will help them determine whether the data contains trend and seasonal components. They will also be able to distinguish between the linear trend, seasonal components, and random residuals in the JUPYTER Notebook and arrange them according to the data in a periodic fashion. If they wish to more clearly see the relationship between the independent and dependent variables, they can additionally deduct the seasonality components from the time series' initial value. They can also use Python to visually illustrate the seasonal and trend decomposition of Google's close stock prices in graphs.

Time Series Evaluation Function

Time series forecasting tools are tested for their ability to predict future values based on their performance. To assess the performance and accuracy of these time series forecasting tools, the researchers utilized time series evaluation functions in their study. By comparing the different metrics (e.g., Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), and R-squared (R2)) across different models or techniques, the researchers can determine which approach provides the best prediction for the close prices of Google's stocks.

Augmented Dickey-Fuller Test (ADF) with Akaike information criterion

To determine whether to reject this study's hypothesis or not, the researchers run an Augmented Dickey-Fuller (ADF) Test with Akaike Information Criterion (AIC) to the data of Google's close stock prices. An Augmented Dickey-Fuller (ADF) Test is a statistical test used to determine the stationarity of a time series. Many time series analysis techniques make stationarity an essential presumption because it enables accurate statistical inference and modeling. On the other hand, the researchers also incorporated the said test with Akaike Information Criterion (AIC), which determines the presence of a minimal error in the forecast.

Train and Test Sets

In this study, the researchers will also do the assignment of the train and test set. Splitting the data in such a manner is considered as a crucial step in the time series forecasting process , as it evaluates the performance of a model when applied to unseen or new data. As a result, there are less data discrepancies , and the model's performance in forecasting the given data is better understood..

Usually, the researchers divided the data into train sets and test sets using a random sample of the data. The model is then trained on the training sets and tested on the testing sets. However, in this study, since the data are available on a daily basis, the researchers decided to assign the last 30 rows of the data set as the test set to be assessed using the train set's predictions while the rest of the data will be on the train set.

Auto-ARIMA

In this study, since there's no seasonality detected in the dataset the researchers proceed to employ the auto-ARIMA function to automate the parameter selection procedure and identify the best ARIMA model parameter for the utilized dataset. With the aid of the said function, the researchers were able to determine the best-fitting model for the used data by examining various combinations of parameters within a fixed range.

The researchers then fitted the ARIMA model using the training data after identifying the best parameters. Using the test data, the performance of the fitted model was assessed. Prediction accuracy was measured using metrics including mean squared error (MSE), mean absolute error (MAE), root mean square error (RMSE), mean absolute percentage error (MAPE), and R-squared (R2). After the model is validated, it is then used to predict the close prices of Google's stocks based on 30-days test data. The researchers then plot those predicted values to visually assess the model's effectiveness.

Diagnostic Checking

In this study, researchers will use a stepwise approach to further assess the model's performance. In this approach, the model is evaluated based on the diagnostic plots where insights into the residuals, normality, distribution, autocorrelation, and other statistical properties are provided. These plots can significantly help the researchers to examine if the assumption criteria within the model needs to be improved.

Forecast

In the forecasting phase, the researchers identified and defined the metrics to be used to evaluate the model by using Auto-ARIMA. To enhance the comprehensiveness of the time series data, confidence intervals were also added. Confidence interval gives a range where the actual values are expected to fall, considering the uncertainty associated with the forecasted values.

To visualize the forecasted values, this study utilized plots and graphs in python. On the other hand, by evaluating the accuracy of the forecasts, the researchers can utilize evaluation metrics chosen for this study. The forecasted values can then be visually compared with the test set, prediction values, and training set using appropriate visualization techniques. Prediction accuracy was also measured.

By running the optimal Auto-ARIMA model through a stepwise model, the researchers may predict the future values of closing prices of Google's stocks. The result of the findings from the stepwise model may also serve as an aid for the selection of the optimal Auto-ARIMA model for generating forecasts.

The forecasting process demonstrates a systematic approach, incorporating statistical techniques, model selection, evaluation of metrics, and visualization of data to generate reliable predictions for the future closing prices of Google's stock price.

Chapter 3

RESULTS AND DISCUSSIONS

Results and Discussions

This chapter presents the data gathered, the results of analysis performed, and the interpretation of the findings.

The figure below is the line graph of the Google stock's Close price. The given set of data in Figure 1 is used to develop a forecasting model.

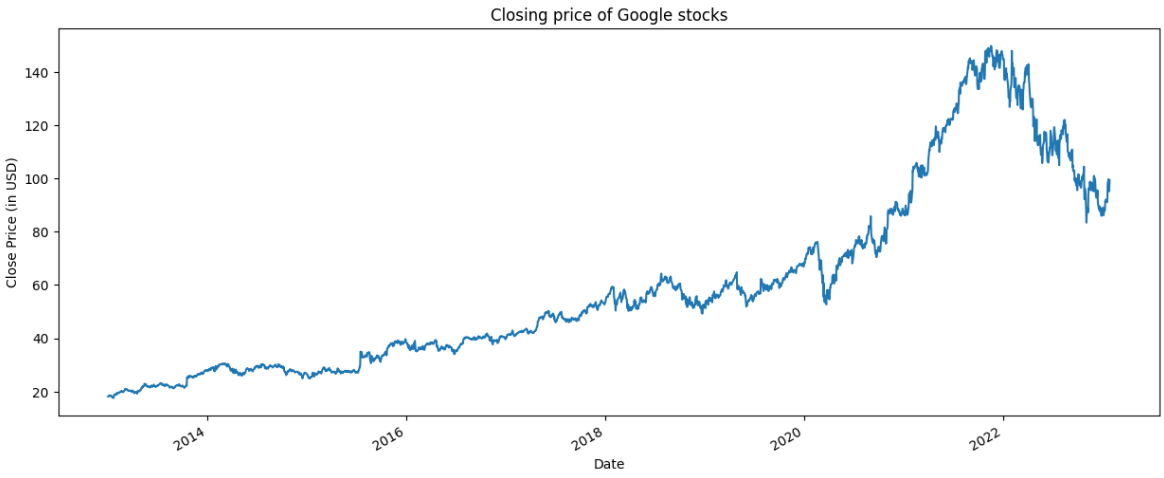


Figure 1. The Close price of Google stock (in U.S. Dollar) from January of 2013 to January of 2023.

The y-axis of the line graph above corresponds to the Close price of Google stock (in USD) , and the x-axis corresponds to the span of the dataset specifically from January of 2013 up until January of 2023. ARIMA was chosen as the best method in creating the best model for forecasting since the time series plot of the historical data shows gradual shifts from low to high for a long period of time.

The figure below is the additive decomposition of the dataset. In Figure 2, the Seasonal Decomposition method is used to estimate the components.

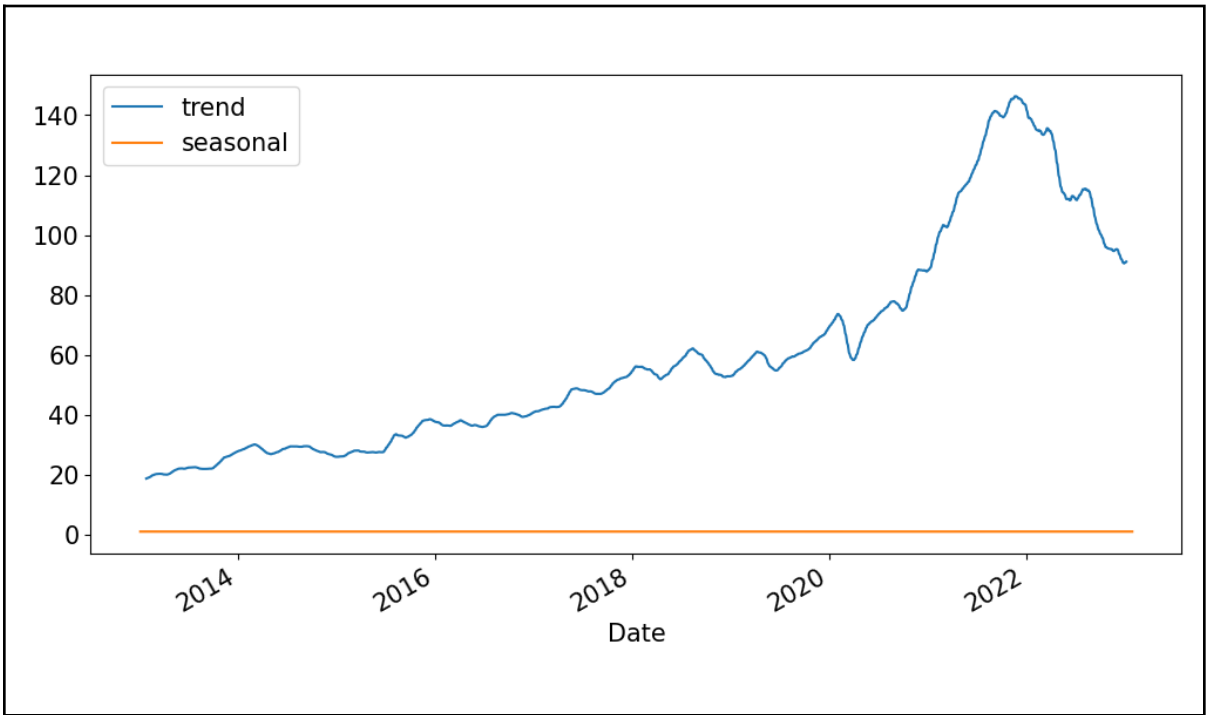


Figure 2. the Seasonal Decomposition

The two components are displayed in Figure 2. The data seen in Figure 1 can be reconstructed by combining these elements. The seasonal component (orange line) may be seen to be a flat horizontal line since the dataset has no seasonality. The data from the other component (blue line) has been shown to be the same as the original dataset line graph, which is the trend component.

Histogram and Kernel Density Estimation (KDE) of the data

The Close price of the Google stock is displayed in Figure 3 along basic exploratory data analysis utilizing line, histogram, and kernel density estimation. Exploratory data

analysis is used to analyze and investigate datasets as well as summarize their main characteristics, often employing data visualization methods.

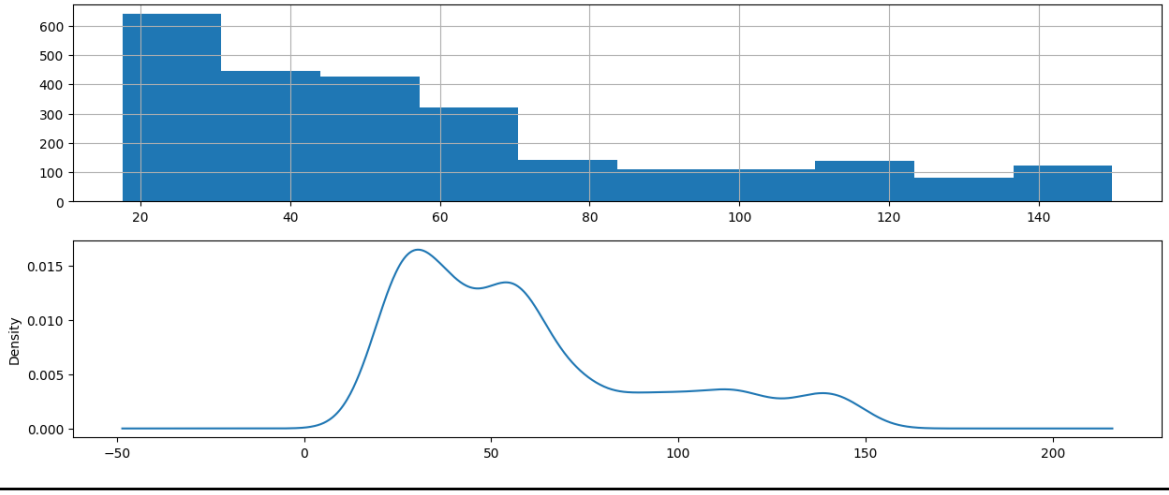


Figure 3. Histogram and KDE

The histogram in the first graph displays the Google stock's Close price frequency bars. The second graph in the figure displays the kernel density estimate plot as essentially a smoothed histogram. The density is determined by the likelihood of having that particular Close price at that moment. The higher the wave on a particular x-axis means the more it is probable to happen. For this dataset, it is more likely to occur when the close price is $20 to $40.

The Augmented Dickey-Fuller Test (ADF) with Akaike Information Criterion

In order to determine whether the data is stationary, the Augmented Dickey-Fuller Test (ADF) test function is used. The time series is classified as stationary if the critical values at the 1 percent, 5 percent, and 10 percent confidence intervals are as close as feasible to the ADF Statistics and the p-value is lower than 0.05.

Table 1. ADF Test	
Results of Dickey-Fuller Test for column: Close	
Test Statistic	-0.776298
p-value	0.825986
No Lags Used	23.000000
Number of Observation Used	2512.000000
Critical Value (1%)	-3.432956
Critical Value (5%)	-2.862691
Critical Value (10%)	-2.567383
dtype: float64 **Conclusion:====>** **Fail to reject the null hypothesis** **Data is non-stationary**	

The table 1 shows that the p-value for the time series data is 0.825986, that is greater than 0.05, indicating failure to reject the null hypothesis thus, the data is non-stationary. Automatic model identification will be applied to make the data stationary. Internally, it will be handled with, and look for several models that will make the time series data stationary.

Model Selection

Results of the Stepwise Model are discussed in the section that follows. The first phase in developing an ARIMA model for a variable's forecasting is model identification. The second step is parameter estimation and selection. The third step is diagnostic checking (or modal validation). Followed by the use of the model for forecasting applications.

Table 2. Stepwise Model	
Performing stepwise search to minimize aic	
ARIMA(1,1,1)(0,0,0)[0] intercept	: AIC=8324.454, Time=2.72 sec
ARIMA(0,1,0)(0,0,0)[0] intercept	: AIC=8335.353, Time=0.14 sec
ARIMA(1,1,0)(0,0,0)[0] intercept	: AIC=8328.448, Time=0.55 sec
ARIMA(0,1,1)(0,0,0)[0] intercept	: AIC=8328.368, Time=0.50 sec
ARIMA(0,1,0)(0,0,0)[0]	: AIC=8334.826, Time=0.14 sec
ARIMA(2,1,1)(0,0,0)[0] intercept	: AIC=8325.608, Time=3.55 sec
ARIMA(1,1,2)(0,0,0)[0] intercept	: AIC=8325.688, Time=2.39 sec
ARIMA(0,1,2)(0,0,0)[0] intercept	: AIC=8330.289, Time=0.84 sec
ARIMA(2,1,0)(0,0,0)[0] intercept	: AIC=8330.420, Time=0.60 sec
ARIMA(2,1,2)(0,0,0)[0] intercept	: AIC=8325.851, Time=5.82 sec
ARIMA(1,1,1)(0,0,0)[0]	: AIC=8324.743, Time=1.23 sec
Best model: ARIMA(1,1,1)(0,0,0)[0] intercept **Total fit time: 18.566 seconds**	

The data of the Close price of Google stock is separated into two categories: train and test data. The train set will include all the data except for the last 30 days, while the test set will contain only the last 30 days of the data to evaluate against the forecasts. The iterations are limited to p=7 and q=7 and the stopping method is utilized to prevent the overfitting problem. The Table 2 above has shown that the best ARIMA model for forecasting the Close price of Google stock is ARIMA(1,1,1)(0,0,0)[0] intercept with the minimum aic score=8324.454. The summary statistics of the stepwise model above is shown in Table 3.

Table 3. Stepwise Model Summary						
Dep. Variable:		y		**No. Observations:**		2506
Model:		SARIMAX(1, 1, 1)		**Log Likelihood:**		-4158.227
Date:		Sat, 15 Jul 2023		**AIC**		8324.454
Time:		11:41:54		**BIC**		8347.758
Sample:		0		**HQIC**		8332.914
		- 2506				
Covariance Type:		opg				
	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
ar.L1	0.0079	0.006	1.268	0.205	-0.004	0.020
ma.L1	0.7440	0.059	12.595	0.000	0.628	0.860
ar.S.L4	-0.7955	0.054	-14.652	0.000	-0.902	-0.689
sigma 2	1.6194	0.019	84.031	0.000	1.582	1.657
Ljung-Box (L1)(Q):		0.36		**Jarque-Bera (JB):**		9025.53
Prob(Q):		0.55		**Prob(JB):**		0.00
Heteroskedasticity(H):		20.34		**Skew:**		-0.12
Prob(H) (two-sided):		0.00		**Kurtosis:**		12.30

The coefficients of AR and MA are seen in Table 3. The p-values of ma.L1, ar.S.L4, sigma 2 terms are highly significant since their values are less than 0.05, while the p-value of ar.L1 is not statistically significant since the p-value=0.205 is not less than 0.05.

Table 4. Test set evaluation metrics	
MSE	**38.56767982424719**
MAE	**5.598494093992771**
RMSE	**6.2102882239270665**
MAPE	**6.26573810822338**
R2	**-1.4829164864241617**

In Table 4, this demonstrates the Test set evaluation metrics wherein presents the values of different forecasting accuracy measures. For MSE or the Mean Squared Error, it has 38.57% forecasting error. While the MAE, RMSE, and MAPE have only 5.6%, 6.21%, and 6.27% forecasting error respectively.

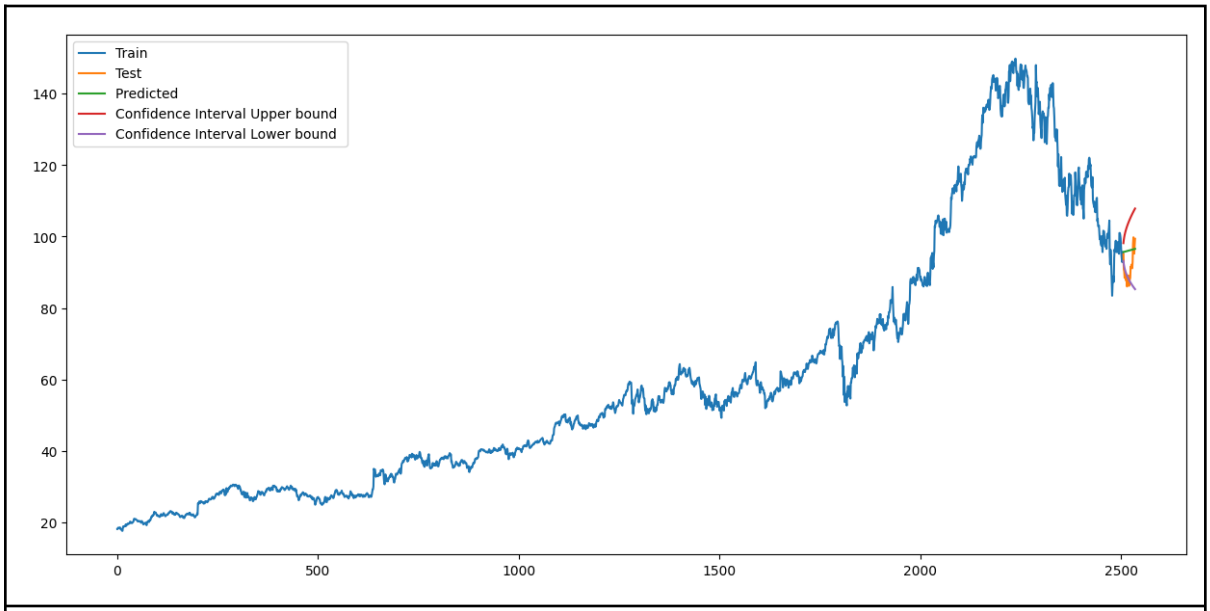


Figure 4. Train, Test, and Prediction Plot

Figure 4 displays the plot of the train, test, and predictions, as well as the upper and lower boundaries of the confidence interval. However, the model does not appear to provide

a directionally accurate forecast based on the chart. Each of the predicted forecasts is not consistently close to the test values. That means the model does not fit the data well.

Diagnostic Checking

To examine the distribution of forecasting errors (standard residuals), the errors are plotted in different ways. Figures 5(a), 5(b), 5(c), and 5(d) show various plots and histograms of standard residuals of the ARIMA(1,1,1)(0,0,0)[0] model:

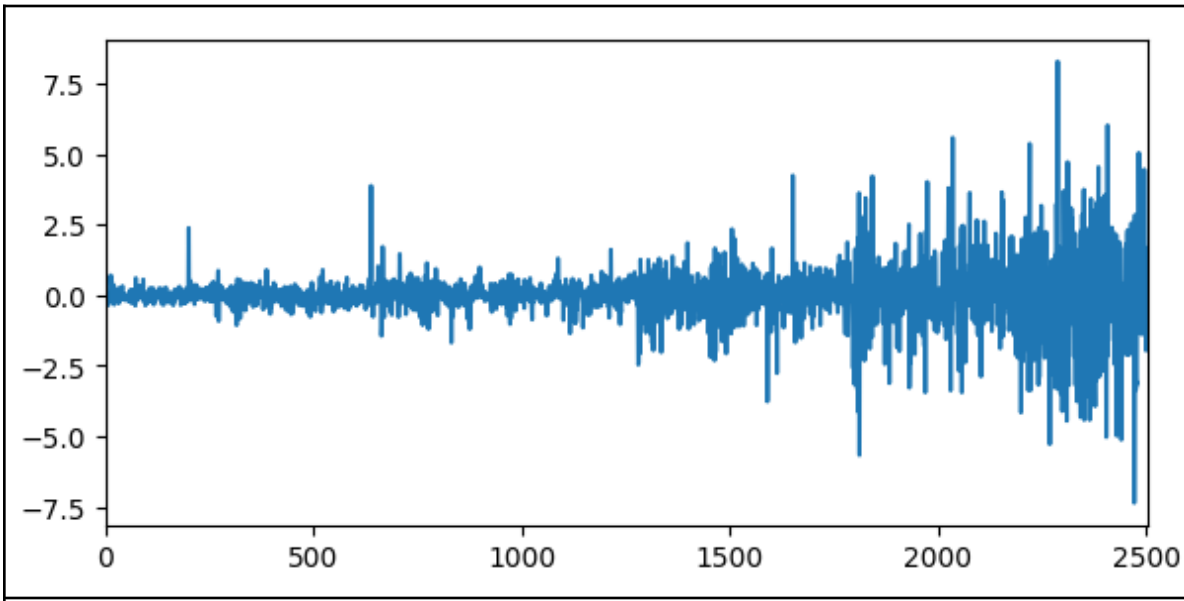


Figure 5(a). Standardized Residual

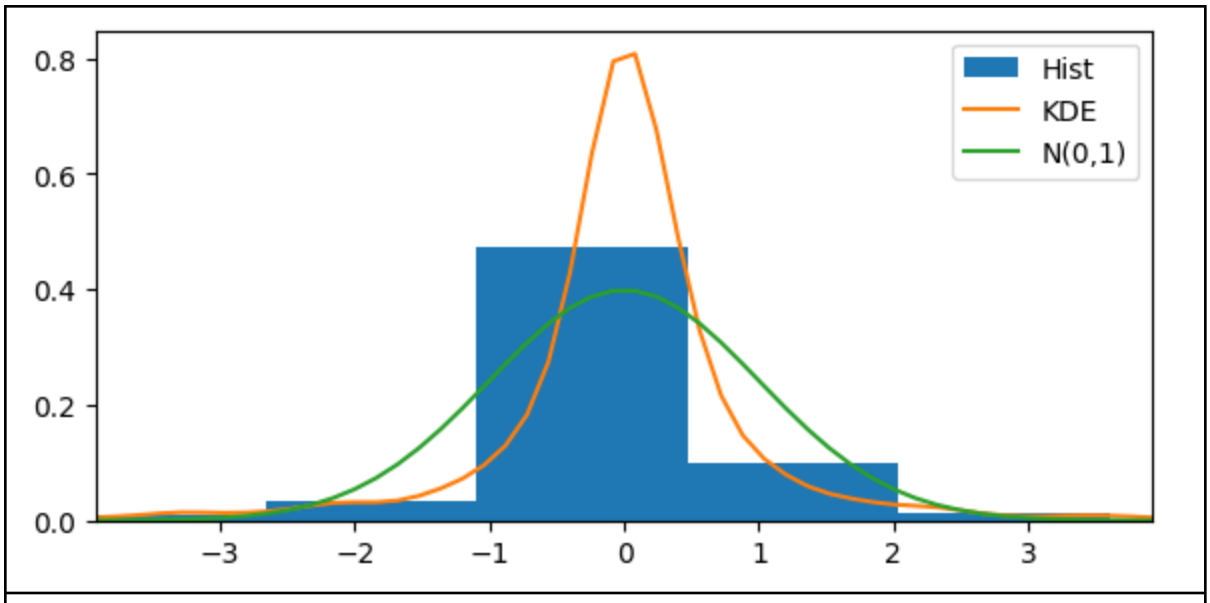


Figure 5(b). Histogram plus Estimated Density

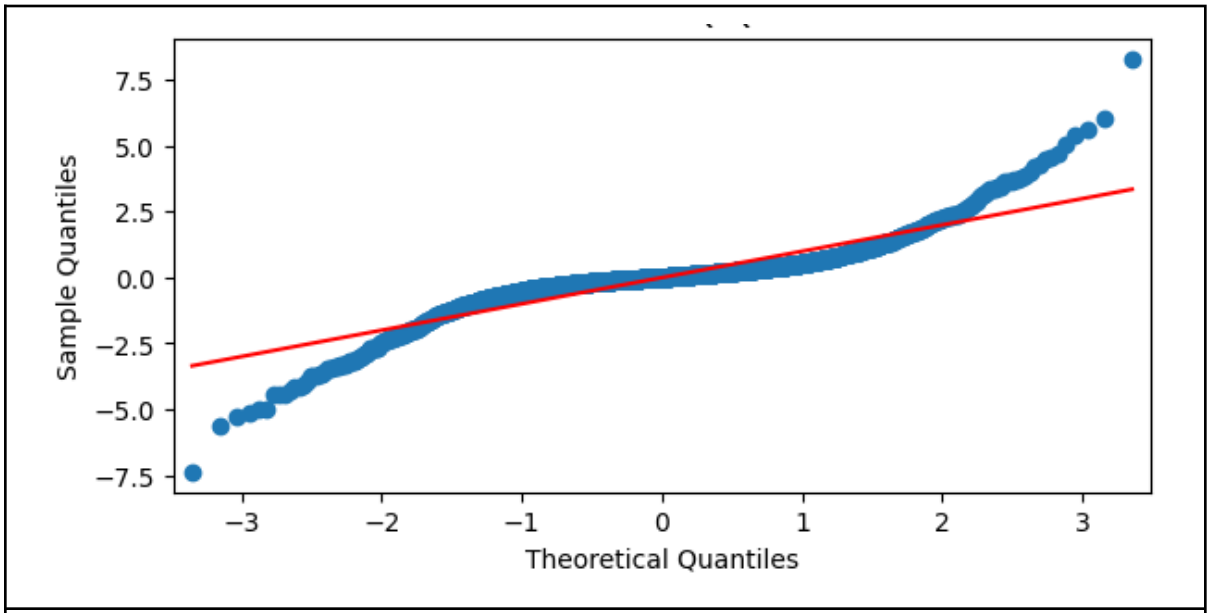


Figure 5(c). Normal Q-Q

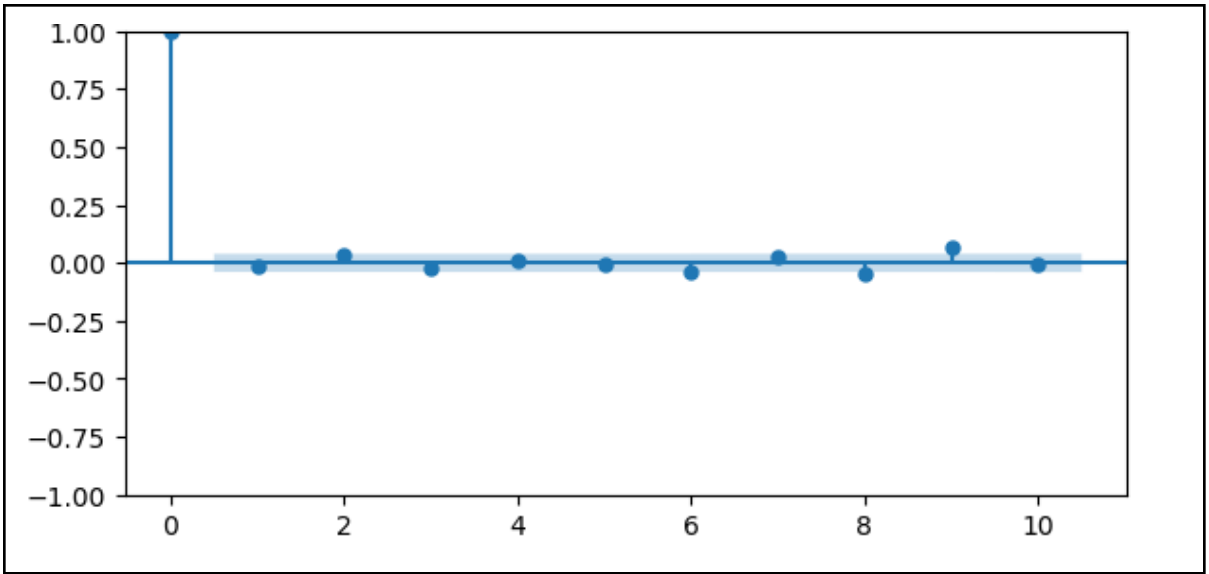


Figure 5(d). Correlogram

Forecasting

The ARIMA(1,1,1)(0,0,0)[0] model, as shown in Table 5, had been used to forecast values for the next 6 months from January 28, 2023 to July 31, 2023. The entire forecasting plot had been depicted in Figure 6.

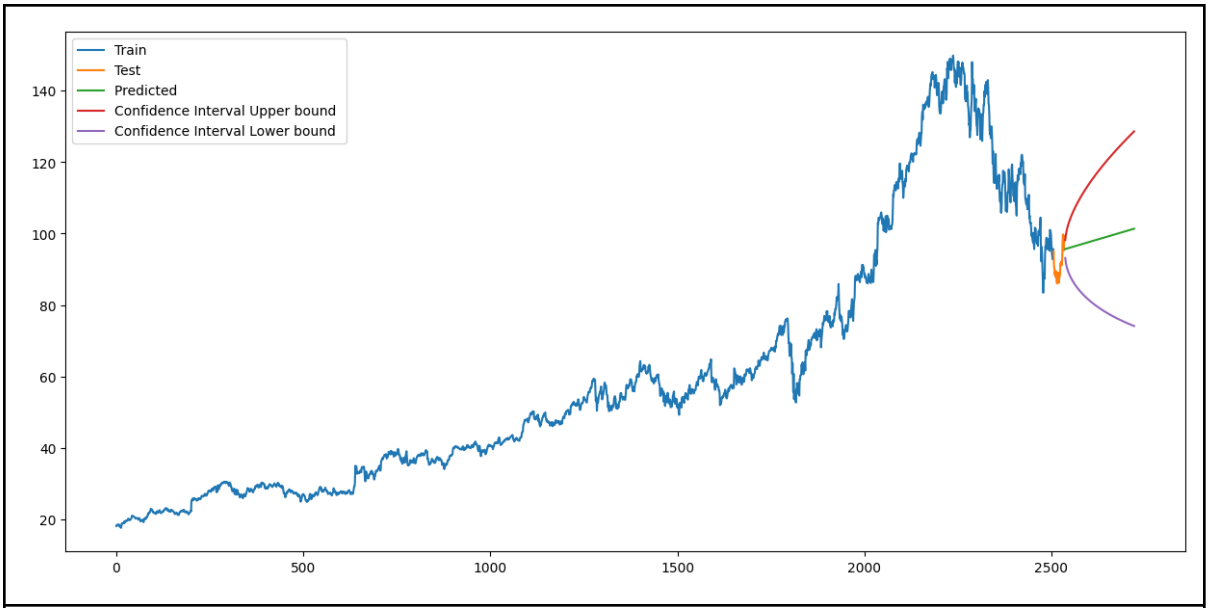


Figure 6. Time Series Plot of the Original Data and Forecasted Values

Table 5. The Forecasting Result

Month	Day	Prediction
January	28	95.660749
	29	95.691544
	30	95.722373
	31	95.753225
February	1	95.784096
	2	95.814980
	3	95.845874
	4	95.876776
	5	95.907683
	6	95.938595
	7	95.969509

	8	96.000426
	9	96.031344
	10	96.062264
	11	96.093184
	12	96.124106
	13	96.155028
	14	96.185950
	15	96.216872
	16	96.247795
	17	96.278718
	18	96.309641
	19	96.340564
	20	96.371487
	21	96.402411
	22	96.433334
	23	96.464257
	24	96.495180
	25	96.526104
	26	96.557027
	27	96.587950
	28	96.618874
March	1	96.649797
	2	96.680720
	3	96.711644
	4	96.742567

	5	96.773491
	6	96.804414
	7	96.835337
	8	96.866261
	9	96.897184
	10	96.928107
	11	96.959031
	12	96.989954
	13	97.020877
	14	97.051801
	15	97.082724
	16	97.113648
	17	97.144571
	18	97.175494
	19	97.206418
	20	97.237341
	21	97.268264
	22	97.299188
	23	97.330111
	24	97.361034
	25	97.391958
	26	97.422881
	27	97.453805
	28	97.484728
	29	97.515651

	30	97.546575
	31	97.577498
April	1	97.608421
	2	97.639345
	3	97.670268
	4	97.701191
	5	97.732115
	6	97.763038
	7	97.793962
	8	97.824885
	9	97.855808
	10	97.886732
	11	97.917655
	12	97.948578
	13	97.979502
	14	98.010425
	15	98.041349
	16	98.072272
	17	98.103195
	18	98.134119
	19	98.165042
	20	98.195965
	21	98.226889
	22	98.257812
	23	98.288735

	24	98.319659
	25	98.350582
	26	98.381506
	27	98.412429
	28	98.443352
	29	98.474276
	30	98.505199
May	1	98.536122
	2	98.567046
	3	98.597969
	4	98.628892
	5	98.659816
	6	98.690739
	7	98.721663
	8	98.752586
	9	98.783509
	10	98.814433
	11	98.845356
	12	98.876279
	13	98.907203
	14	98.938126
	15	98.969049
	16	98.999973
	17	99.030896
	18	99.061820

	19	99.092743
	20	99.123666
	21	99.154590
	22	99.185513
	23	99.216436
	24	99.247360
	25	99.278283
	26	99.309206
	27	99.340130
	28	99.371053
	29	99.401977
	30	99.432900
	31	99.463823
June	1	99.494747
	2	99.525670
	3	99.556593
	4	99.587517
	5	99.618440
	6	99.649363
	7	99.680287
	8	99.711210
	9	99.742134
	10	99.773057
	11	99.803980
	12	99.834904

	13	99.865827
	14	99.896750
	15	99.896750
	16	99.958597
	17	99.989520
	18	100.020444
	19	100.051367
	20	100.082291
	21	100.113214
	22	100.144137
	23	100.175061
	24	100.205984
	25	100.236907
	26	100.267831
	27	100.298754
	28	100.329678
	29	100.360601
	30	100.391524
July	1	100.422448
	2	100.453371
	3	100.484294
	4	100.515218
	5	100.546141
	6	100.577064
	7	100.607988

	8	100.638911
	9	100.669835
	10	100.700758
	11	100.731681
	12	100.762605
	13	100.793528
	14	100.824451
	15	100.855375
	16	100.886298
	17	100.917221
	18	100.948145
	19	100.979068
	20	101.009992
	21	101.040915
	22	101.071838
	23	101.102762
	24	101.133685
	25	101.164608
	26	101.195532
	27	101.226455
	28	101.226455
	29	101.288302
	30	101.288302
	31	101.350149

Based on the results, the estimated closing price of Google stocks is expected to follow an upward trend over the next 6 months of 2023.

Conclusion

Based on the gathered data, the researchers concluded that the observations do not possess seasonality, as determined by the seasonal decomposition method. The p-value of 0.825986, which is greater than 0.05, indicates that the time series data is non-stationary. The data was made stationary using automatic model identification.

The best model was found to be ARIMA(1,1,1)(0,0,0)[0]. However, the evaluation metrics suggest that the model may not have performed well in forecasting the closing price of Google stocks, as indicated by high error values and a low R-squared value of -1.4829164864241617. This suggests that the model does not fit the data well. It is important to note that stock market prices are known to be volatile and noisy, making accurate forecasting challenging.

Despite this, the forecasted values using the ARIMA model indicate that the predicted closing price of Google stocks will tend to increase over the next six months of 2023.

Recommendation

Based on the observations and findings, the researchers suggest the following recommendations for further studies:

1. Explore other time series analysis techniques, such as exponential smoothing, neural networks, or deep learning, to compare their performance and accuracy with the ARIMA model.

2. Incorporate external factors, such as macroeconomic indicators, market sentiment, or news events, that may affect the stock price movements and volatility.
3. Extend the forecasting horizon to longer periods, such as one year or more, and evaluate the stability and reliability of the ARIMA model over time.
4. Apply the ARIMA model to other stock markets or sectors, such as emerging markets, technology, or healthcare, and analyze the similarities and differences in the patterns and forecasts.

REFERENCES

Alphabet Inc. (2023). *Stock forecast & price targets - Stock analysis*. Stock Analysis. https://stockanalysis.com/stocks/goog/forecast/

Ariyo, A. A., Adewumi, A. O., & Ayo, C. K. (2014). *Stock Price Prediction Using the ARIMA Model*. https://doi.org/10.1109/uksim.2014.67

Autoregressive Integrated Moving Average (ARIMA) Prediction Model. (2023). Investopedia. https://www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arima.asp

Brownlee, J. (2017). *Introduction to time series forecasting with Python*. https://machinelearningmastery.com/introduction-to-time-series-forecasting-with-python/

Dereje Workneh. (2020, June 6). *Stock price prediction using ARIMA Model - Dereje Workneh - Medium*. Medium; Medium. https://medium.com/@derejeabera/stock-price-prediction-using-arima-model-251ddb4ee52a

Google | History & Facts | Britannica. (2023). In *Encyclopædia Britannica*. https://www.britannica.com/topic/Google-Inc

Hardikkumar Dhaduk. (2021, July 18). *Stock market forecasting using Time Series analysis With ARIMA model*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/07/stock-market-forecasting-using-time-series-analysis-with-arima-model/

Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and practice*. https://otexts.com/fpp2/

Jillani Soft Tech. (2015). *Google 10 years stock price dataset*. Kaggle.com. https://www.kaggle.com/datasets/jillanisofttech/google-10-years-stockprice-dataset

Mehar Vijh, Deeksha Chandola, Vinay Anand Tikkiwal, & Kumar, A. (2020). Stock closing price prediction using machine learning techniques. *167*, 599–606. https://doi.org/10.1016/j.procs.2020.03.326

Case Study Codes

July 15, 2023

In [1]:
```
# importing necessary libraries

import warnings
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import statsmodels.api as sm
from pmdarima import auto_arima
from sklearn import metrics
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import adfuller
warnings.filterwarnings("ignore")
```

In [2]:
```
# reading GoogleStock Price Updated.csv dataset

df = pd.read_csv('GoogleStock Price Updated.csv')
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True)
```

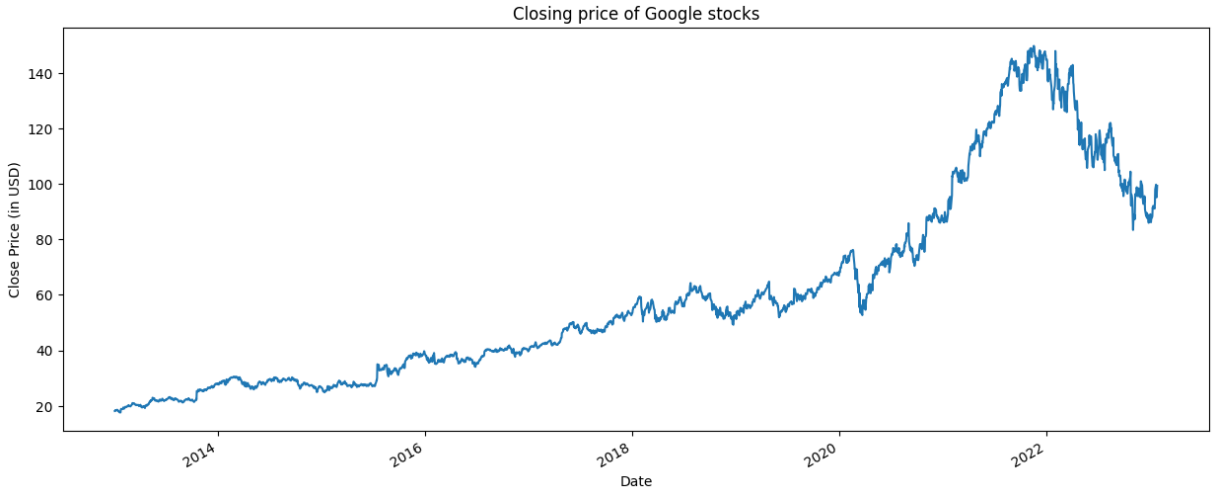
In [3]:
```
# showing the first ten rows of Google stock data

df.head(10)
```

Out[3]:

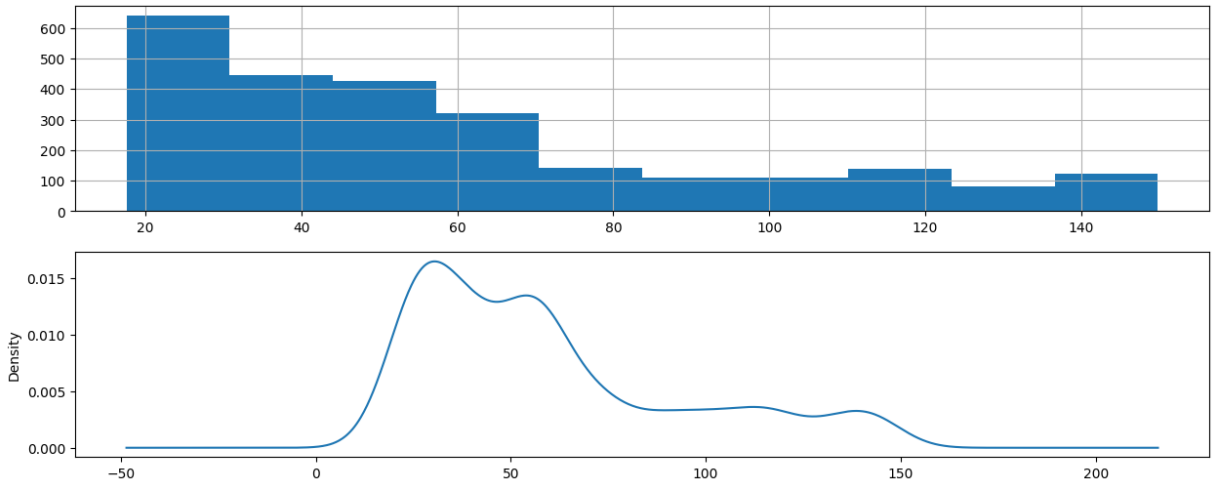
Date	Open	High	Low	Close	Adj Close	Volume
2013-01-02	18.003504	18.193193	17.931683	18.099348	18.099348	101550348
2013-01-03	18.141392	18.316566	18.036036	18.109859	18.109859	92635272
2013-01-04	18.251753	18.555305	18.210211	18.467718	18.467718	110429460
2013-01-07	18.404655	18.503002	18.282784	18.387136	18.387136	66161772
2013-01-08	18.406906	18.425926	18.128880	18.350851	18.350851	66976956
2013-01-09	18.325075	18.477226	18.233232	18.471472	18.471472	80907012
2013-01-10	18.589338	18.643644	18.355856	18.555555	18.555555	73354572
2013-01-11	18.568569	18.579329	18.425926	18.518269	18.518269	51356592
2013-01-14	18.443443	18.573574	18.076826	18.099348	18.099348	114441444
2013-01-15	18.001251	18.393393	17.820320	18.141392	18.141392	156950892

In [4]:
```
# Explanatory data analysis
```

```
df["Close"].plot(figsize=(15, 6))
plt.xlabel("Date")
plt.ylabel("Close Price (in USD)")
plt.title("Closing price of Google stocks")
plt.show()
```



In [5]:
```
# Kernel density estimation

plt.figure(1, figsize=(15,6))
plt.subplot(211)
df["Close"].hist()
plt.subplot(212)
df["Close"].plot(kind='kde')
plt.show()
```

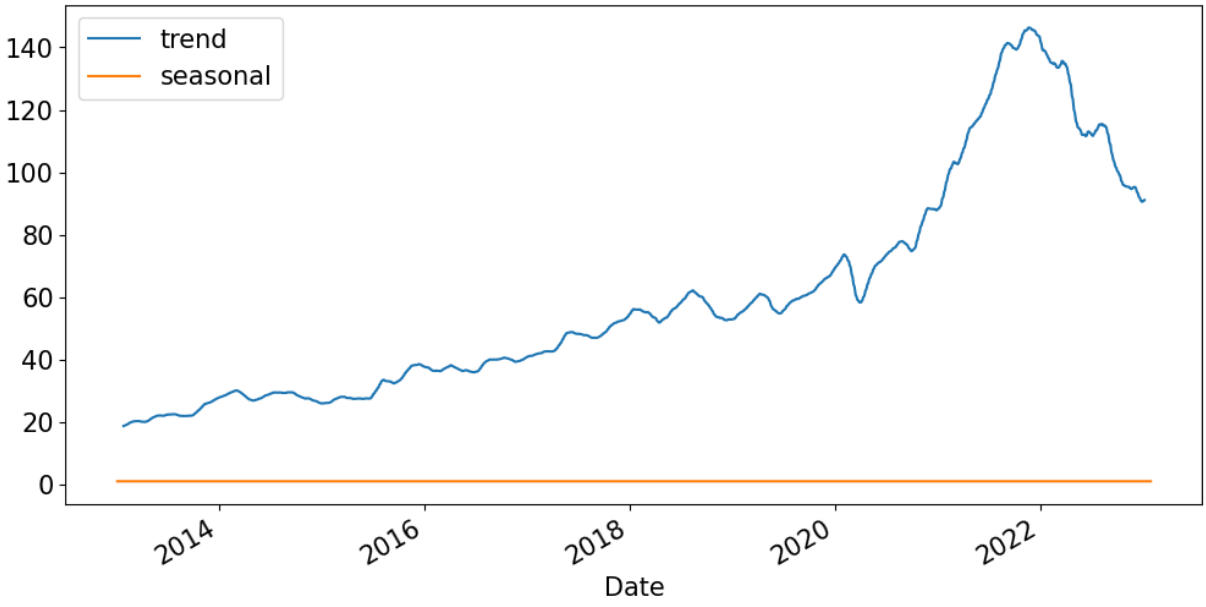


In [6]:
```
decomposition = sm.tsa.seasonal_decompose(df['Close'], model='multiplicative',
                                          period=30)
```

In [7]:
```
# Extract the trend and seasonal components

trend = decomposition.trend
seasonal = decomposition.seasonal
df_decomposed = pd.concat([trend, seasonal], axis=1)
print(df_decomposed.head(30))
```

Date	trend	seasonal
2013-01-02	NaN	0.996802
2013-01-03	NaN	0.998491
2013-01-04	NaN	1.001984
2013-01-07	NaN	1.002228
2013-01-08	NaN	1.001632
2013-01-09	NaN	0.999782
2013-01-10	NaN	1.000786
2013-01-11	NaN	0.998532
2013-01-14	NaN	1.000995
2013-01-15	NaN	1.003345
2013-01-16	NaN	1.001374
2013-01-17	NaN	1.002212
2013-01-18	NaN	0.999136
2013-01-22	NaN	0.999561
2013-01-23	NaN	0.998783
2013-01-24	18.707061	0.999986
2013-01-25	18.762863	1.000374
2013-01-28	18.820462	0.999855
2013-01-29	18.873261	1.001670
2013-01-30	18.923286	1.000619
2013-01-31	18.974929	1.001335
2013-02-01	19.021175	1.001385
2013-02-04	19.062646	0.998438
2013-02-05	19.115478	0.999499
2013-02-06	19.179208	0.999850
2013-02-07	19.248974	0.999034
2013-02-08	19.332883	0.999438
2013-02-11	19.434764	0.997538
2013-02-12	19.544290	0.996598
2013-02-13	19.635886	0.998738

In [8]:
```
# Plot the values of the df_decomposed DataFrame

dfd = df_decomposed.plot(figsize=(12, 6), fontsize=15);

# Specify axis labels

dfd.set_xlabel('Date', fontsize=15);
plt.legend(fontsize=15);
```



AUTO ARIMA

In [9]:
```
df = pd.read_csv('GoogleStock Price Updated.csv', parse_dates = True)
```

In [10]:
```
# Time series Evaluation

def timeseries_evaluation_metrics_func(y_true, y_pred):
    def mean_absolute_percentage_error(y_true, y_pred):
        y_true, y_pred = np.array(y_true), np.array(y_pred)
        return np.mean(np.abs((y_true - y_pred) / y_true)) * 100
    print('Evaluation metric results:-')
    print(f'MSE is : {metrics.mean_squared_error(y_true,y_pred)}')
    print(f'MAE is : {metrics.mean_absolute_error(y_true,y_pred)}')
    print(f'RMSE is :{np.sqrt(metrics.mean_squared_error(y_true, y_pred))}')
    print(f'MAPE is : {mean_absolute_percentage_error(y_true,y_pred)}')
    print(f'R2 is : {metrics.r2_score(y_true,y_pred)}',end='\n\n')
```

In [11]:
```
# ADF with Akaike information criterion

def Augmented_Dickey_Fuller_Test_func(series , column_name):
    print (f'Results of Dickey-Fuller Test for column: {column_name}')
    dftest = adfuller(series, autolag='AIC')
    dfoutput = pd.Series(dftest[0:4],
                         index=['Test Statistic', 'p-value',
                                'No Lags Used','Number of Observations Used'])
    for key,value in dftest[4].items():
        dfoutput['Critical Value (%s)'%key] = value
    print (dfoutput)
    if dftest[1] <= 0.05:
        print("Conclusion:====>")
        print("Reject the null hypothesis")
        print("Data is stationary")
    else:
        print("Conclusion:====>")
```

```
        print("Fail to reject the null hypothesis")
        print("Data is non-stationary")
```

In [12]:
```
# ADF Test

Augmented_Dickey_Fuller_Test_func(df['Close'],'Close')
```

```
Results of Dickey-Fuller Test for column: Close
Test Statistic                  -0.776298
p-value                          0.825986
No Lags Used                    23.000000
Number of Observations Used    2512.000000
Critical Value (1%)              -3.432956
Critical Value (5%)              -2.862691
Critical Value (10%)             -2.567383
dtype: float64
Conclusion:====>
Fail to reject the null hypothesis
Data is non-stationary
```

In [13]:
```
# Splitting data - assignment of train and test set

X = df[['Close' ]]
train, test = X[0:-30], X[-30:]
```

In [14]:
```
# AUTO ARIMA

stepwise_model = auto_arima(train,start_p=1, start_q=1, max_p=7, max_q=7,
                            seasonal=False,d=None, trace=True,
                            error_action='ignore', suppress_warnings=True,
                            stepwise=True)
stepwise_model.summary()
```

```
Performing stepwise search to minimize aic
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=8324.454, Time=1.45 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=8335.353, Time=0.06 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=8328.448, Time=0.26 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=8328.368, Time=0.28 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=8334.826, Time=0.08 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=8325.608, Time=2.07 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=8325.688, Time=1.39 sec
 ARIMA(0,1,2)(0,0,0)[0] intercept   : AIC=8330.289, Time=0.42 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=8330.420, Time=0.29 sec
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=8325.851, Time=3.68 sec
 ARIMA(1,1,1)(0,0,0)[0]             : AIC=8324.743, Time=0.64 sec

Best model:  ARIMA(1,1,1)(0,0,0)[0] intercept
Total fit time: 10.693 seconds
```

Out[14]:

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	2506
Model:	SARIMAX(1, 1, 1)	**Log Likelihood**	-4158.227
Date:	Sat, 15 Jul 2023	**AIC**	8324.454
Time:	11:41:54	**BIC**	8347.758
Sample:	0	**HQIC**	8332.914
	- 2506		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	0.0079	0.006	1.268	0.205	-0.004	0.020
ar.L1	0.7440	0.059	12.595	0.000	0.628	0.860
ma.L1	-0.7955	0.054	-14.652	0.000	-0.902	-0.689
sigma2	1.6194	0.019	84.031	0.000	1.582	1.657

Ljung-Box (L1) (Q):	0.36	**Jarque-Bera (JB):**	9025.53
Prob(Q):	0.55	**Prob(JB):**	0.00
Heteroskedasticity (H):	20.34	**Skew:**	-0.12
Prob(H) (two-sided):	0.00	**Kurtosis:**	12.30

Warnings:

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

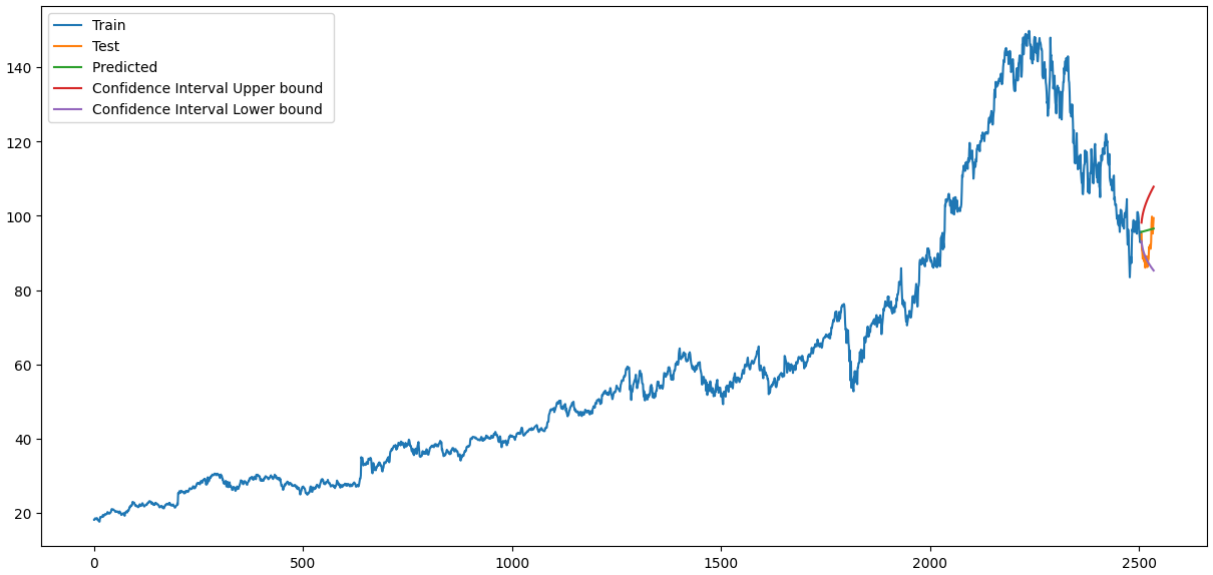
In [15]:

```
# Forecast both results and the confidence for the next 30 days and store it
# in a DataFrame

forecast,conf_int = stepwise_model.predict(n_periods=30,return_conf_int=True)
forecast = pd.DataFrame(forecast,columns=['close_pred'])
df_conf = pd.DataFrame(conf_int,columns= ['Lower_bound','Upper_bound'])
df_conf["new_index"] = range(2506, 2536)
df_conf = df_conf.set_index("new_index")
timeseries_evaluation_metrics_func(test, forecast)
```

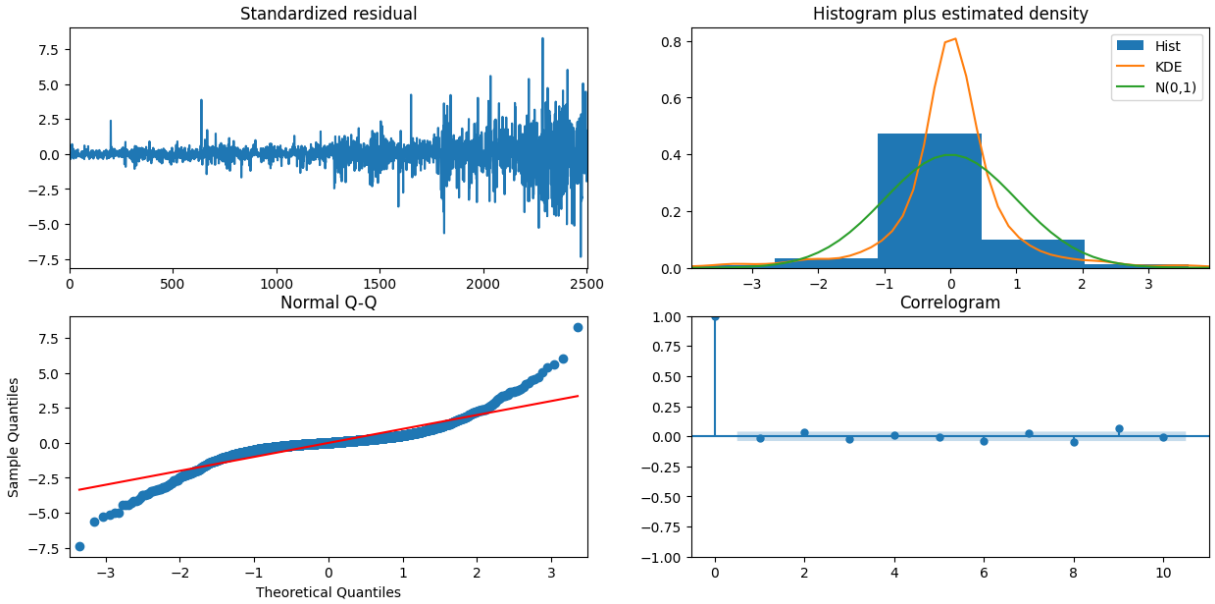
```
Evaluation metric results:-
MSE is : 38.56767982424719
MAE is : 5.598494093992771
RMSE is :6.2102882239270665
MAPE is : 6.26573810822338
R2 is : -1.4829164864241617
```

In [16]:
```
# Rearrange the indexes for the plots to align

forecast["new_index"] = range(2506, 2536)
forecast = forecast.set_index("new_index")
```

In [17]:
```
# Plot the results with confidence bounds

get_ipython().run_line_magic('matplotlib', 'inline')
plt.rcParams["figure.figsize"] = [15,7]
plt.plot( train, label='Train ')
plt.plot(test, label='Test ')
plt.plot(forecast, label='Predicted ')
plt.plot(df_conf['Upper_bound'], label='Confidence Interval Upper bound ')
plt.plot(df_conf['Lower_bound'], label='Confidence Interval Lower bound ')
plt.legend(loc='best')
plt.show()
```



In [18]:
```
# diagnostic plot

stepwise_model.plot_diagnostics();
```



In [19]:
```
# forecast using the best model

pred,conf_int = stepwise_model.predict(n_periods=185,return_conf_int=True)
pred = pd.DataFrame(pred,columns=['Close'])
df_conf = pd.DataFrame(conf_int,columns= ['Lower_bound','Upper_bound'])
df_conf["new_index"] = range(2537, 2722)
df_conf = df_conf.set_index("new_index")
timeseries_evaluation_metrics_func(test, forecast)
```

```
Evaluation metric results:-
MSE is : 38.56767982424719
MAE is : 5.598494093992771
RMSE is :6.2102882239270665
MAPE is : 6.26573810822338
R2 is : -1.4829164864241617
```

In [20]:
```
# Predicting the close price from January 28, 2023 to July 31, 2023

pred["new_index"] = range(2537, 2722)
pd.set_option('display.max_rows',None)
pred = pred.set_index("new_index")
pred
```

Out[20]:

new_index	Close
2537	95.660749
2538	95.691544
2539	95.722373
2540	95.753225
2541	95.784096
2542	95.814980
2543	95.845874
2544	95.876776
2545	95.907683
2546	95.938595
2547	95.969509
2548	96.000426
2549	96.031344
2550	96.062264
2551	96.093184
2552	96.124106
2553	96.155028
2554	96.185950
2555	96.216872
2556	96.247795
2557	96.278718
2558	96.309641
2559	96.340564
2560	96.371487
2561	96.402411
2562	96.433334
2563	96.464257
2564	96.495180
2565	96.526104

new_index	Close
2566	96.557027
2567	96.587950
2568	96.618874
2569	96.649797
2570	96.680720
2571	96.711644
2572	96.742567
2573	96.773491
2574	96.804414
2575	96.835337
2576	96.866261
2577	96.897184
2578	96.928107
2579	96.959031
2580	96.989954
2581	97.020877
2582	97.051801
2583	97.082724
2584	97.113648
2585	97.144571
2586	97.175494
2587	97.206418
2588	97.237341
2589	97.268264
2590	97.299188
2591	97.330111
2592	97.361034
2593	97.391958
2594	97.422881

new_index	Close
2595	97.453805
2596	97.484728
2597	97.515651
2598	97.546575
2599	97.577498
2600	97.608421
2601	97.639345
2602	97.670268
2603	97.701191
2604	97.732115
2605	97.763038
2606	97.793962
2607	97.824885
2608	97.855808
2609	97.886732
2610	97.917655
2611	97.948578
2612	97.979502
2613	98.010425
2614	98.041349
2615	98.072272
2616	98.103195
2617	98.134119
2618	98.165042
2619	98.195965
2620	98.226889
2621	98.257812
2622	98.288735
2623	98.319659

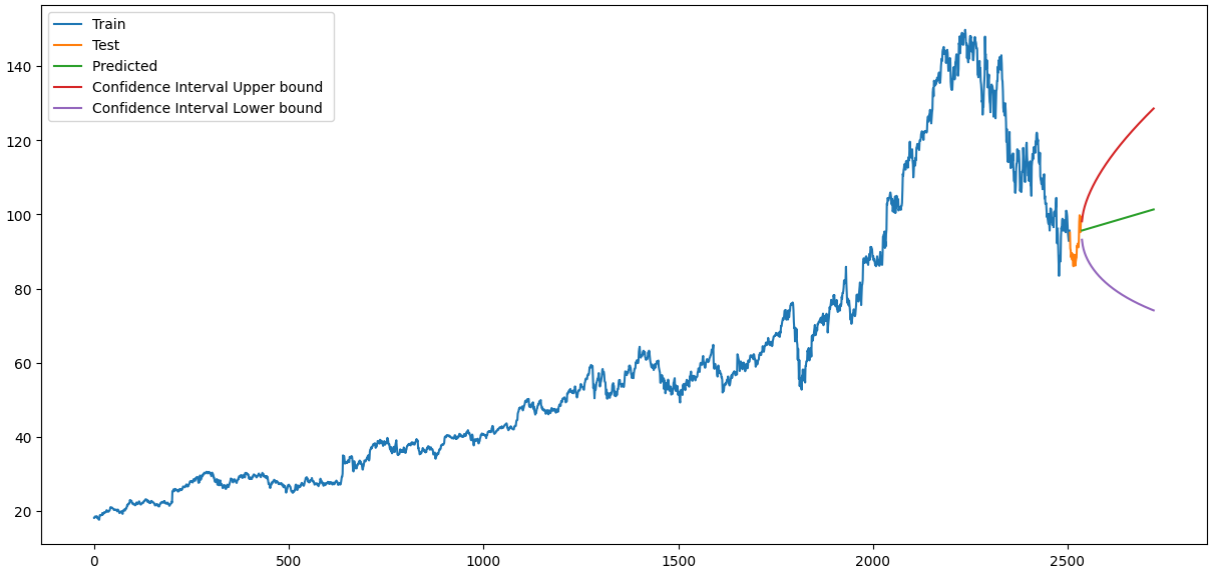
new_index	Close
2624	98.350582
2625	98.381506
2626	98.412429
2627	98.443352
2628	98.474276
2629	98.505199
2630	98.536122
2631	98.567046
2632	98.597969
2633	98.628892
2634	98.659816
2635	98.690739
2636	98.721663
2637	98.752586
2638	98.783509
2639	98.814433
2640	98.845356
2641	98.876279
2642	98.907203
2643	98.938126
2644	98.969049
2645	98.999973
2646	99.030896
2647	99.061820
2648	99.092743
2649	99.123666
2650	99.154590
2651	99.185513
2652	99.216436

new_index	Close
2653	99.247360
2654	99.278283
2655	99.309206
2656	99.340130
2657	99.371053
2658	99.401977
2659	99.432900
2660	99.463823
2661	99.494747
2662	99.525670
2663	99.556593
2664	99.587517
2665	99.618440
2666	99.649363
2667	99.680287
2668	99.711210
2669	99.742134
2670	99.773057
2671	99.803980
2672	99.834904
2673	99.865827
2674	99.896750
2675	99.927674
2676	99.958597
2677	99.989520
2678	100.020444
2679	100.051367
2680	100.082291
2681	100.113214

new_index	Close
2682	100.144137
2683	100.175061
2684	100.205984
2685	100.236907
2686	100.267831
2687	100.298754
2688	100.329678
2689	100.360601
2690	100.391524
2691	100.422448
2692	100.453371
2693	100.484294
2694	100.515218
2695	100.546141
2696	100.577064
2697	100.607988
2698	100.638911
2699	100.669835
2700	100.700758
2701	100.731681
2702	100.762605
2703	100.793528
2704	100.824451
2705	100.855375
2706	100.886298
2707	100.917221
2708	100.948145
2709	100.979068
2710	101.009992

	Close
new_index	
2711	101.040915
2712	101.071838
2713	101.102762
2714	101.133685
2715	101.164608
2716	101.195532
2717	101.226455
2718	101.257378
2719	101.288302
2720	101.319225
2721	101.350149

In [21]:
```
# Plot the predicted results with confidence bounds

get_ipython().run_line_magic('matplotlib', 'inline')
plt.rcParams["figure.figsize"] = [15,7]
plt.plot( train, label='Train ')
plt.plot(test, label='Test ')
plt.plot(pred, label='Predicted ')
plt.plot(df_conf['Upper_bound'], label='Confidence Interval Upper bound ')
plt.plot(df_conf['Lower_bound'], label='Confidence Interval Lower bound ')
plt.legend(loc='best')
plt.show()
```



End