

**School of Computer Science and Engineering**

**Predicting Stock Movement Using Market Correlation Networks**

**CSE3021 – Social and Information Networks a Project Report**

**Submitted to:**

**Dr Punitha.K**

**Submitted by:**

Kashish Doaj (18BCE1343)

Aditya Mohan Gupta (18BCE1017)

Mahavir Lunkad (19BCE1774)

In partial fulfilment for the award of the degree of

**Bachelor of Technology**

in

**Computer Science and Engineering**

*June 2021*

**TABLE OF CONTENTS**

**Page**

1. Abstract……………………………………………………………………………..…...3

2. Introduction…………………………………………………………………………….. 4

3. Feasibility study………………………………………………………………………….5

4. About the dataset…………………………………………………………………………7

5.Design and flow of models………………………………………………………………..8

6. Module list

6.1. Exploratory Data Analysis (EDA)……………...…………………………………..9

6.2. Data Pre-Processing……………………………..…………………………………..12

6.3 Building Various Models…………………………………………………………….14

7. Algorithms……………………………………………………………………………..…..15

8.Risk Analysis…………………….………………………………………………..……….19

9.Implementation……….………………………………………………………………….....20

9.1.Analysis………………………………………………………………………………20

9.2.Visualization…………………………………………………………………………22

9.3.Comparing the results.…………….………………………………………………...31

10. Results………………………………………………………………………………… 33

11.Conclusion…………………………………………………………………………… …34

12.References…………………………………………………………………………… ….35

13.List of figures……………………………………………………………………………..36

14.Appendix code…………………………………………………………………………….38

**ABSTRACT**

The aim of Predicting Stock Movement is to provide the customers with measured predicted data and analysis of the stock market. This project will use a correlation network to deal with all functionalities a stock market will encounter. This project is best suited for stock movements which have precise previous data which can be analysed to predict the future movements. This project helps to track and analyse the current and previous stock movements and predict the future ones. It Stores the data of stock movements happening. While working on this project it also helps us to know our stocks, learn machine learning properly and understand the models of Stock Prediction.

**INTRODUCTION**

The goal for this project is to discern whether network properties of financial markets can be used to predict market dynamics. Building on previous work involving networks derived from market price correlations, we augment basic price correlation networks with additional information (revenue, sentiment, and news-ow). Our intuition is that these alternative networks will capture relationships beyond price correlations (e.g., business model exposures) that could eventually enhance downstream predictive models. The final insight we aim to provide is a prediction of future market behaviour based on features that incorporate both standard trading information (price, volume, etc.) and market network characteristics (centrality, clustering coefficient, etc.).

The project methodology comprises three components: structural, analytical, and predictive. In the structural component, we filter the data to find and visualize the underlying structural motifs of the network. In the analytical component, additional metrics are computed for graphs built from the full dataset, and we do statistical testing to see whether our graph features have predicting power for stock prices. These two components use correlations of both prices and the newly introduced news/sentiment variables when building networks. They also featurize properties of our market correlation networks for sub-periods of years or quarters to see how these networks change over time. Finally, the predictive component incorporates features/metrics generated by the structural and analytical components into a recurrent neural network (RNN) to predict binary market movements (up/down) over a future period of interest.

**FEASIBILITY STUDY**

Feasibility Study Depending on the results of the initial investigation a more detailed feasibility study is prepared so as to evaluate if the project is viable or not. “FEASIBILITY STUDY” is a test of system proposal according to its workability, impact of the organization, ability to meet needs and effective use of the resources. It focuses on these major questions:

1. What are the user’s needs and how does the proposed system meet them?

2. What resources are available with the user?

3. Is it worth solving the problem?

During feasibility analysis for this project, following primary areas of interest were considered. Investigation and generating ideas about the new solution gives the following results.

**Technical feasibility**

A study of resource availability that may affect the ability to achieve an acceptable system. This evaluation determines whether the technology needed for the proposed system is available or not.

This is concerned with specifying equipment and software that will successfully satisfy the user requirement.

During this project the

1. Software requirement

1.1 Any basic Operating System like Linux, Windows or MAC

1.2 Google Colab

2. Hardware requirement

2.1 8GB RAM for faster training of the neural network

The technical feasibility is the most important part of this project as having the required system only can enable the use of the proposed solution.

**Economic feasibility**

Economic justification is generally the “Bottom Line” consideration for most systems. Economic justification includes a broad range of concerns that includes cost benefit analysis. In this we weigh the cost and the benefits associated with the proposed solution and if it suits the basic purpose of the organization generating revenues. Economic feasibility for this project is not required

**Operational Feasibility**

It is mainly related to human organizations and political aspects. The system is operationally feasible as it is very easy to operate. It only needs basic information about any OS platform.

**Schedule feasibility**

Time evaluation is the most important consideration in the development of a project. The time schedule required for the development of this project is very important since more development time affects machine time, cost and causes delay in the development of other systems. The model can be developed in the considerable amount of time

#### ABOUT DATASET

The dataset consists of details about the stock market prices and are listed below:

#### The types of stock movements are:

#### Date

#### Open

#### Close

#### High

#### Low

#### Volume

#### File descriptions

#### • train.csv - the training set, contains details of stock price with their labels (has 1259rows \* 6 columns)

#### • test.csv - the test set for which the stock market price will be predicted (has 21rows \* 6 columns)

#### Datasets:

#### <https://www.kaggle.com/akram24/google-stock-price-train>

#### <https://www.kaggle.com/akram24/google-stock-price-test>

#### DESIGN AND FLOW OF MODULES:

#### Following are the modules in which we have divided are project:

#### Forming the strategy for making prediction model and setting the definite approach.

#### Analysing the data

#### Visualizing the data using several types of plots

#### Creating a train and test dataset from the available data and making a feature extraction model.

#### Implementing different ML models and algorithms and doing a comparative analysis to find out where the accuracy is best.

#### MODULE LIST

#### Exploratory Data Analysis (EDA):

#### Exploratory Data Analysis is one of the important steps in the data analysis process. Here, the focus is on making sense of the data in hand — things like formulating the correct questions to ask to your dataset, how to manipulate the data sources to get the required answers, and others.

#### First, we will import the required libraries

#### FIGURE 1: IMPORTING LIBRARIES

#### Load the csv dataset and check the first five rows as default from it.

#### FIGURE 2: LOADING DATASET

#### Get the overall information of the dataset used.

#### FIGURE 3: INFORMATION REGARDING DATASET USED

#### Plot the graph of one of the columns, here, OPEN to show the correctness

#### FIGURE 4: CORRECTNESS GRAPH

#### FIGURE 5: GRAPH REPRESENTATION

#### FIGURE 6: GRAPH REPRESENTATION

#### Data Pre-processing:

#### Remove any punctuation marks from the columns

#### FIGURE 7: REMOVAL OF PUNCTUATION MARKS

#### Plotted Open and Close Column

#### FIGURE 8: OPEN AND CLOSE REPRESENTATION

#### Data Cleaning and setting the training data frame

#### 

#### FIGURE 9: DATA CLEANING

#### 

#### FIGURE 10: FEATURE SCALING

#### Building Various Models:

We use various models to predict the stock movements in the market. The models that we have used are as follows:

LSTM:

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. This is a behaviour required in complex problem domains like machine translation, speech recognition, and more. LSTMs are a complex area of deep learning. It can be hard to get your hands around what LSTMs are, and how terms like bidirectional and sequence-to-sequence relate to the field.

ARIMA:

An ARIMA model is a class of statistical models for analysing and forecasting time series data. It explicitly caters to a suite of standard structures in time series data, and as such provides a simple yet powerful method for making skilful time series forecasts. It's a model used in statistics and econometrics to measure events that happen over a period of time. The model is used to understand past data or predict future data in a series.

FBPROPHET:

Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet is an additive regression model with a piecewise linear or logistic growth curve trend. It includes a yearly seasonal component modelled using Fourier series and a weekly seasonal component modelled using dummy variables.

#### ALGORITHMS

**LSTM Model:**

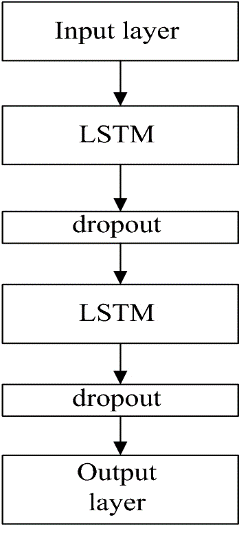
In order to build the LSTM, we need to import a couple of modules from [Keras](https://heartbeat.fritz.ai/introduction-to-deep-learning-with-keras-c7c3d14e1527" \t "_blank):

* Sequential for initializing the neural network.
* Dense for adding a densely connected neural network layer
* LSTM for adding the Long Short-Term Memory layer
* Dropout for adding dropout layers that prevent overfitting

We add the LSTM layer and later add a few Dropout layers to prevent overfitting. We add the LSTM layer with the following arguments:

* 50 units which is the dimensionality of the output space
* return\_sequences=True which determines whether to return the last output in the output sequence, or the full sequence
* input\_shape as the shape of our training set.

When defining the Dropout layers, we specify 0.2, meaning that 20% of the layers will be dropped. Thereafter, we add the Dense layer that specifies the output of 1 unit. After this, we compile our model using the popular adam optimizer and set the loss as the mean\_squarred\_error. This will compute the mean of the squared errors. Next, we fit the model to run on 100 epochs with a batch size of 32.

**Diagram 1: Flow algorithm of LSTM Diagram 2: Block diagram**

**Arima Model:**

The time series model is thought to consist of three systematic components including level, trend, seasonality, and one non-systematic component called noise.

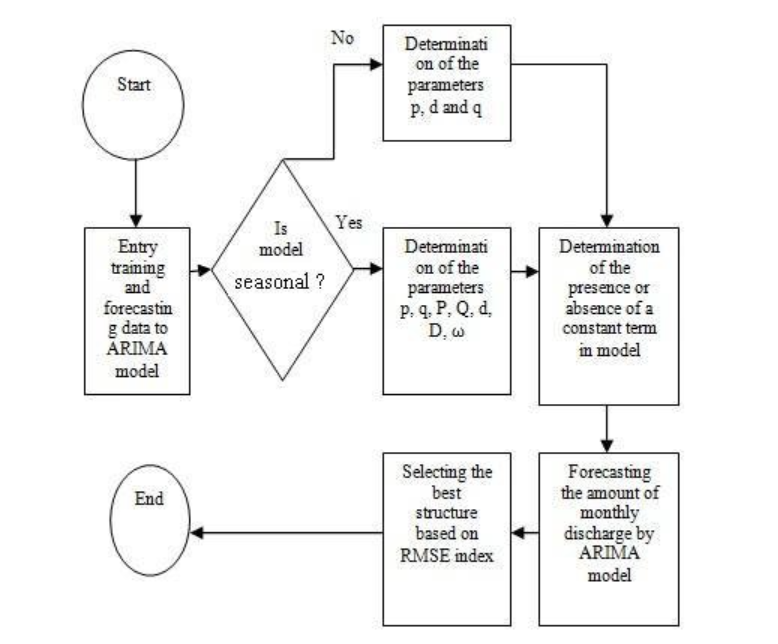
These components are defined as follows:

* **Level**: The average value in the series.
* **Trend**: The increasing or decreasing value in the series.
* **Seasonality**: The repeating short-term cycle in the series.
* **Noise**: The random variation in the series.

First, we need to check if a series is stationary or not because time series analysis only works with stationary data.

Let’s divide the data into a training (70 % ) and test (30%) set. For this tutorial we select the following ARIMA parameters: p=4, d=1 and q=0.

A rolling forecasting procedure is required given the dependence on observations in prior time steps for differencing and the AR model. To this end, we re-create the ARIMA model after each new observation is received.



**Diagram 3: Flow Diagram of ARIMA**

**FBPROPHET MODEL:**

For implementing in Python, it is needed to have some of the libraries installed: **Pandas, Matplotlib, FB Prophet** (we can install it through! pip install fbprophet).

Prophet works under the specific format that is the data frame should have only **‘ds**and **‘y’** as names **ds** for dates and **y** for observations*.*

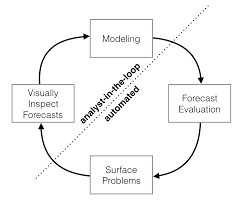
After renaming the columns now the next thing to do is to change the data to date time format using Pandas as Prophet algorithm uses DateTime format.

After the model is created it is fitted with the data(*data\_to\_use*here) like in other Deep Learning/Machine Learning Models i.e.

model.fit(data\_to\_use)

One of the interesting things about Prophet is that with make\_future\_dataframe function Periods for the future prediction can be provided.

When the models are loaded with future dates, the **‘y’** component values get predicted.



**Diagram 4: Flow Diagram of Prophet**

**Adam Optimizer**

Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data.

When introducing the algorithm, the authors list the attractive benefits of using Adam on non-convex optimization problems, as follows:

* Straightforward to implement.
* Computationally efficient.
* Little memory requirements.
* Invariant to diagonal rescale of the gradients.
* Well suited for problems that are large in terms of data and/or parameters.
* Appropriate for non-stationary objectives.
* Appropriate for problems with very noisy/or sparse gradients.
* Hyper-parameters have intuitive interpretation and typically require little tuning.

**Tikhonov Regularization**

Tikhonov regularization, named for Andrey Tikhonov, is a method of regularization of ill-posed problems. Also known as ridge regression, it is particularly useful to mitigate the problem of multicollinearity in linear regression, which commonly occurs in models with large numbers of parameters. In general, the method provides improved efficiency in parameter estimation problems in exchange for a tolerable amount of bias (see bias-variance trade-off).

**Risk Analysis**

**Identifying the Risk**

The potential risks involved while doing this project were the time limit and the CPU/GPU/TPU requirement to trains the vast number of neural networks and machine learning models. With a consumer grade computer, it would be not be possible to train all the models hence most of the training process has to be done on cloud platforms like google Colab or Aws sage maker. Even these online cloud platforms come their own personal risks like the RAM getting overloaded with data due to the vast amount of training data.

**Probability of Risk affecting the project**

Although google Colab has the necessary support to run the models even in free version but still due to the sheer size of the corpus there is some chance that colab may crash due to excessive load on the RAM.

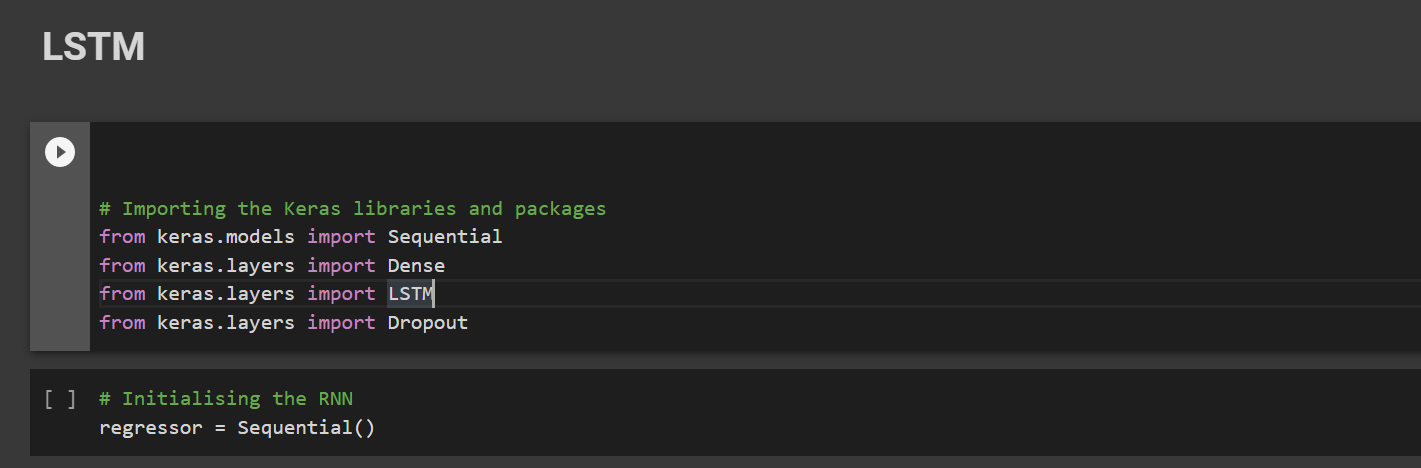
**Impact on the project**

The impact of the above-mentioned risk is quite minimal as if the RAM does not support the entire training data at a time, then measures can be taken to feed chunks of training data and then go for an ensemble model after training each individual model. This will be better overall and could lead to faster training with lesser RAM consumption

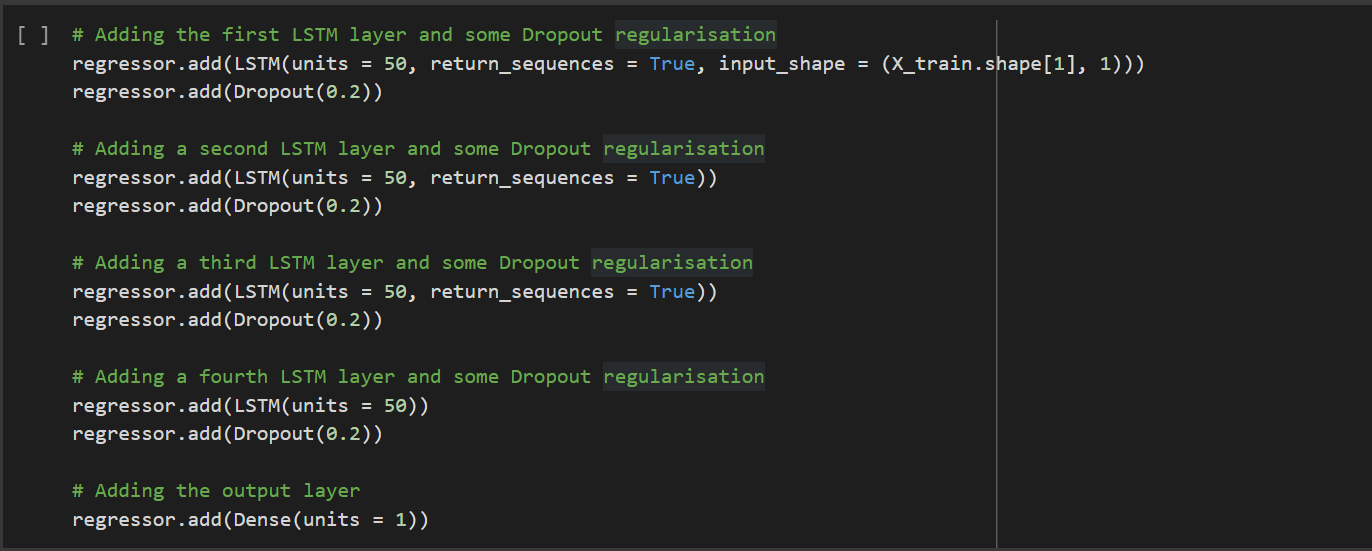
**IMPLEMENTATION**

1. **Analysis**

* Importing the libraries and initializing the RNN

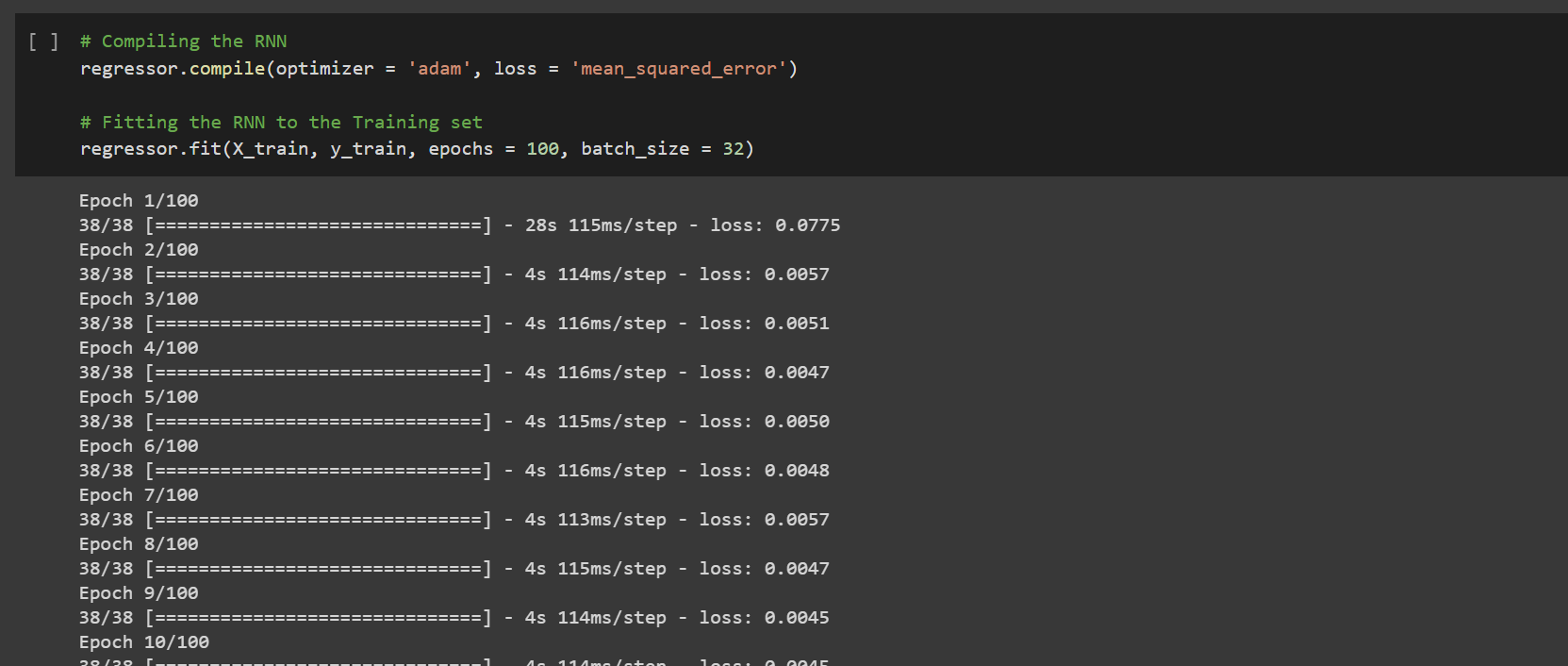


**FIGURE 11: IMPORTING REQUIRED LIBRARIES**



**FIGURE 12: LSTM MODEL IMPLEMENTATION**

* Compiling the RNN and using adam optimizer



**FIGURE 13: COMPILATION OF RNN**

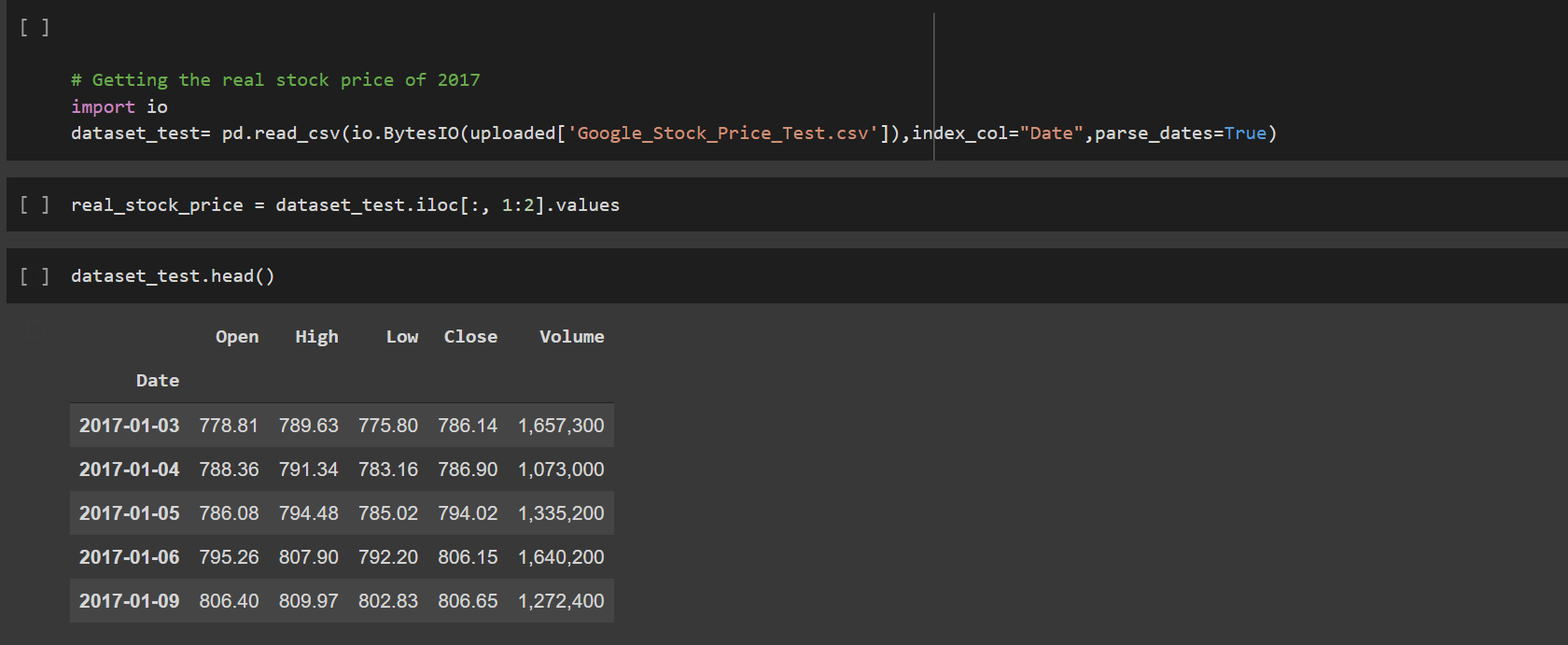
* Importing the Test data set from Kaggle



**FIGURE 14: DATASET IMPORT**

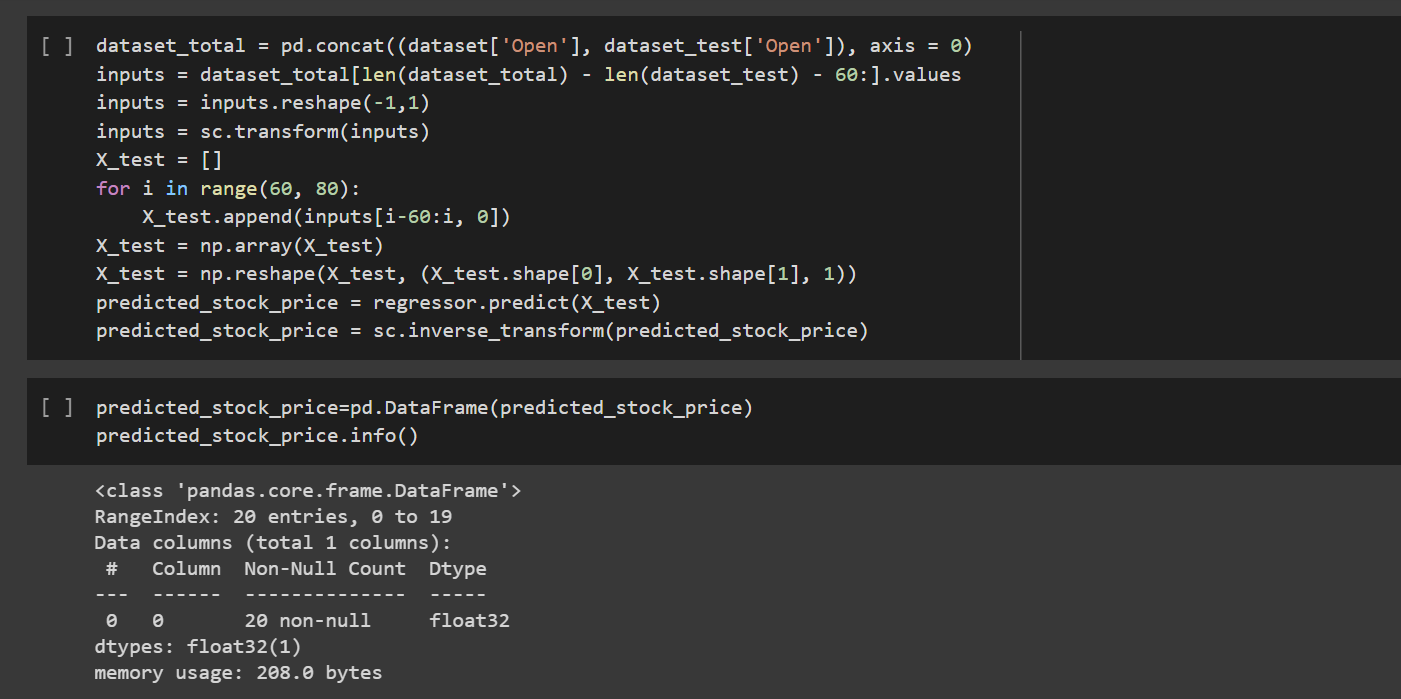
* Making predictions and visualizing the data

Importing Test dataset and extracting information about it.



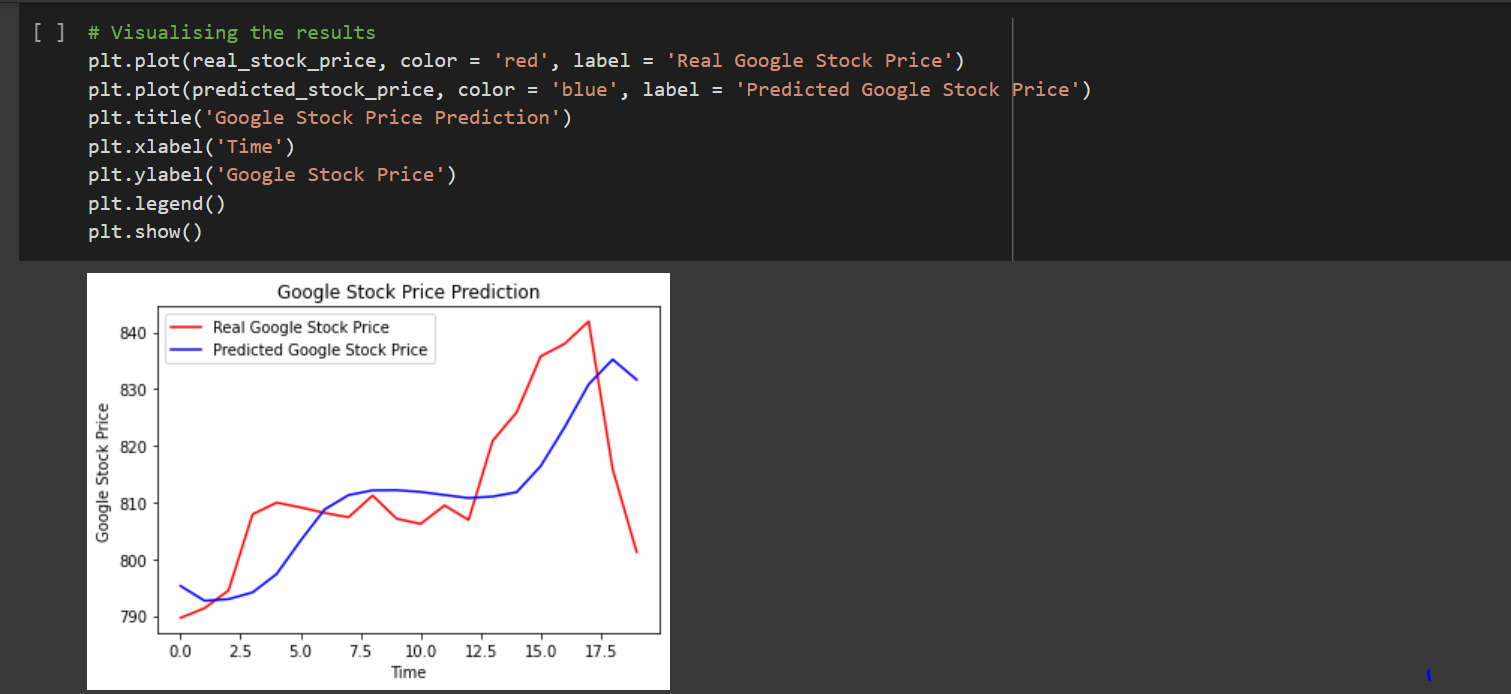
**FIGURE 15: PREDICTION IMAGE**





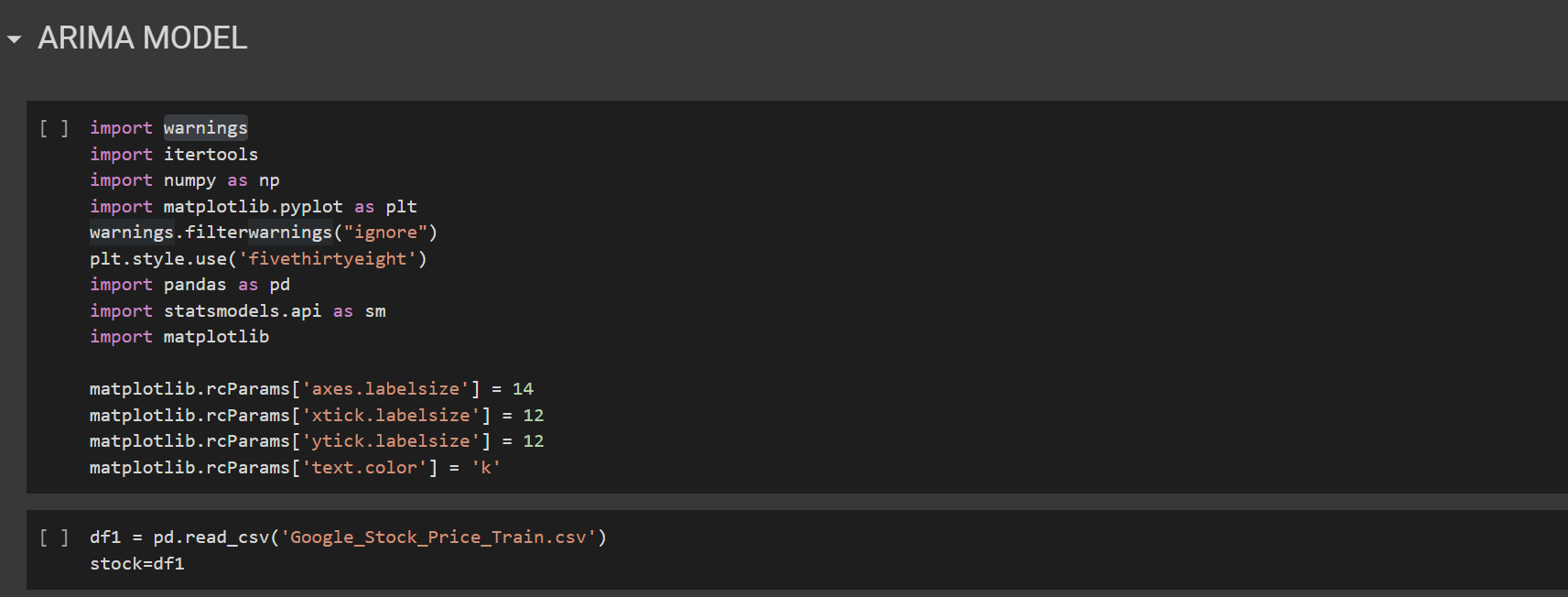
**FIGURE 16: PREDICTION IMAGE**

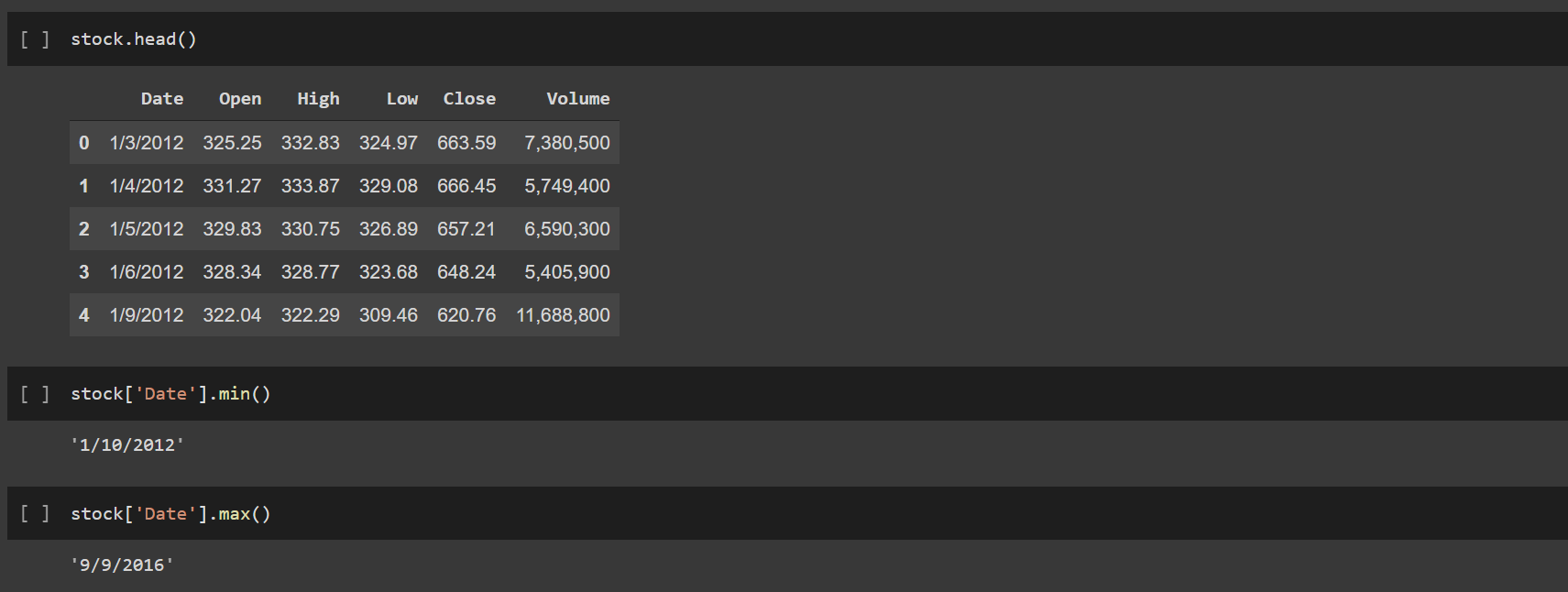
1. **Visualization**



**FIGURE 17: PREDICTION**

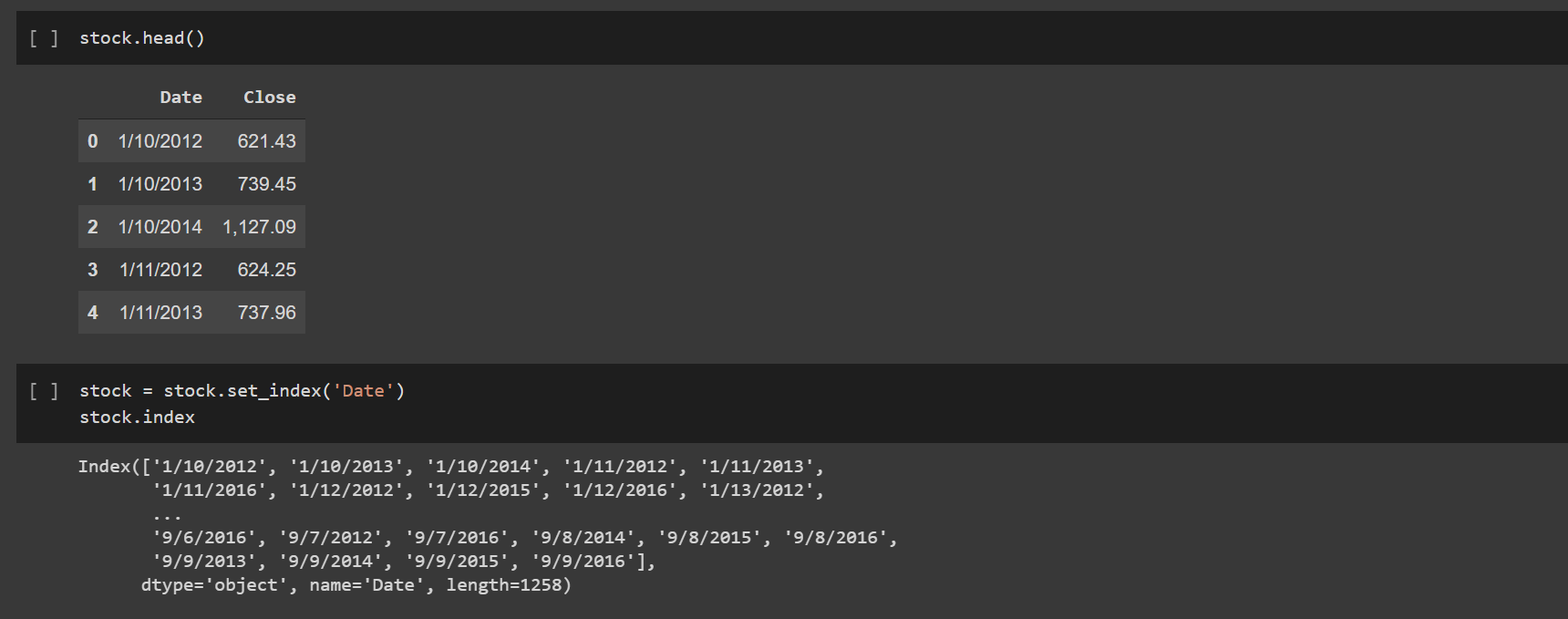
**ARIMA MODEL:**







**FIGURE 18: ARIMA MODEL IMPLEMENTATION**



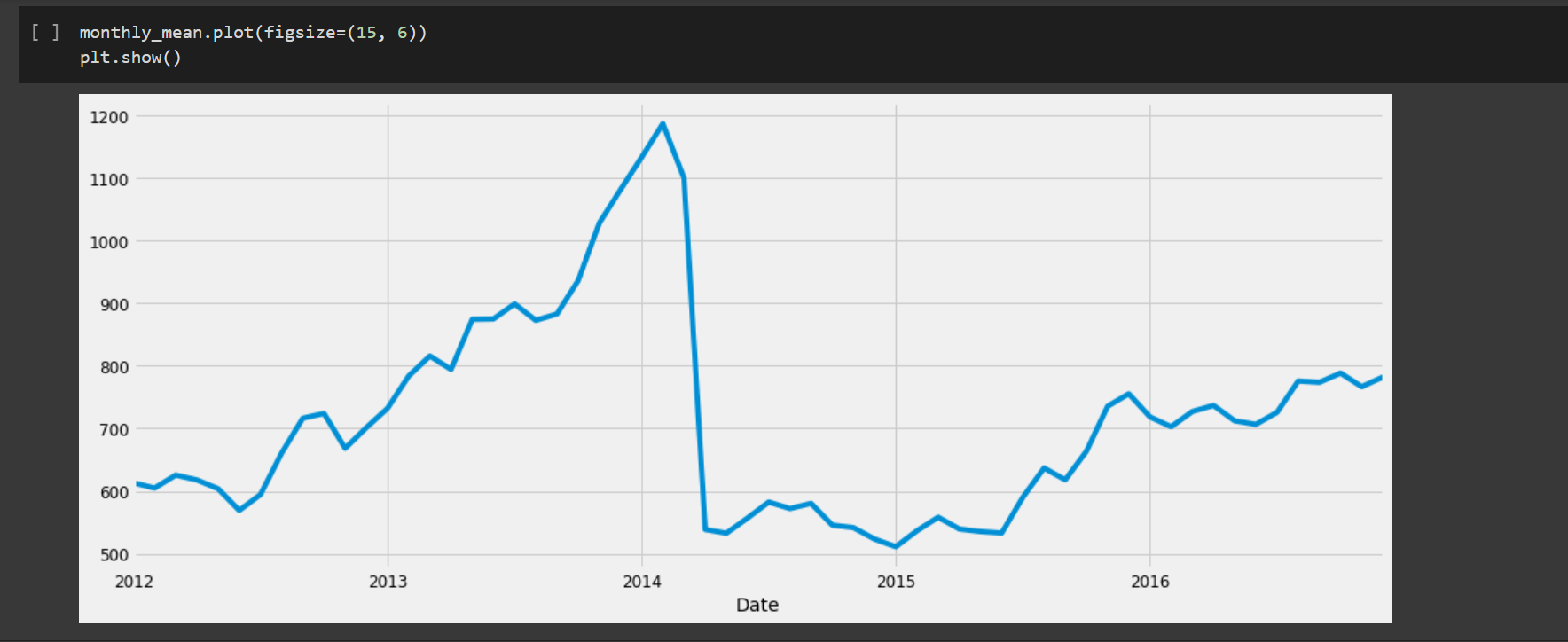
**FIGURE 19: ARIMA MODEL WORKING**

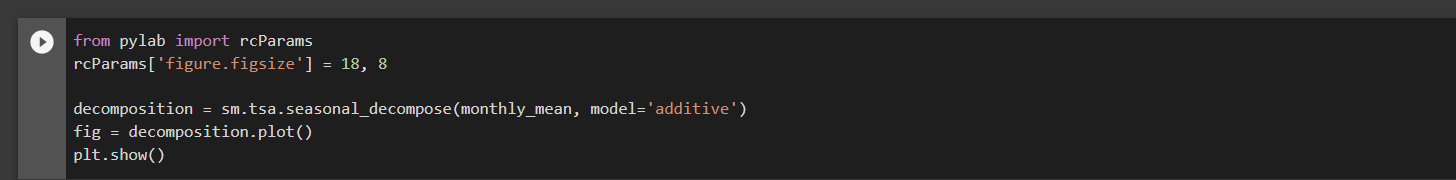


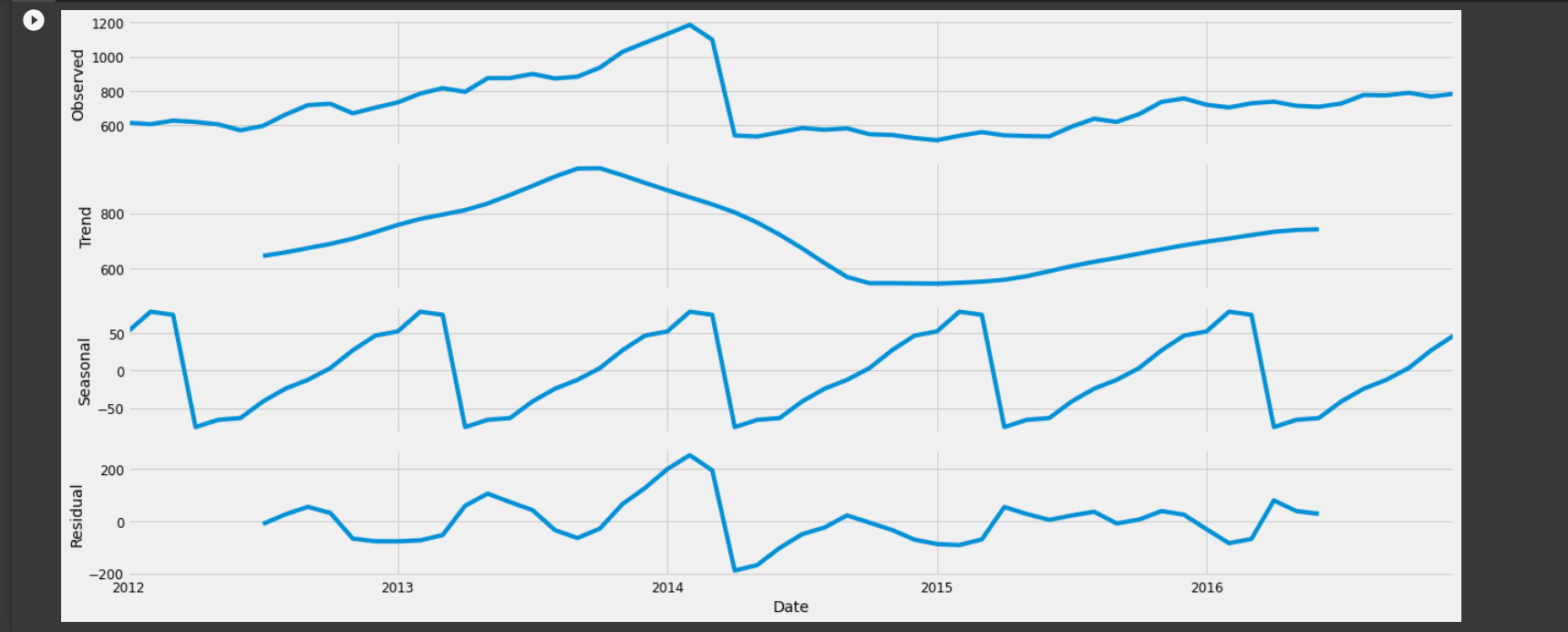
**FIGURE 20: ARIMA MODEL OUTPUT**

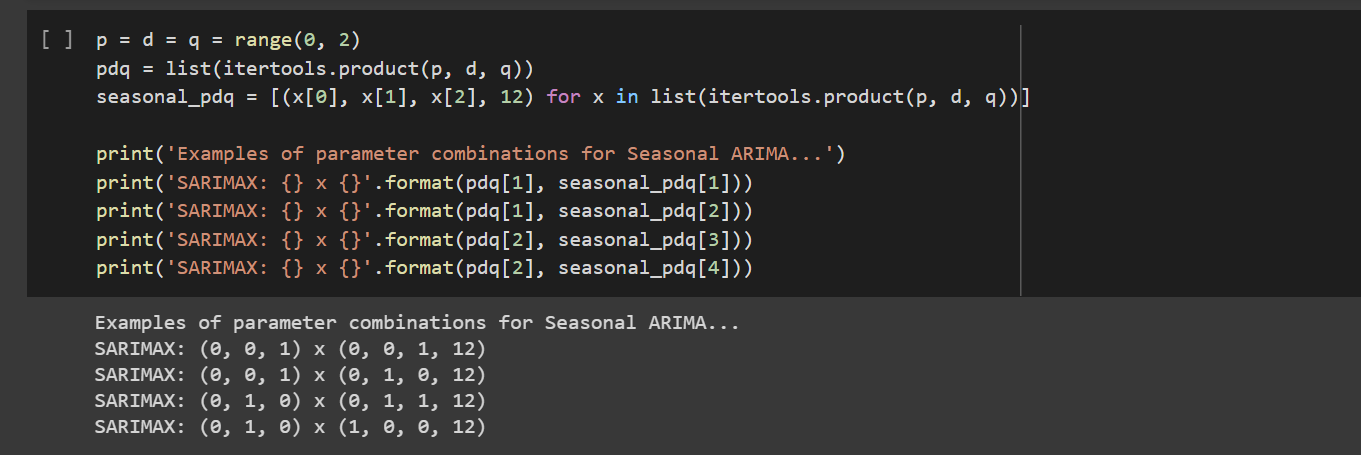


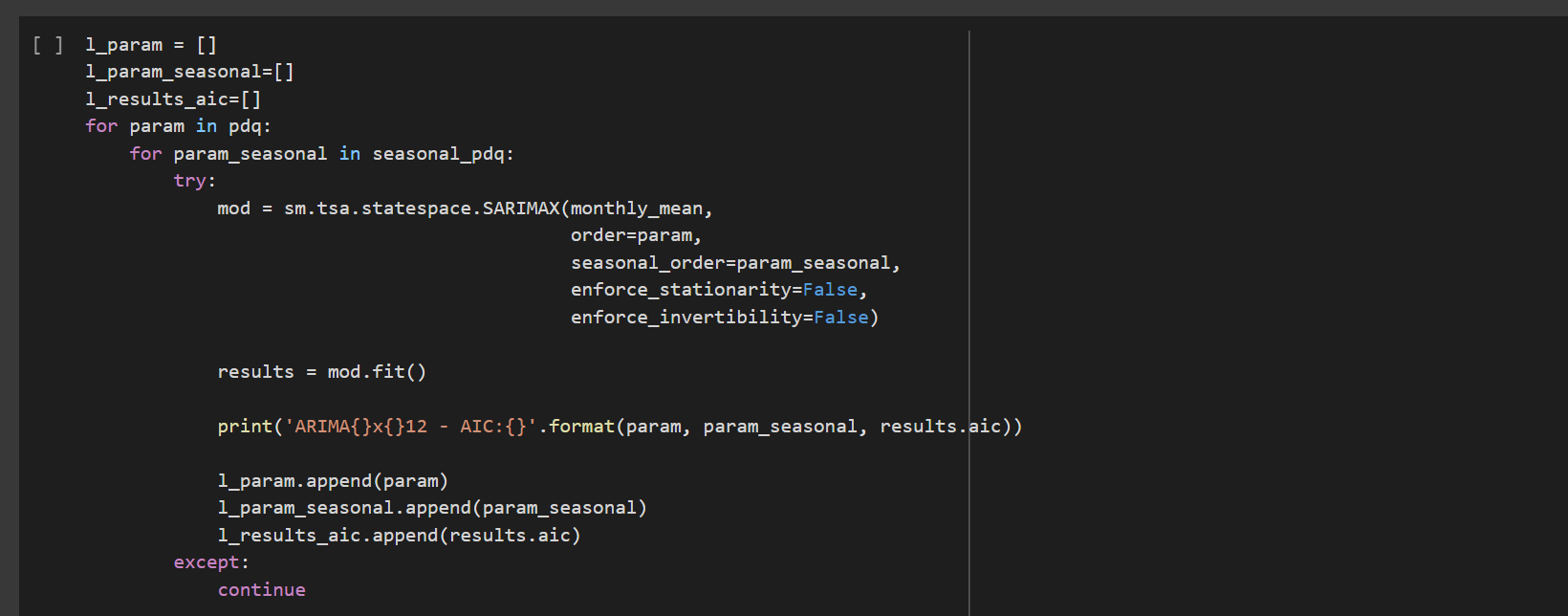
**FIGURE 21: OUTPUT**

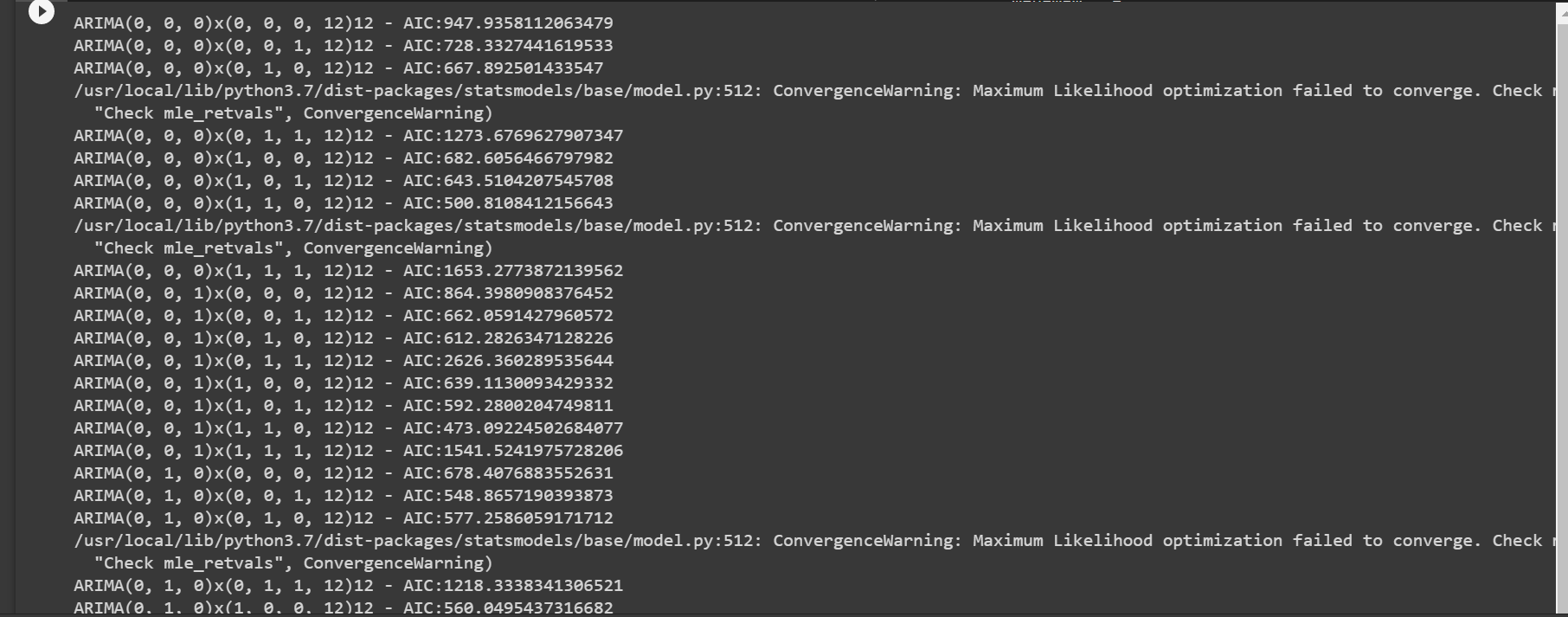


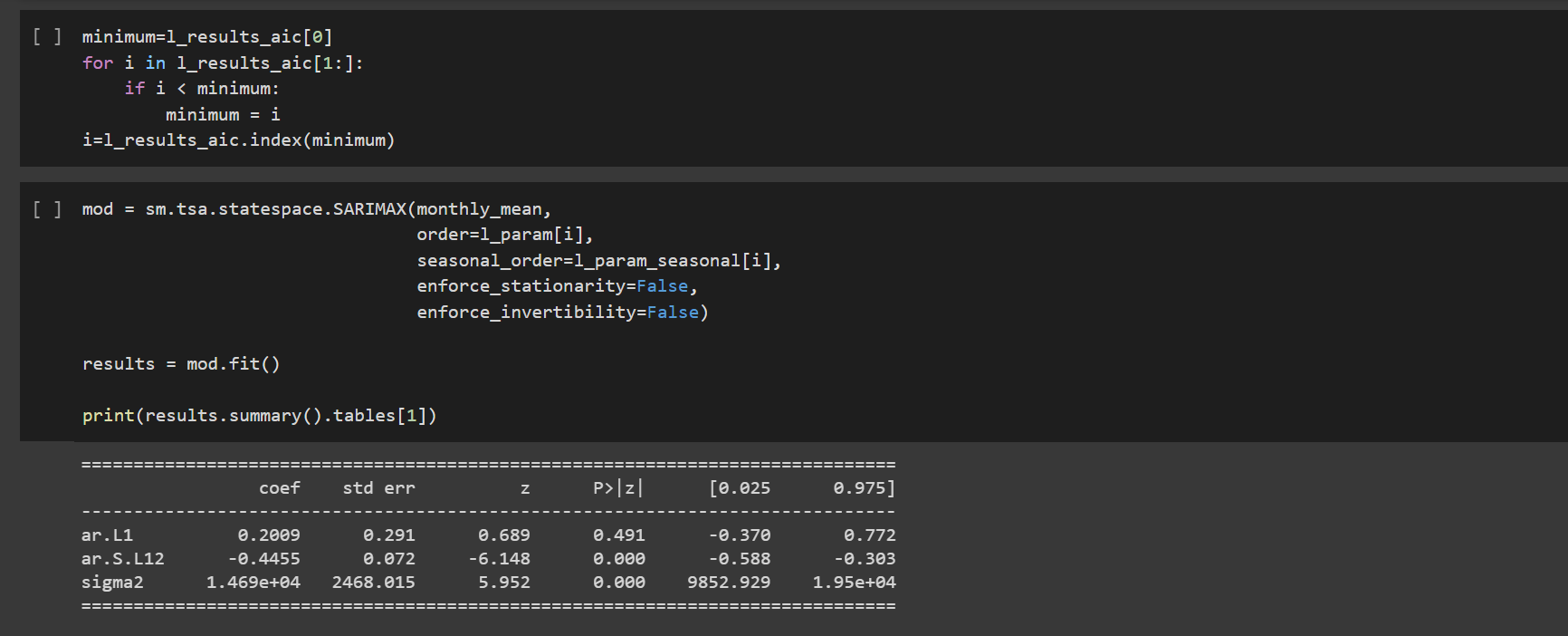




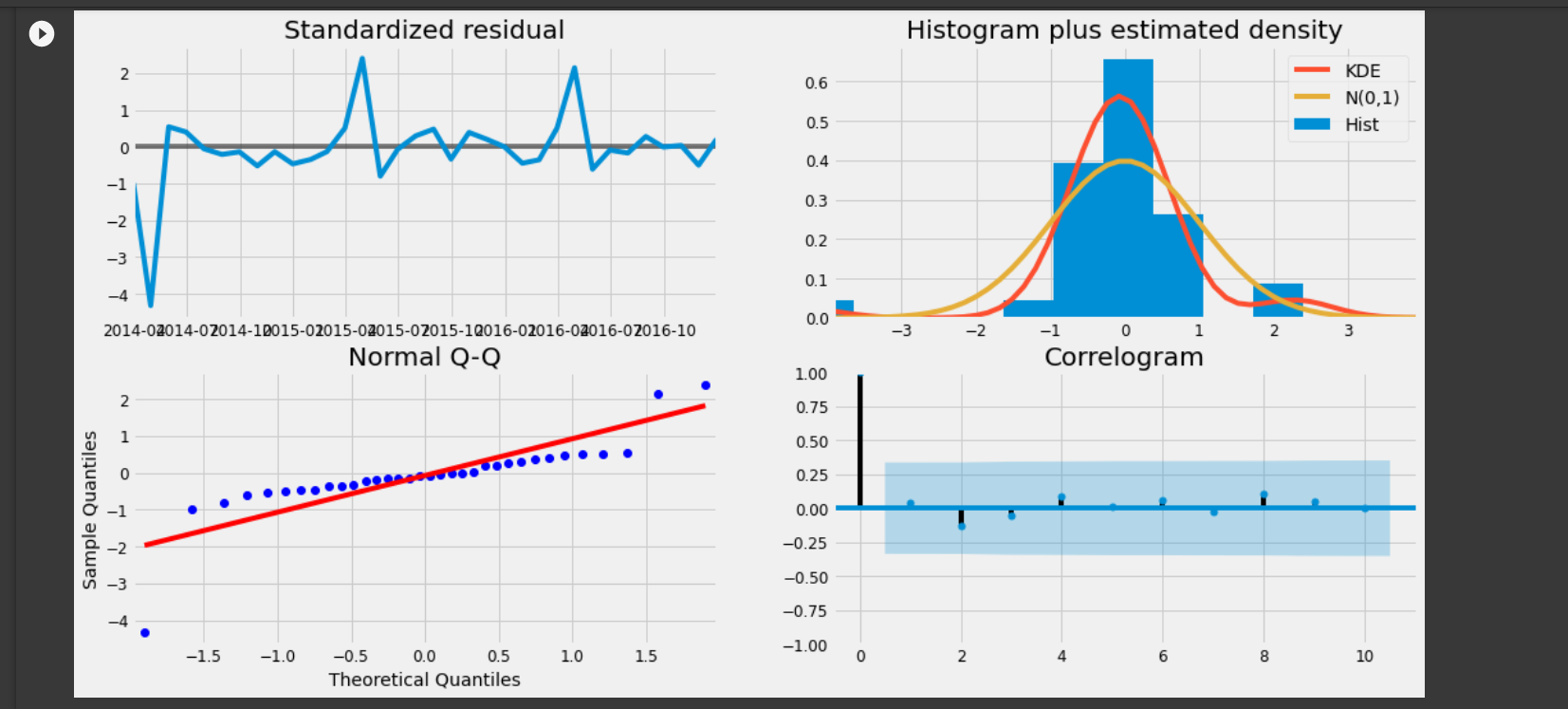
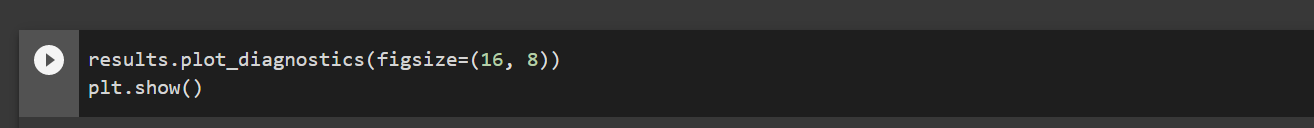
 **FIGURE 22: ARIMA MODEL VISUALISATION**

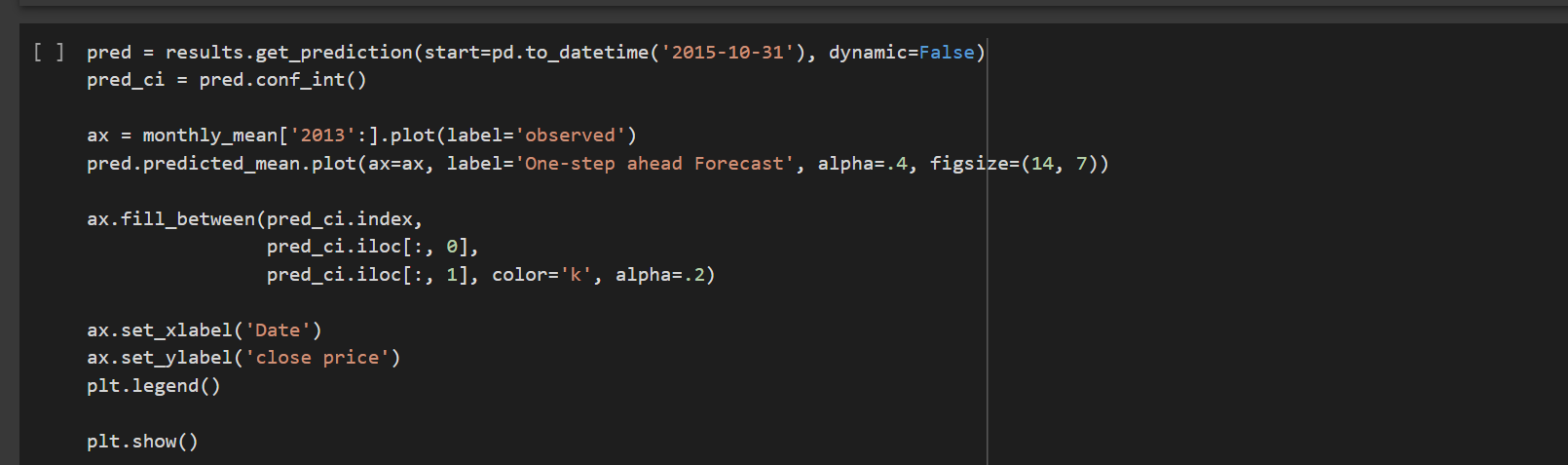


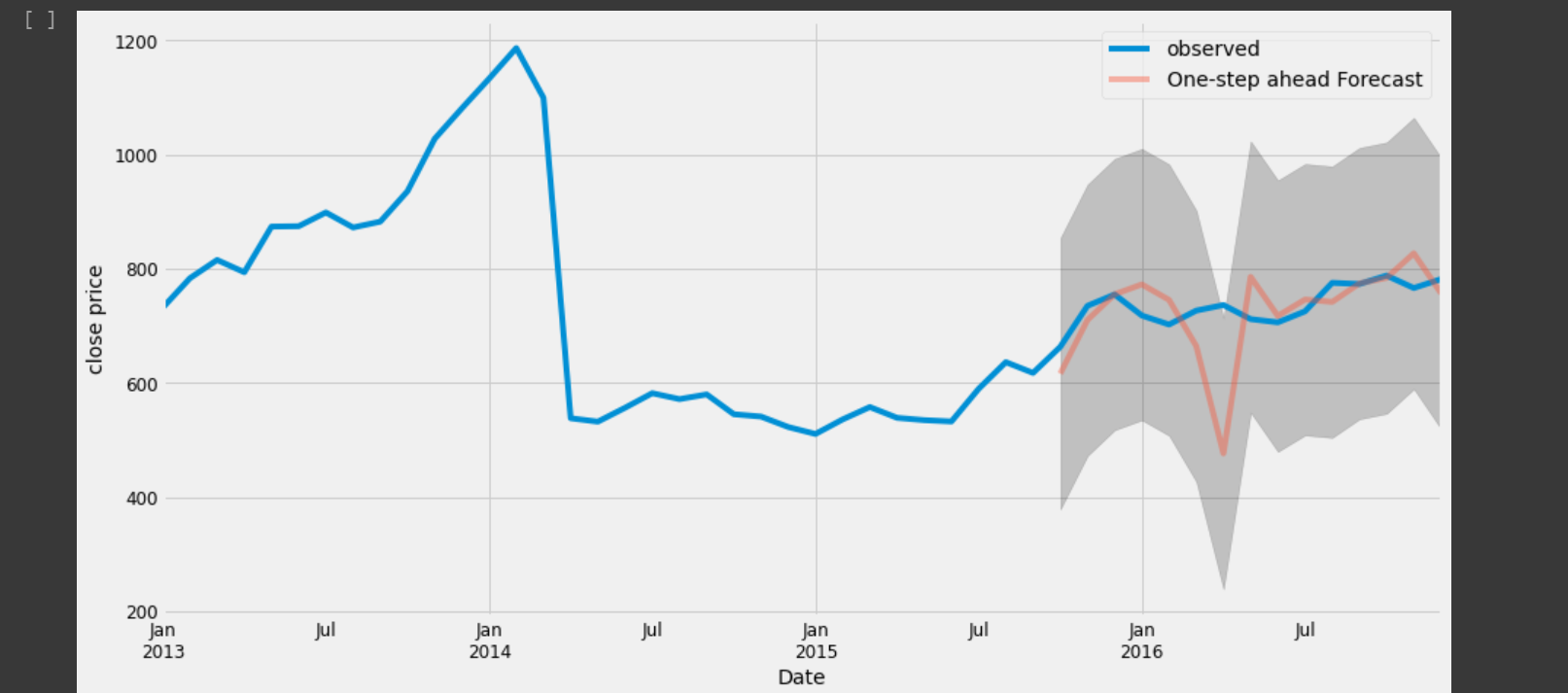




**FIGURE 23: GETTING OPEN AND CLOSE VALUE**

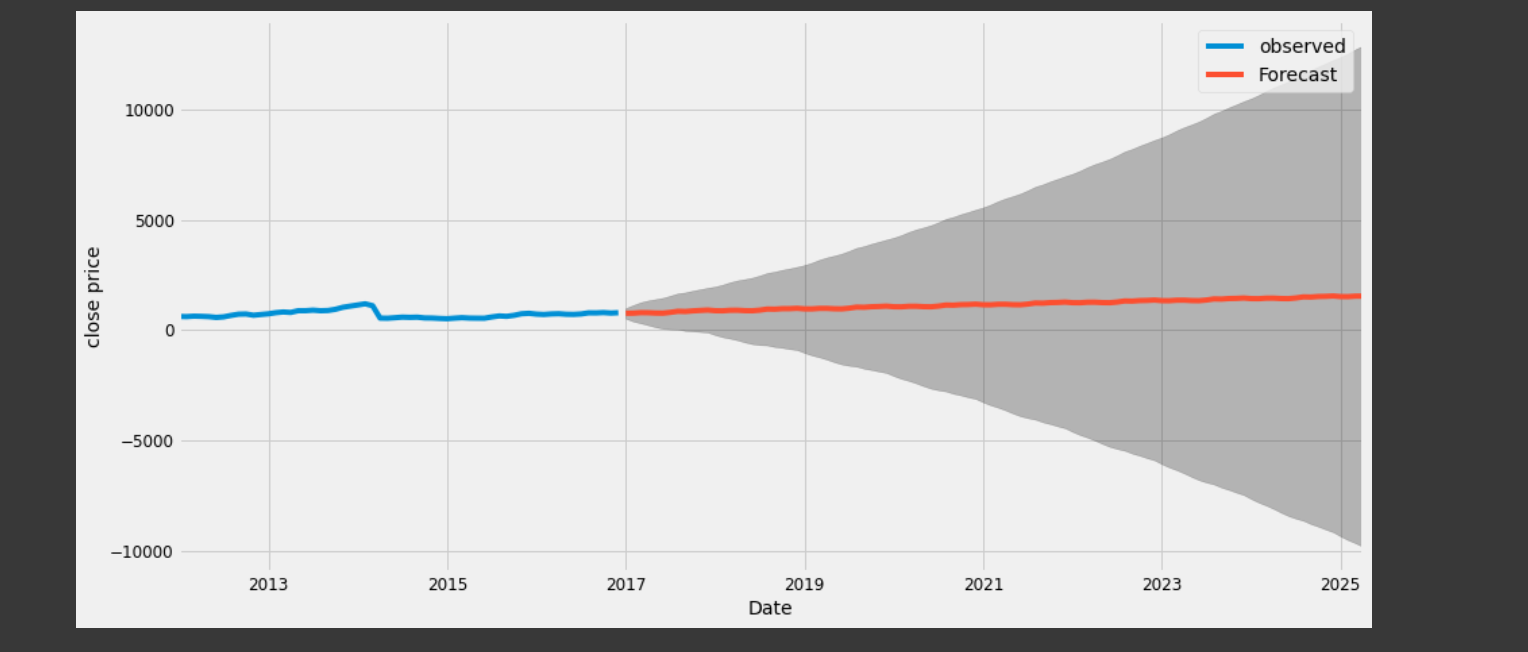






**FIGURE 24: GRAPHICAL ANALYSIS**

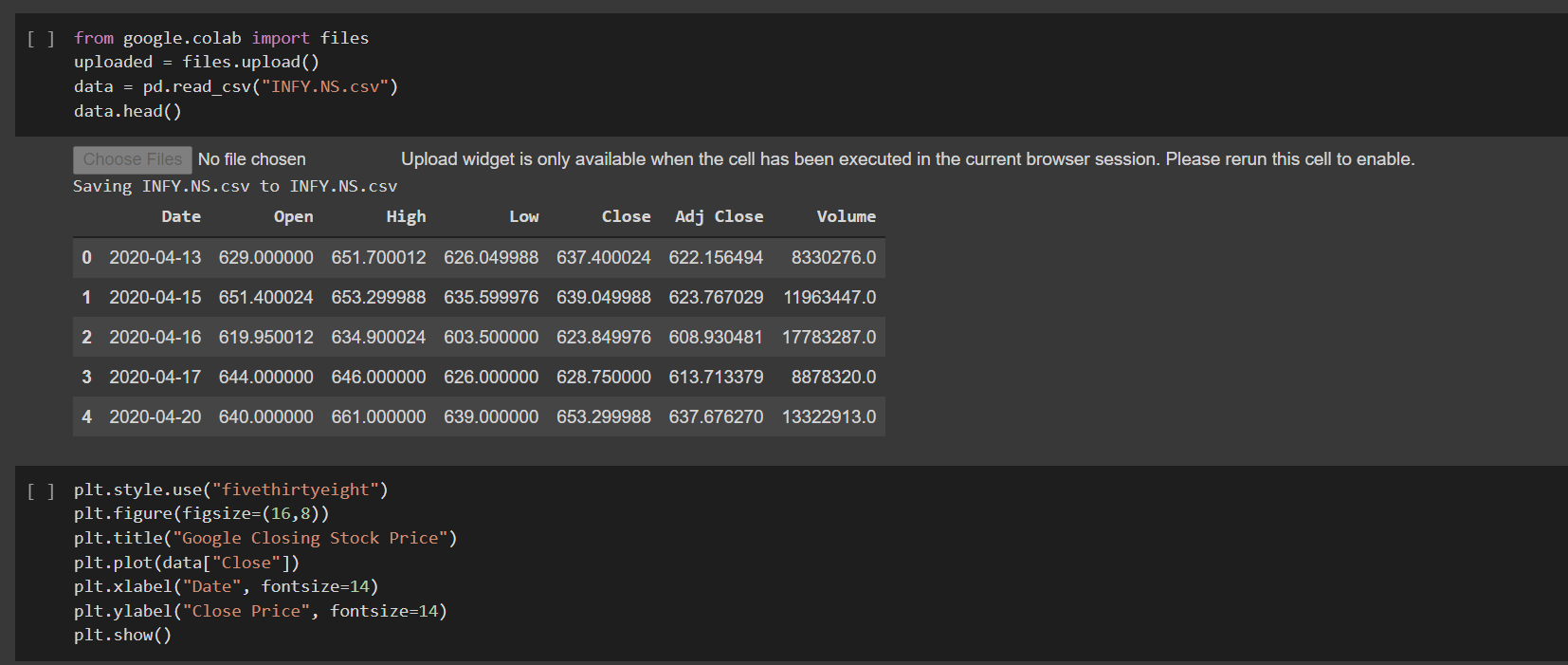




**FIGURE 25: GRAPHICAL REPRESENTATION OF MODEL**

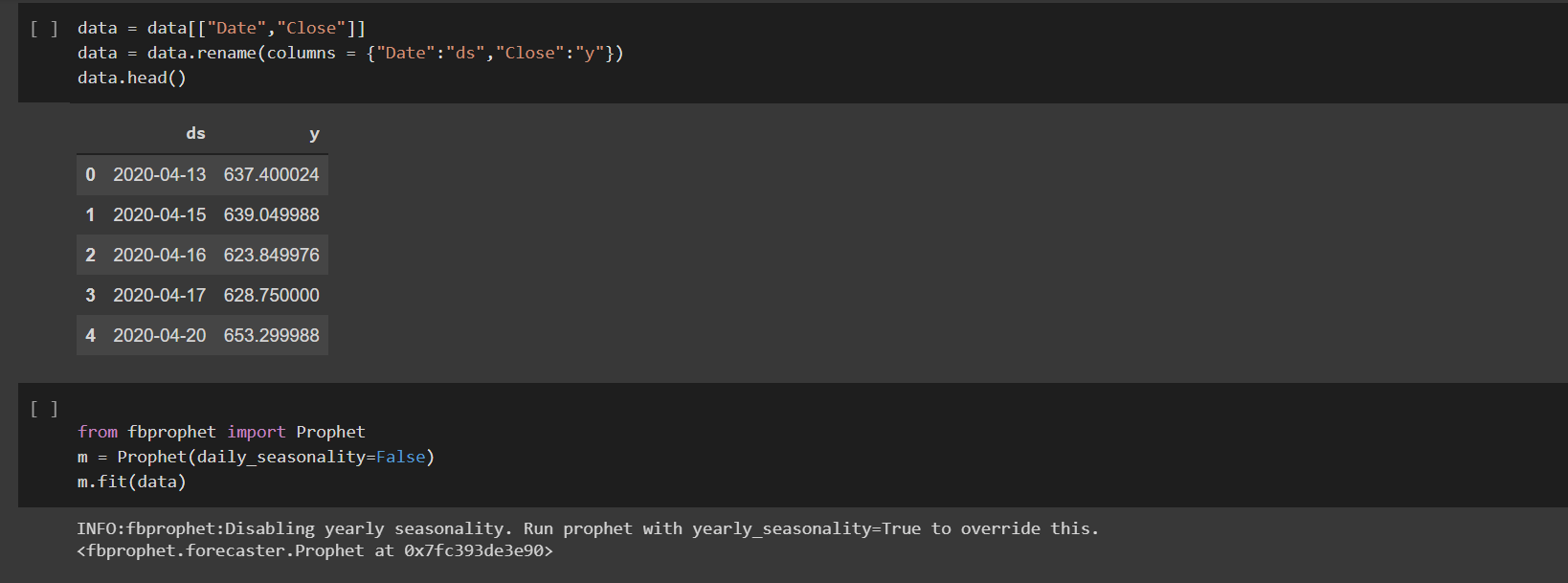
**PROPHET MODEL:**





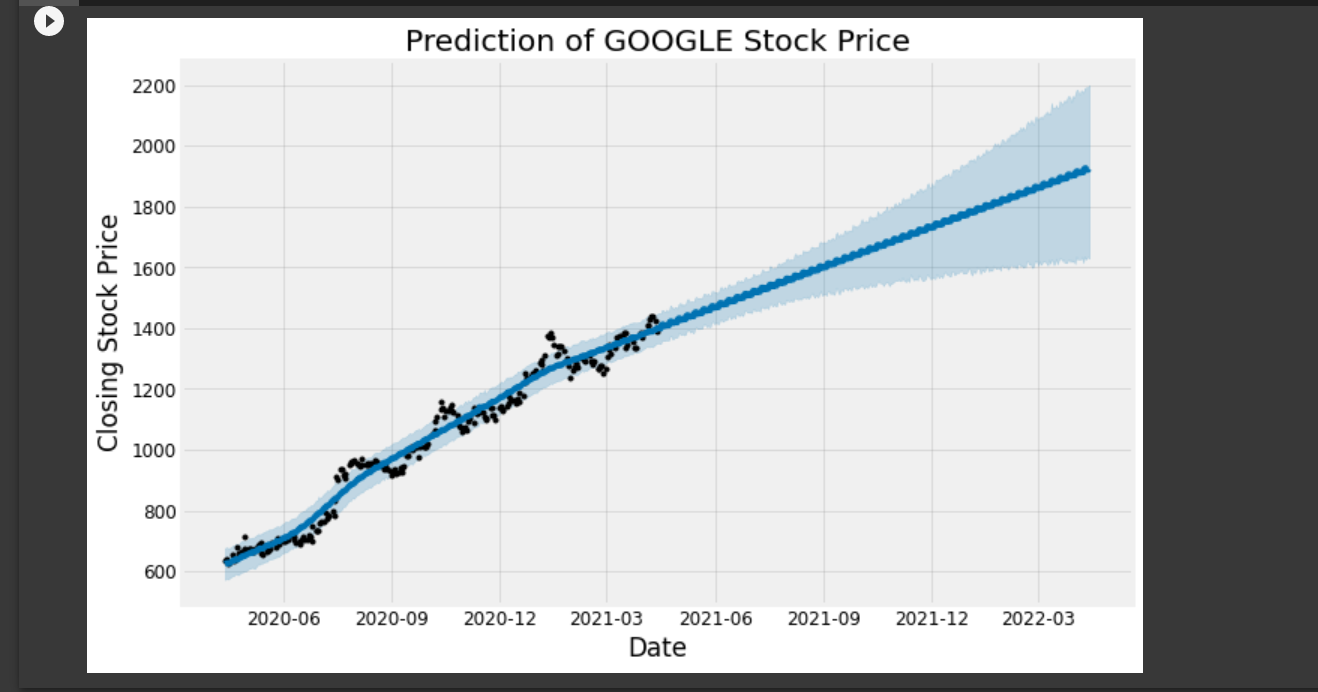


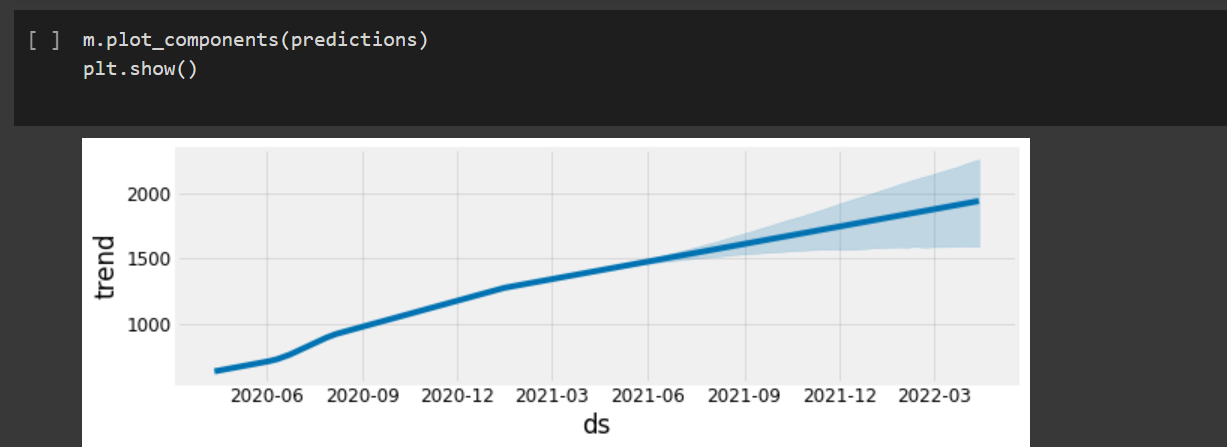
**FIGURE 26: IMPLEMENTATION AND GRAPH**

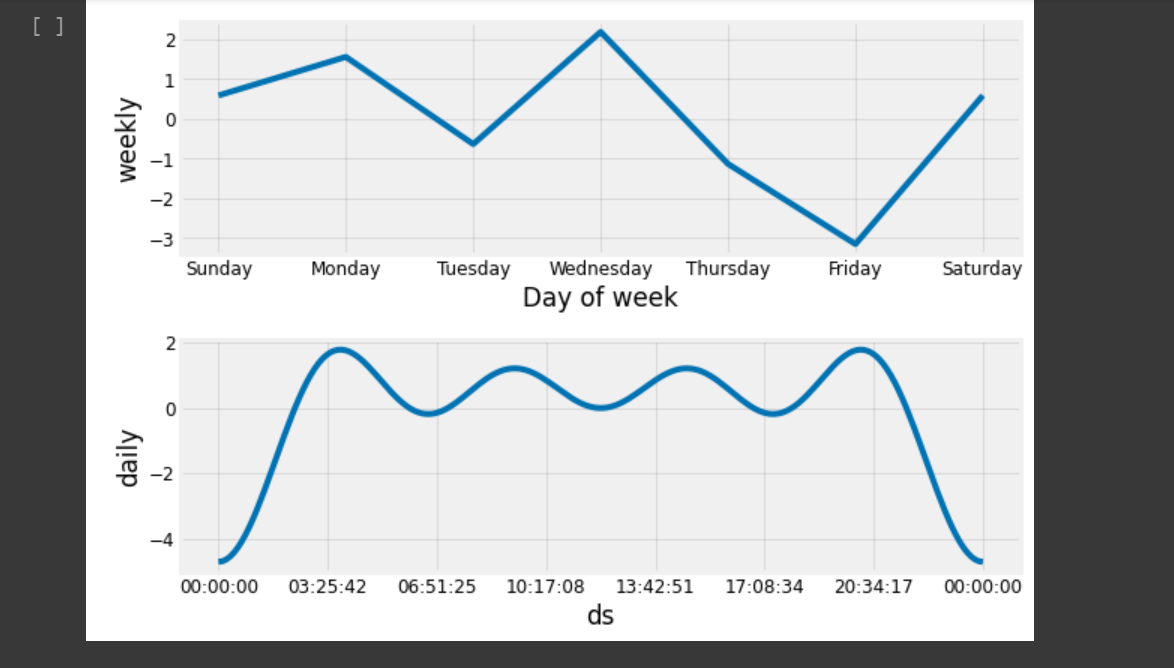




**FIGURE 27 PROPHET MODEL IMPLEMENTATION**



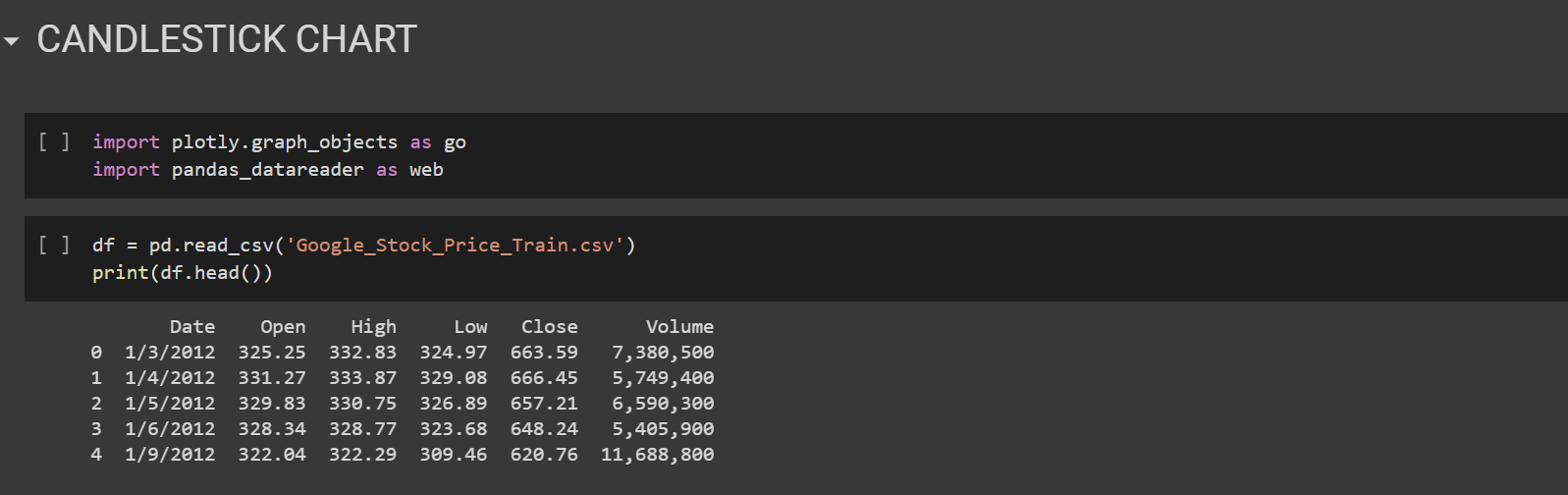


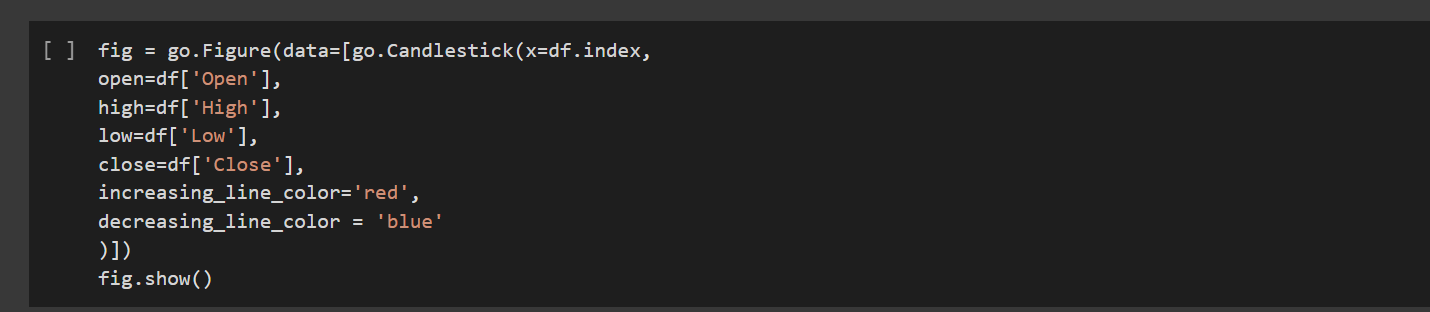


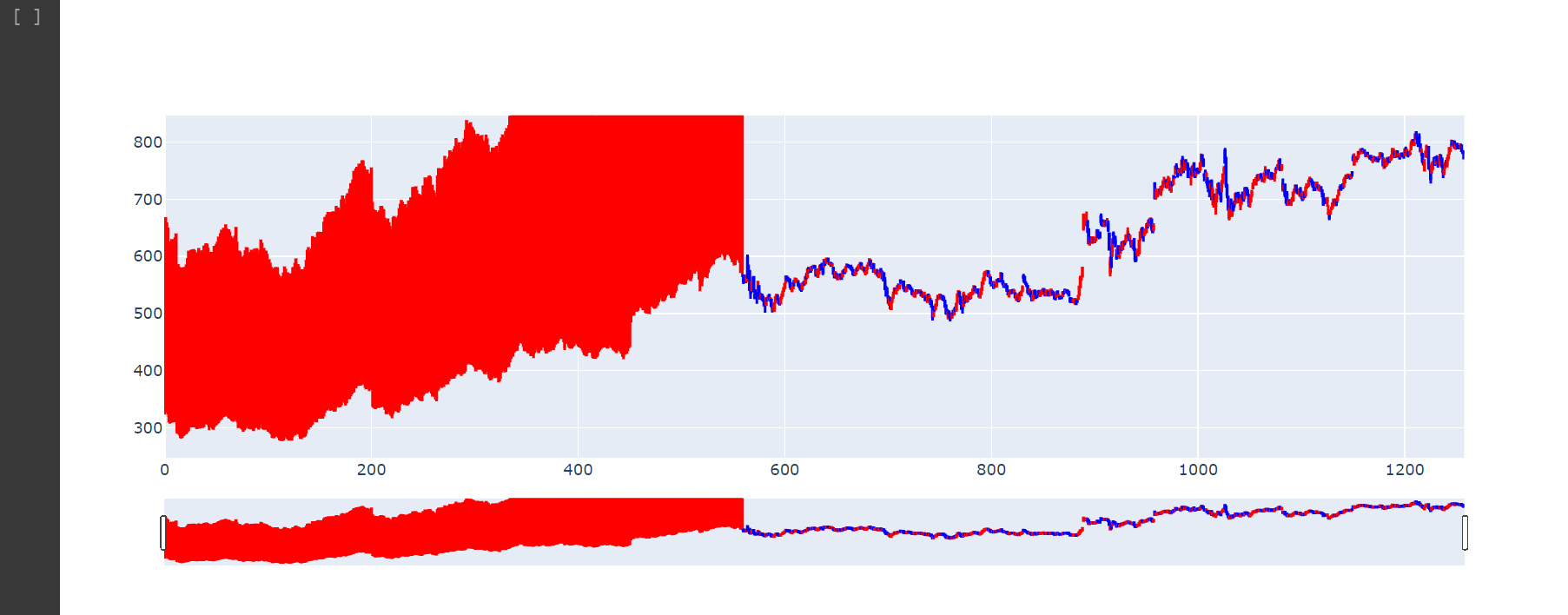
**FIGURE 28: PROPHET MODEL OUTCOME**

1. **COMPARING THE RESULTS**

**CANDLE STICK CHART:**







**FIGURE 29: CANDLESTICK CHART ANALYSIS**

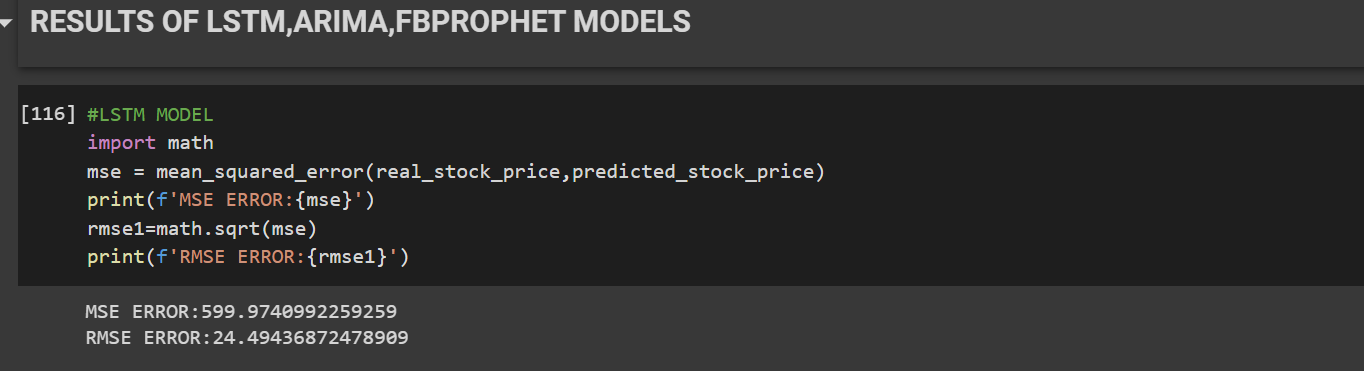
**RESULTS**

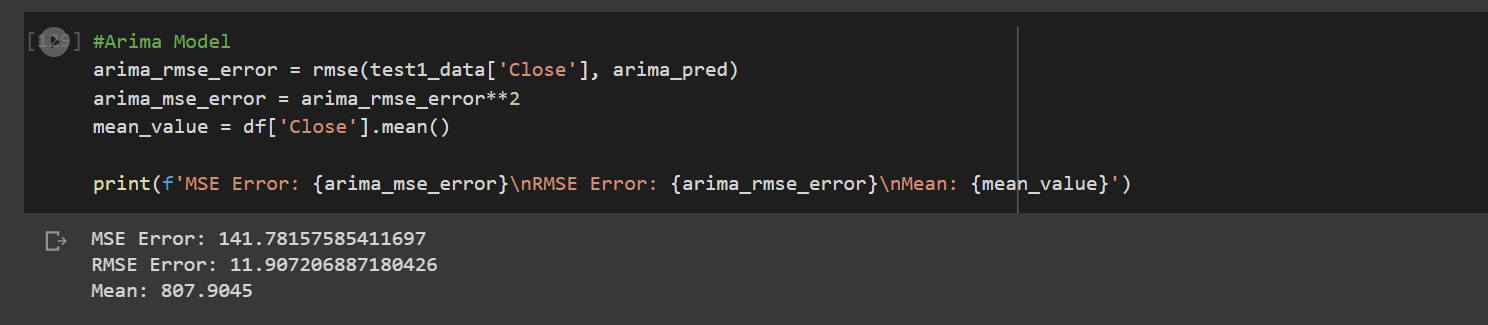
On trying out all the three models we get to know that the most effective model is Prophet Model. It gives approximately same value thus predicting the prices to the nearest value.

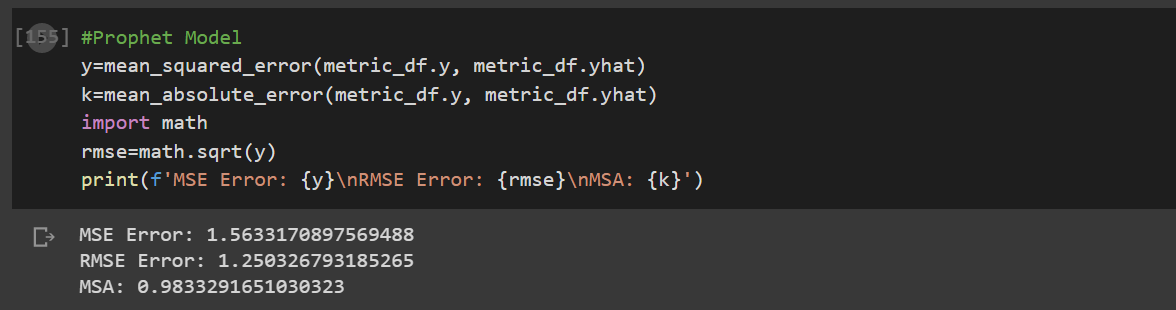
We see that ARIMA outperforms LSTM and even more powerful Prophet model while making the predictions. From the experiment, we can see that SARIMAX model forecasting has better accuracy than the Prophet model forecasting. The RMSE for the SARIMAX model was around 8% while Prophet Model had RMSE of 11.4%.

Facebook Prophet is a good package to use because the syntax is very similar to SKlearn and easy to plot in Plotly for trend analysis.

Below is the demonstration of the same:







**FIGURE 30: RESULTS AND COMPARISON**

**CONCLUSION**

Stock markets are hard to monitor and require plenty of context when trying to interpret the movement and predict prices. LSTMs perform better as they are able to keep track of the context-specific temporal dependencies between stock prices for a longer period of time while performing predictions. An analysis of the results also indicates that both models give better accuracy when the size of the dataset increases. With more data, more patterns can be fleshed out by the model, and the weights of the layers can be better adjusted. At its core, the stock market is a reflection of human emotions. Pure number crunching and analysis have their limitations; a possible extension of this stock prediction system would be to augment it with a news feed analysis from social media platforms such as Twitter, where emotions are gauged from the articles. This sentiment analysis can be linked with the LSTM to better train weights and further improve accuracy.

In conclusion, I would like to say that the popularity of stock market trading is growing extremely rapidly which is encouraging researchers to find out new methods for the prediction using new techniques, the forecasting technique is not only helpful to researchers but also helps investors or any person dealing with stock market in order to help predict the stock indices a forecasting model with good accuracy is required in this work we have used one of the most precise forecasting technology is using RNN and LSTM which helps investors analysts or any person interested in the stock market by providing them a good knowledge of the future situation of the stock market in such little project at least we were able to get the trend right.

**REFERENCES**

[1] Tse, C. K., Liu, J., and Lau, F. C.M., 2010. A network perspective of the stock market. Journal of Empirical Finance 17 (2010), p. 659-667.

[2] Boginski, V., Butenko, S., and Pardalos, P. M., 2004. Statistical analysis of financial networks. Computational Statistics & Data Analysis 48 (2005), p. 431-443.

[3] Giles, C., Lawrence, S., and Tsoi, A., 2001. Noisy Time Series Prediction using a Recurrent Neural Network and Grammatical Inference. Machine Learning, Volume 44, Number 12, July/August, pp. 161-183.

[4] Timmermann, A., and Granger, C. W.J., 2004. Efficient market hypothesis and forecasting. International Journal of Forecasting, Volume 20, Issue 1, January-March 2004, p. 15-27.

[5] Hochreiter, S., and Schmidhuber, J., 1997. Long Short-Term Memory. Neural Computation 9, p. 1735-1780.

[6] Hermans, M., and Schrauwen, B., 2013. Training and Analyzing Deep Recurrent Neural Networks. NIPS 2013. [http://machinelearning.wustl.edu/mlpapers/paper files/NIPS2013](http://machinelearning.wustl.edu/mlpapers/paper%20files/NIPS2013) 5166.pdf

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Explanation** | **Page No.** |
| Figure 1 | Importing Libraries | 9 |
| Figure 2 | Loading Dataset | 9 |
| Figure 3 | Information Regarding Dataset Used | 10 |
| Figure 4 | Correctness Graph | 10 |
| Figure 5 | Graph Representation | 11 |
| Figure 6 | Graph Representation | 11 |
| Figure 7 | Removal of Punctuation Marks | 12 |
| Figure 8 | Open and Close Representation | 12 |
| Figure 9 | Data Cleaning | 13 |
| Figure 10 | Feature Scaling | 13 |
| Figure 11 | Importing required libraries | 20 |
| Figure 12 | LSTM model implementation | 20 |
| Figure 13 | Compilation of RNN | 21 |
| Figure 14 | Dataset Import | 21 |
| Figure 15 | Prediction Image | 21 |
| Figure 16 | Prediction Image | 22 |
| Figure 17 | Prediction | 22 |
| Figure 18 | ARIMA model implementation | 23 |
| Figure 19 | ARIMA model working | 24 |
| Figure 20 | ARIMA model output | 24 |
| Figure 21 | Output | 24 |
| Figure 22 | ARIMA model visualization | 25 |
| Figure 23 | Getting open and close value | 26 |
| Figure 24 | Graphical analysis | 27 |
| Figure 25 | Graphical representation of model | 28 |
| Figure 26 | Implementation and graph | 29 |
| Figure 27 | Prophet model implementation | 30 |
| Figure 28 | Prophet model outcome | 31 |
| Figure 29 | Candlestick Analysis | 32 |
| Figure 30 | Results and comparisons | 33 |
| Diagram 1 | Flow algorithm of LSTM | 15 |
| Diagram 2 | Block diagram | 15 |
| Diagram 3 | Flow diagram of ARIMA | 16 |
| Diagram 4 | Flow Diagram of Prophet | 17 |

APPENDIX CODE:

<https://colab.research.google.com/drive/1nKialsTMdYCu4p_UOVQIM3ZY_oIdklHm?usp=sharing>