**Predicting Stock Prices using ARIMA Model**

**Tonny Kattitharayil Podiyan**

***Abstract:* The stock market is a place which can build a person’s fortune but to blindly invest in any stock is a very risky move. The value of the stock depends on many different parameters like company performance, growth, sentiment of the people and other physical and mental parameters. In this research work, you will see how stock prices can be predicted using the ARIMA model.**

**Key Words: Stock Market; Machine Learning; Python; Predictive Analysis; Time Series; ARIMA.**

**1.INTRODUCTION**:

Stock markets are one of the essential components of a country’s economy. It enables companies to generate wealth in exchange of some part of ownership of the company. Investing in the stock market helps keep the money within the country, creating more job opportunities by expanding company businesses/operations. However, considering the number of companies listed on the market investing in the stock market is not an easy task.

Although it is difficult to predict whether the price of a stock would increase or decrease, investors are always on lookouts for new techniques to minimize the risk of investment and increase profits. ARIMA is one such model which can be used to predict future stock prices. In this research we will be using ARIMA model to predict future stock prices of GOOGLE.

**2.LITERATURE REVIEW:**

This literature review will follow narrative approach to gain insight into research topic. “Time Series Analysis is a statistical technique that deals with time series data. [1]”. To further understand time series analysis, we first need to understand what a Time series is. Time series is a sequence of data points recorded in equal intervals of time. The recorded data used can be hourly, daily, weekly, monthly, or annually. Time series analysis consists of methods to analyze historical data to find stats and other characteristics of data. “Time series forecasting is a trend analysis technique that focuses on analyzing cyclical fluctuations &

seasonality issues and looking at past data and related patterns to make predictions [2]”.

Success is not guaranteed in this method, though it throws a hint about future trends.

Time series forecasting uses the Box-Jenkins model, which includes three methods for predicting future data: autoregression, differencing and moving averages. In this study, we will see how to use the ARIMA model to predict stock prices.

**3.METHODOLOGY**:

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 [3].

To predict future stock prices, we have built an ARIMA model in Python using multiple libraries. Stock prices of Google for the last 18 years was considered. Detailed information regarding the dataset and the process are presented in the below sub-sections.

***3.1 Dataset Description:***

Dataset for GOOGLE stock was retrieved using the Google Finance function with the help of google sheets. The formula used to retrieve the data is as follows-

*=GOOGLEFINANCE ("GOOGL", "price", "01/01/2004", "01/12/2022", "DAILY")*

Where GOOGL is the ticker (Abbreviation for the stock name), price- is the closing price of the stock, 01/01/2004 - is the start date and 01/12/2022 is the end date. Figure 1 shows how data was retrieved using the GOOGLEFINANCE function. After using the above formula, we were able to retrieve data from 19th Aug 2004 till 30th Nov 2022.

There was a total of 4605 values. Column 1 was the Date component and column 2 was the price. This data was then saved as a csv file for further processing.

**Fig1: Sample Data set of Google Stock Prices**

Graphical user interface, application, table, Excel

Description automatically generated

***3.2 Exploratory Data Analysis:***

Data pre-processing is a technique by which redundant data is removed from the dataset so that the data, which will be used for forecasting purposes, is clean and error free. For this process we have used excel and python. The Original Date column was in DD-MM-YY HH:MM: SS format. The date column was converted first into DD-MM-YYYY by dropping the HH:MM: SS values.

Now we have aggregated the data into monthly Price by taking Average Price of the stock per month by using pivot tables in excel. Now our revised data has Month as the predictor variable and Price as the target variable. Figure 2 gives a visual representation of how daily stock price data was converted into monthly values.

**Fig2: Aggregating stock prices to Monthly Prices by taking mean**

Graphical user interface, application, table, Excel

Description automatically generated

This data was saved as a .csv file and further imported to python using the pd.read\_csv function. We call our original time series or this dataframe as df. In order to clean the dataset, dropna() function was called by the data-frame object. This function drops the specific row or column which contain the null value [4].

While importing data we have specified arguments index\_col=['Month'], parse\_dates= ['Month’] so that python could interpret first column as a date component and not as a string, also not creating a separate index for a dataframe. We also check the head and tail values of the imported dataset to verify that the data is consistent and there are no unwanted values using the df.head() and df.tail() function. Figure 3 shows code snippet for the same.

**Fig3: Importing .CSV file to Python, dropping null values and checking for potential outliers.**

A screenshot of a computer

Description automatically generated with medium confidence

***3.3******Stationary Test:***

**“**A stationary time series is one whose properties do not depend on the time at which the series is observed.” Thus, time series with trends, or with seasonality, are not stationary, the trend and seasonality will affect the value of the time series at different times [5]. To check if our time series is stationary, there are multiple test’s available like the KPSS (Kwiatkowski–Phillips–Schmidt–Shin) test, Phillips–Perron test, and the Augmented Dickey-Fuller test. We can also check stationarity by plotting our time series using the matplot library in python. Figure 4 gives a visual representation of our time series, indicating a clear upward trend, also confirming that the time series isn’t stationary.

**Fig4: Plotting our time series to check stationarity**

Graphical user interface, chart

Description automatically generated with medium confidence

In this research we are further going to check for stationarity using the ADF test. The ADF test belongs to a type of test called "Unit Root Test", which is a suitable method for evaluating the stationarity of a time series. In Python, The statsmodel package provides a reliable implementation of ADF checks through the adfuller() function in statsmodels.tsa.stattools. The ADF test returns the Test statistic, P-value, Number of lags used and 1%, 5%, and 10% critical values and Estimation of the maximized information criteria. If the returned P-value is higher than 0.05, the time series isn’t stationary., 0.05 is the standard threshold [6].

After running the ADF test P value was found to be 1 for our original timeseries df. To further understand how many times we need to difference the data to make it stationary we can use the ndiffs function from the pmdarima.arima library. The ndiffs functions gives the number of differencing required to make the time series stationary. In our case the optimum order of differencing was found to be 2. After differencing twice using the diff(). function, we again ran the ADF test to check whether the time series is now stationary. P value of our differenced time series was 3.008714e-06. Figure 5 and Figure 6 shows the code snippet for the same. Thus, we can conclude that our time series is now stationary.

**Fig 5: ADF test to check stationarity**

Text

Description automatically generated

**Fig 6: Differencing the time series & again checking for stationarity.**

**Graphical user interface, text

Description automatically generated**

***3.4******ARIMA Model:***

Now that our time series has become stationary our next step is to build an ARIMA model, short for ‘Auto Regressive Integrated Moving Average.’ “ARIMA is a class of models that explains a given time series based on its own past values, that is, its own lags and the lagged forecast errors, so that equation can be used to forecast future values. An ARIMA model is characterized by 3 terms: p, d, q

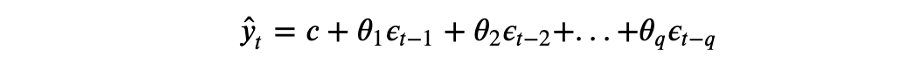
where, p is the order of the AR term, q is the order of the MA term ,d is the number of differencing required to make the time series stationary [7]”.

**Autoregression**(p)-is a regression model with lagged values ​​of y, up to the pth time in the past, as predictors. Here, p = number of observations shifted in the model, ε is the white noise at time t, c is a constant, and φ**s** are the parameters [8].



**Integrated**(d) - The difference is taken ‘d’ times until the original series becomes stationary. A stationary time series is a series whose property is independent of when the series is observed [8].

**Moving average(q)** - A moving average model uses a regression-like model on past forecast errors. Here, ε is white noise at time t, c is a constant, and θ’s are parameter [8].



To determine values of p,d,q we have two methods. First method is to use the auto\_arima function from pmdarima library. Auto-ARIMA works by performing differencing tests to determine value of d, and then fits the models in the ranges start\_p , max\_p , start\_q , max\_q . If the seasonal option is enabled, ARIMA automatically also attempts to determine the optimal P and Q hyperparameters after performing Canova-Hansen to determine the optimal order of seasonal differences [9].

The second method is to determine values of p and q (d =2 as we differenced the original time series twice) using ACF and PACF plots. In these plots, the blue shaded area represents the significant threshold levels. Anything that crosses over these two lines reveals the significant correlations. When looking at these plots, we ignore the long spike at lag 0 [10]. To get p value the spikes are at 1,2 and 8 in the PACF plots. Thus, p values can be 1,2 or 8, similarly for q the spikes are at 1,2 and 11, so q values can be 1,2 or 11. Figure 7 and Figure 8 gives a visual representation for the same.

**Fig7: ACF plot to determine value of q**

Chart, box and whisker chart

Description automatically generated

**Fig8: PACF plot to determine value of p**

Chart

Description automatically generated

So as per the acf and pacf plots we take p =8 and q=2. Now that we have successfully determined values for p,d,q we now split our data into training and testing part where 80% of the data is used for training the model and remaining 20% of the data is used for testing purposes.

After fitting the model using the ARIMA().fit() module we can now make predictions using the .predict() function.

***3.5******Future Prediction:***

After successfully fitting the model the next process is to predict future prices using our training data set with the help of model\_fit.predict() function. We would use length of the training set as the starting parameter and one less than the length of the testing set as the ending parameter. Figure 9 shows the code snippet for the same.

Since the data is not scaled as per the original time series, to scale the differenced series and predictions, we define a new function in Python and call it inverse\_diff. What this function does is it adds the first value of the last differenced time series to the current differenced time series and then takes a cumulative sum of the series. Figure 10 shows the code snippet for the same. We need to call the inverse\_diff function twice since we had differenced the original time series two times to make it stationary.

After making these predictions and scaling back the data, we plot our predictions against our original data to get a visual representation of how our predictions look when compared to our testing data. Figure 11 Shows the visual representation for the same, where the green solid line shows our actual time series, and the red dotted line shows the forecasted price.

**Fig9: Specifying start & end length for predictions**

A screenshot of a computer

Description automatically generated with medium confidence

**Fig10: Creating inverse diff function for scaling back the differenced time series**

Text

Description automatically generated

**Fig11: Plot of predictions against Time Series**

Chart

Description automatically generated

***3.6******Model Performance:***

It is mandatory to check the error in our model to evaluate model performance. For this there are multiple methods available like the Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and the Mean Absolute Percentage Error (MAPE). In this step we check the performance of our model using the mean squared error method.

The mean squared error, or MSE, is calculated by taking the average of the squared forecast errors. By squaring the forecast error values, the values become positive and in turn gives more weightage to large errors. Thus, if there are any big error forecasts then that would substantially increase the mean of the squared forecast errors. In effect, the score gives worse performance to those models that make large wrong forecasts.

From the sklearn.metrics module in python we can use the mean\_squared\_error function to correctly calculate the mean squared error of the forecasted predictions.In this case the mean squared error was found to be 5.063 and the mean of the testing data was found to be 0.0210. Figure 12 shows the code snippet for the same.

**Fig12: Calculating Mean Absolute Error**

Graphical user interface, application

Description automatically generated

This error cannot be ignored and it can be concluded that there are better models available, or better p,d,q values can be obtained to minimize the error and make future predictions . It tells us that there is still room for improvement, and we can find better p, d, q values or use a different model altogether.

**4.CRITICAL REFLECTION AND RECOMMENDATIONS:**

There are multiple methods to find the optimum values of p,d and q and using the Auto Arima function doesn’t necessarily give the best model, using the acf and pacf plots can help build a better model with less error.

Every data must be analyzed, and the model must be trained differently, even if the dataset belongs to the same company they might need to be analyzed differently altogether depending upon the length of the dataset. There are multiple python libraries which are readily available which help speed up the forecasting process.

**Potential benefits of using ARIMA models**:

1. Good for short-term predictions.

2.Requires only historic data from a time series to make predictions.

3.The time series model is not fixed.

**Potential disadvantages of using ARIMA models:**

1. It is difficult to predict changes in the trend.

2. There are quite a few subjective factors involved in determining the order (p,d,q) of the model.

3. Not suitable for long-term forecasting.

4. Cannot be used when seasonality is involved in the time series.

**5.CONCLUSION:**

Investing in the stock market is highly risky but there are multiple tools available like the ARIMA model which help minimize the risk of investing by making satisfactory predictions. One drawback of ARIMA model is that it can be used only for short term forecasting.

# References

|  |  |
| --- | --- |
| [1] | Influx Data, "What is time series data?," [Online]. Available: https://www.influxdata.com/what-is-time-series-data/. [Accessed 13 December 2022]. |
| [2] | A. Hayes, "What Is a Time Series and How Is It Used to Analyze Data?," 12 June 2022. [Online]. Available: https://www.investopedia.com/terms/t/timeseries.asp#toc-time-series-forecasting. [Accessed 13 December 2022]. |
| [3] | J. Brownlee, "How to Create an ARIMA Model for Time Series Forecasting in Python," 9 January 2017. [Online]. Available: https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/#:~:text=An%20ARIMA%20model%20is%20a,making%20skillful%20time%20series%20forecasts.. [Accessed 02 December 2022]. |
| [4] | A. K. ARAVIND GANESAN, "Stock Price Prediction using ARIMA Model," vol. 8, no. 8, p. 4, 2021. |
| [5] | R. J. H. a. G. Athanasopoulos, Forecasting: Principles and Practice, Melbourne, Australia, 2018. |
| [6] | D. Radečić, "Time Series From Scratch — Stationarity Tests and Automation," 24 July 2021. [Online]. Available: https://towardsdatascience.com/time-series-from-scratch-stationarity-tests-and-automation-14b02fa5ca4d. [Accessed 04 December 2022]. |
| [7] | S. Prabhakaran, "ARIMA Model – Complete Guide to Time Series Forecasting in Python," 22 August 2021. [Online]. Available: https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/. [Accessed 4 December 2022]. |
| [8] | CapitalOne, "Understanding ARIMA Models for Machine Learning," 8 November 2021. [Online]. Available: https://www.capitalone.com/tech/machine-learning/understanding-arima-models/. [Accessed 9 December 2022]. |
| [9] | T. G. Smith, "pmdarima.arima.auto\_arima," 2017. [Online]. Available: https://alkaline-ml.com/pmdarima/modules/generated/pmdarima.arima.auto\_arima.html. [Accessed 9 December 2022]. |
| [10] | TrainDataHub, "How to Interpret ACF and PACF plots for Identifying AR, MA, ARMA, or ARIMA Models," 1 December 2021. [Online]. Available: https://medium.com/@ooemma83/how-to-interpret-acf-and-pacf-plots-for-identifying-ar-ma-arma-or-arima-models-498717e815b6#:~:text=The%20basic%20guideline%20for%20interpreting,q%20for%20MA(q).. [Accessed 4 December 2022]. |