International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 08 | Aug 2021 www.irjet.net p-ISSN: 2395-0072

Stock Price Prediction using ARIMA Model

ARAVIND GANESAN[1], ADARSH KANNAN[2]

[1]*STUDENT, Department of Computer Science and Engineering, Sri Chandrasekharendra Saraswathi Viswa Mahavidyalaya, Enathur, Kanchipuram, 631561*

[2]*STUDENT, Department of Computer Science and Engineering, Sri Chandrasekharendra Saraswathi Viswa Mahavidyalaya, Enathur, Kanchipuram, 631561*

Abstract - *The Stock Market, as we know, is volatile in nature and the prediction of the same is a cumbersome task. Stock prices depend upon not only economic factors, but they relate to various physical, psychological, rational and other important parameters. In this research work, the stock prices are predicted using the Auto Regressive Integrated Moving Average (ARIMA) Model. Stock price predictive models have been developed and run-on published stock data acquired from Yahoo Finance. The experimental results lead to the conclusion that ARIMA Model can be used to predict stock prices for a short period of time with reasonable accuracy.*

Key Words: **Machine Learning; Stock Market; Predictive Analysis; Financial Time Series Forecasting**

1.INTRODUCTION

One of the vital elements of a market economy is stock market. The reason behind this is mainly because of the foundation it lays for public listed companies to gain capital via investors, who invest to buy equity in the company. With the aid of refinements in the industries, stock market is expanding rapidly. In order for the investors to gain returns (profits), they should take in consideration the disparities involved in the stock market on regular basis. The stock market is volatile in nature and the prediction of the same is not an easy task. Stock prices depend upon a variety of factors including economic, physical, psychological, rational and other important aspects. Although, the stock trend is difficult to predict, investors seem to find new techniques in order to minimise the risk of investment and increase the probability of profiting from the investments [1]. The variability in stock market makes it an interesting field for researchers to forge new forecasting models.

Time-series analysis is an important subset of prediction algorithms and functions. It is regarded as an apt tool for predicting the trends in stock market and logistics. Before making any investment, an investor gathers intel on the past stock trends, periodic changes and various other factors that affect the capital of a company. An ARIMA model is a vibrant univariate forecasting method to project the future values of a time-series. Since, it is essential to identify a model to analyse trends of stock prices with adequate information for decision making, it is proposed to use the ARIMA model for stock price prediction[2,6].

2. Proposed System

In the proposed system, stock prices of ICICI Bank and Reliance Industries have been predicted using ARIMA model and implemented with various packages in python. Historical stock data of ICICI Bank and Reliance Industries have been collected from Yahoo Finance [3]. Detailed description about the dataset and further process are presented in the following sub-sections.

2.1 Dataset Description

Datasets for historical stock data have been taken from Yahoo Finance [4]. Both datasets are similar in nature but differ in terms of actual stock price. The datasets have the following components: Date, Open, High, Low, Close, Adjusted Close and volume. Table 1 and Table 2 display the sample datasets for ICICI Bank and Reliance Industries respectively. Important components that make sense to the model are Close and Date. Predictor variable is used to predict the target variable. Target variable is the variable that is to be predicted. In this case, for both datasets, 'Date' will be the predictor variable and 'Close' will be the target variable. Close is generally referred as the last price at which a stock trades during the regular hours of a trading session [5]. Datatype of each component will be described in the upcoming sections.

Table -1: Sample dataset of ICICI Bank obtained from Yahoo Finance

Date	Open	High	Low	Close	Adjusted Close (Adj. Close)	Volume
30-03-2020	330.1	333.9	311.1	313.4	313.4	36559880
31-03-2020	324.7	334.85	316	323.75	323.75	46279768
01-04-2020	319	323.75	308.1	311.15	311.15	33141186
03-04-2020	309.5	309.5	281.5	286.65	286.65	57326314
07-04-2020	308.3	329.6	296.85	326.1	326.1	57661076
08-04-2020	322.85	352.75	315.1	318.95	318.95	73931321
09-04-2020	332.4	345	322.65	342.7	342.7	52431174
13-04-2020	341.05	345.8	329.25	330.65	330.65	30994190
15-04-2020	342.7	351.9	325	327.35	327.35	49132329
16-04-2020	325.3	347.8	319.35	342	342	56494027

Table -2: Sample dataset of Reliance Industries obtained from Yahoo Finance

Date	Open	High	Low	Close	Adjusted Close (Adj. Close)	Volume
31-03-2020	1060.904	1118.358	1038.914	1101.964	1097.842	2139567
01-04-2020	1105.678	1114.346	1034.902	1070.464	1066.46	833265
03-04-2020	1119.348	1120.933	1047.036	1068.037	1064.042	1487426
07-04-2020	1095.574	1202.259	1090.622	1195.028	1190.558	1258845
08-04-2020	1163.924	1217.366	1150.898	1180.912	1176.496	1082580
09-04-2020	1199.783	1221.13	1182.002	1207.707	1203.19	630725
13-04-2020	1196.613	1203.547	1168.877	1178.04	1173.634	450162
15-04-2020	1189.679	1224.3	1131.929	1139.209	1134.948	842092
16-04-2020	1135.792	1176.158	1135.792	1157.683	1153.354	597698
17-04-2020	1208.5	1218.307	1181.457	1213.502	1208.964	564636

1.2 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. Few conventional methods that are practiced in order to remove redundancy are: Remove null values, Delete duplicate valued data. Figure 1 and Figure 2 shows the code for which Null values are checked in the ICICI datasets. Figure 3 and 4 shows the code for which Null values are checked in the Reliance Industries datasets. Upon checking, there were few NaN values (Null Values) in both datasets. In order to clean the dataset, dropna() function is called by the data-frame object. This function drops the specific row or column which contain the null value

```
In [4]: #CHECKING NULL VALUES IN ic_6mSET
        ic_6m.isnull().sum()
Out[4]: Date         0
        Open         1
        High         1
        Low          1
        Close        1
        Adj Close    1
        Volume       1
        dtype: int64

In [5]: ic_6m[ic_6m.isna().any(axis=1)]
Out[5]:
          Date        Open  High  Low  Close  Adj Close  Volume
    31    2020-11-14  NaN   NaN   NaN  NaN    NaN        NaN

In [6]: #SINCE THE ic_6mSET HAD NULL VALUES, WE REMOVE IT USING DROPNA()
        ic_6m1=ic_6m.dropna()
        ic_6m1
```

Fig -1: Identifying Null Values from ICICI Bank 6-month dataset and handling them

```
In [4]: #CHECKING NULL VALUES IN ic_1yrSET
        ic_1yr.isnull().sum()
Out[4]: Date         0
        Open         1
        High         1
        Low          1
        Close        1
        Adj Close    1
        Volume       1
        dtype: int64

In [5]: ic_1yr[ic_1yr.isna().any(axis=1)]
Out[5]:
          Date        Open  High  Low  Close  Adj Close  Volume
    157   2020-11-14  NaN   NaN   NaN  NaN    NaN        NaN

In [6]: #SINCE THE ic_1yrSET HAD NULL VALUES, WE REMOVE IT USING DROPNA()
        ic_1yr1=ic_1yr.dropna()
        ic_1yr1
```

Fig -2: Identifying Null Values from ICICI Bank 1 year dataset and handling them

```
In [4]: #CHECKING NULL VALUES IN rel_6mSET
        rel_6m.isnull().sum()
Out[4]: Date         0
        Open         1
        High         1
        Low          1
        Close        1
        Adj Close    1
        Volume       1
        dtype: int64

In [5]: rel_6m[rel_6m.isna().any(axis=1)]
Out[5]:
          Date        Open  High  Low  Close  Adj Close  Volume
    31    2020-11-14  NaN   NaN   NaN  NaN    NaN        NaN

In [6]: #SINCE THE rel_6mSET HAD NULL VALUES, WE REMOVE IT USING DROPNA()
        rel_6m1=rel_6m.dropna()
        rel_6m1
```

Fig -3: Identifying Null Values from Reliance Industries 6-month dataset and handling them

```
In [4]: #CHECKING NULL VALUES IN rel_1yrSET
        rel_1yr.isnull().sum()
Out[4]: Date         0
        Open         1
        High         1
        Low          1
        Close        1
        Adj Close    1
        Volume       1
        dtype: int64

In [5]: rel_1yr[rel_1yr.isna().any(axis=1)]
Out[5]:
          Date        Open  High  Low  Close  Adj Close  Volume
    157   2020-11-14  NaN   NaN   NaN  NaN    NaN        NaN

In [6]: #SINCE THE rel_1yrSET HAD NULL VALUES, WE REMOVE IT USING DROPNA()
        rell_1yr=rel_1yr.dropna()
        rell_1yr
```

Fig -4: Identifying Null Values from Reliance Industries 1 year dataset and handling them

The next step is to make sure that the datatype of dataset aligns with the compatibility of ARIMA model. To manipulate the dates, which is the predictor variable, the datatype of 'Date' attribute in the datasets should be of dateTime. There were no issues with the datatype of 'Date' in the ICICI Bank datasets, but, in the Reliance datasets, the datatype is String. Thus, to do that, the 'Date' attribute is converted into dateTime from String with the help of to_datetime() function. Figure 5 and Figure 6 shows the code snippet for the same.

```
In [12]: type(rel_6m1.Date[0])
Out[12]: str

In [14]: rel_6m['Date'] = pd.to_datetime(rel_6m['Date'], format='%Y-%m-%d')

In [16]: type(rel_6m.Date[0])
Out[16]: pandas._libs.tslibs.timestamps.Timestamp
```

Fig -5: Conversion of 'Date' Attribute to dateTime for Reliance Industries 6-month dataset

```
In [12]: type(rell_1yr.Date[0])
Out[12]: str

In [13]: rell_1yr['Date'] = pd.to_datetime(rell_1yr['Date'], format='%Y-%m-%d')

In [14]: type(rell_1yr.Date[0])
Out[14]: pandas._libs.tslibs.timestamps.Timestamp
```

Fig -6: Conversion of 'Date' Attribute to dateTime for Reliance Industries 1 year dataset

To finish off the Exploratory Data Analysis, groupby() function is called by the data-frame objects of ICICI bank and Reliance Industries. This function is used to group only the relevant attributes from the dataset, that is, 'Date' and 'Close'. The result of this ensures that irrelevant attributes from the datasets do not take part in the forecasting models, thereby not disrupting the accuracy of ARIMA model. Figure 7 and figure 8 shows the code snippets for ICICI Bank. Figure 9 and figure 10 shows the code snippets for Reliance Industries.

```
In [10]: ic_6m2 = ic_6m1.groupby('Date')[['Close']].mean()

In [11]: ic_6m2
Out[11]:
```

Date	Close
2020-10-01	369.149994
2020-10-05	373.149994
2020-10-06	380.450012
2020-10-07	382.950012
2020-10-08	387.549988
...	...
2021-03-23	586.299988
2021-03-24	567.450012
2021-03-25	571.400024
2021-03-26	578.549988
2021-03-30	591.349976

122 rows × 1 columns

Fig -7: Groupby() function to group 'Date' and 'Close' attribute for ICICI Bank 6-month dataset.

```
In [10]: ic_1yr2 = ic_1yr1.groupby('Date')[['Close']].mean()

In [11]: ic_1yr2
Out[11]:
```

Date	Close
2020-03-31	324.500000
2020-04-01	311.450012
2020-04-03	286.500000
2020-04-07	326.100006
2020-04-08	319.000000
...	...
2021-03-23	586.299988
2021-03-24	567.450012
2021-03-25	571.400024
2021-03-26	578.549988
2021-03-30	591.349976

248 rows × 1 columns

Fig -8: Groupby() function to group 'Date' and 'Close' attribute for ICICI Bank 1 year dataset.

```
In [10]: rel_6m2 = rel_6m1.groupby('Date')[['Close']].mean()

In [11]: rel_6m2
Out[11]:
```

Date	Close
2020-10-01	2225.050049
2020-10-05	2211.149902
2020-10-06	2210.149902
2020-10-07	2257.149902
2020-10-08	2238.899902
...	...
2021-03-23	2089.050049
2021-03-24	2047.300049
2021-03-25	1992.750000
2021-03-26	1994.250000
2021-03-30	2028.599976

122 rows × 1 columns

Fig- 9: Groupby() function to group 'Date' and 'Close' attribute for Reliance Industries 6-month dataset.

```
In [10]: relll_1yr = rell_1yr.groupby('Date')[['Close']].mean()

In [11]: relll_1yr
Out[11]:
```

Date	Close
2020-03-31	1101.963745
2020-04-01	1070.463501
2020-04-03	1068.036621
2020-04-07	1195.028076
2020-04-08	1180.912476
...	...
2021-03-23	2089.050049
2021-03-24	2047.300049
2021-03-25	1992.750000
2021-03-26	1994.250000
2021-03-30	2028.599976

248 rows × 1 columns

Fig- 10: Groupby() function to group 'Date' and 'Close' attribute for Reliance Industries 6-month dataset.

1.3 Stationarity test

The next phase in the process is to check the stationarity of data. A data is said to be stationary if the mean, variance and autocorrelation structure do not show any difference over time. In other words, the data should not contain any trends or seasonality and has to show a constant variance and autocorrelation structure over time [5].

Figure 11 and 12 shows the Autocorrelation and Partial Autocorrelation functions (ACF and PACF) for ICICI Bank. Figure 13 and 14 shows the Autocorrelation and Partial Autocorrelation functions for Reliance Industries. On observing the ACF and PACF plots we conclude that both of these series are not stationary. To move forward, number of lags are to be calculated.

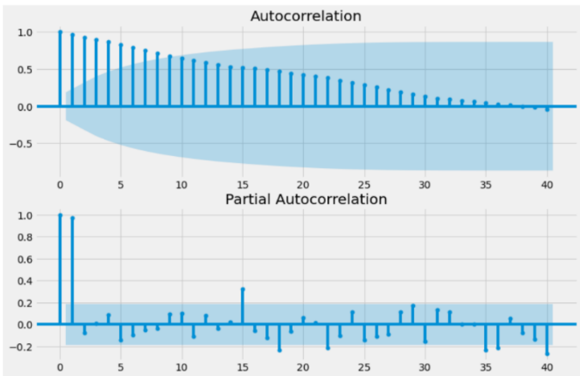

Fig- 11: ACF and PACF plots for ICICI Bank 6-month dataset

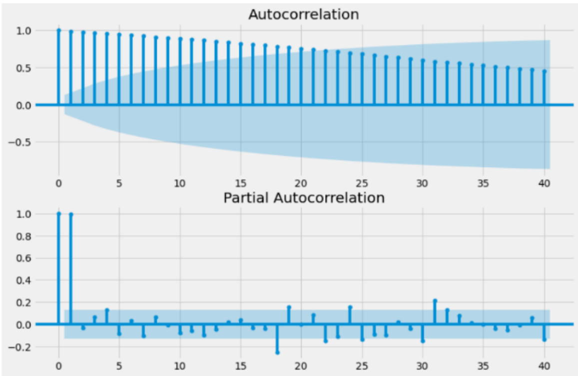

Fig- 12: ACF and PACF plots for ICICI Bank 1 year dataset

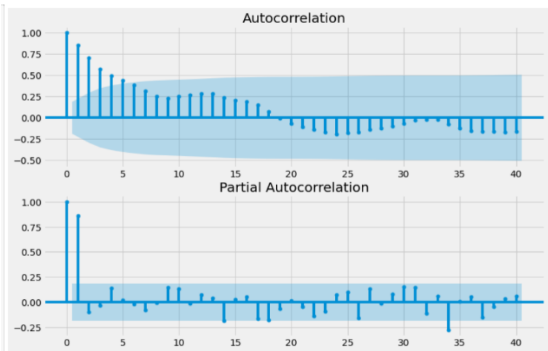

Fig- 13: ACF and PACF plots for Reliance Industries 6-month dataset

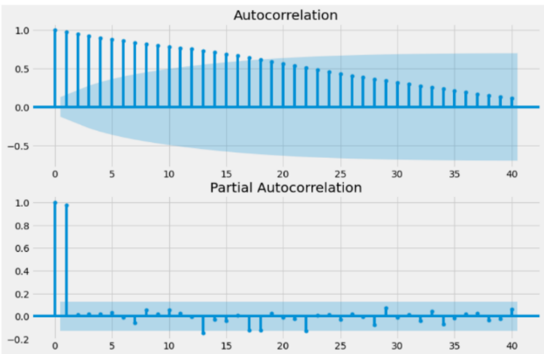

Fig- 14: ACF and PACF plots for Reliance Industries 1 year dataset

To further support the findings about the stationarity, Dickey Fuller Test was performed [7], which is based on Null Hypothesis. The assumed Null-Hypothesis is 'Dataset is not Stationary'. For a dataset to be stationary, p-value must be less than 5%. But in this case, the p-value for both Reliance Industries (6 month and 1 year) and ICICI Bank (6 month and 1 year) is more than 5 %. Hence, assumed Null hypothesis was true. Figure 15 and 16 shows the code snippet for ICICI Bank datasets. Figure 17 and 18 shows the code snippet for Reliance Industries datasets.

```
In [27]: adfuller_test(ic_6m2['Close'])

ADF Test Statistic : -1.675555925767742
p-value : 0.44374875443855466
#Lags Used : 4
Number of Observations Used : 117
weak evidence against null hypothesis, time series has a unit root, indicating it is non-stationary
```

Fig- 15: Dicky Fuller Test for ICICI Bank 6-month dataset

```
adfuller_test(ic_1yr2['Close'])

ADF Test Statistic : -0.6464613524636841
p-value : 0.8601343682570478
#Lags Used : 0
Number of Observations Used : 247
weak evidence against null hypothesis, time series has a unit root, indicating it is non-stationary
```

Fig- 16: Dicky Fuller Test for ICICI Bank 1 year dataset

```
adfuller_test(rel_6m2['Close'])

ADF Test Statistic : -2.676180190111085
p-value : 0.07825161057258742
#Lags Used : 0
Number of Observations Used : 121
weak evidence against null hypothesis, time series has a unit root, indicating it is non-stationary
```

Fig- 17: Dicky Fuller Test for Reliance Industries 6-month dataset

```
adfuller_test(relll_1yr['Close'])

ADF Test Statistic : -2.8568871073566466
p-value : 0.05059604845210965
#Lags Used : 0
Number of Observations Used : 247
weak evidence against null hypothesis, time series has a unit root, indicating it is non-stationary
```

Fig- 18: Dicky Fuller Test for Reliance Industries 1 year dataset

To discard the Null Hypothesis of Non-stationarity, a technique called as differencing is adapted to the datasets. The number of differences done constitutes to the integrated difference of the ARIMA model. Figure 19 and Figure 20 shows the code snippet for ICICI Bank. Figure 21 and Figure 22 shows the code snippet Reliance industries.

```
ic_6m2['Close First Difference'] = ic_6m2['Close'] - ic_6m2['Close'].shift(1)
ic_6m2['Close'].shift(1)
```

```
Date
2020-10-01          NaN
2020-10-05   369.149994
2020-10-06   373.149994
2020-10-07   380.450012
2020-10-08   382.950012
                ...
2021-03-23   573.400024
2021-03-24   586.299988
2021-03-25   567.450012
2021-03-26   571.400024
2021-03-30   578.549988
Name: Close, Length: 122, dtype: float64
```

```
adfuller_test(ic_6m2['Close First Difference'].dropna())
ic_6m2['Close First Difference'].plot(figsize=(12,8))
plt.xlabel("Year")
plt.ylabel("Price")
```

```
ADF Test Statistic : -5.578169517183394
p-value : 1.4164283308196047e-06
#Lags Used : 3
Number of Observations Used : 117
strong evidence against the null hypothesis(Ho), reject the null hypothesis. ic_6m has no unit root and is stationary
```

Fig- 19: Differencing for ICICI Bank 6-month dataset

```
ic_1yr2['Close First Difference'] = ic_1yr2['Close'] - ic_1yr2['Close'].shift(1)
ic_1yr2['Close'].shift(1)
```

```
Date
2020-03-31          NaN
2020-04-01   324.500000
2020-04-03   311.450012
2020-04-07   286.500000
2020-04-08   326.100006
                ...
2021-03-23   573.400024
2021-03-24   586.299988
2021-03-25   567.450012
2021-03-26   571.400024
2021-03-30   578.549988
Name: Close, Length: 248, dtype: float64
```

```
adfuller_test(ic_1yr2['Close First Difference'].dropna())
ic_1yr2['Close First Difference'].plot(figsize=(12,8))
plt.xlabel("Year")
plt.ylabel("Price")
```

```
ADF Test Statistic : -15.826195186760655
p-value : 1.0066789426812739e-28
#Lags Used : 0
Number of Observations Used : 246
strong evidence against the null hypothesis(Ho), reject the null hypothesis. ic_1yr has no unit root and is stationary
```

Fig- 20: Differencing Test for ICICI Bank 1 year dataset

```
rel_6m2['Close First Difference'] = rel_6m2['Close'] - rel_6m2['Close'].shift(1)
rel_6m2['Close'].shift(1)
```

```
Date
2020-10-01           NaN
2020-10-05   2225.050049
2020-10-06   2211.149902
2020-10-07   2210.149902
2020-10-08   2257.149902
                 ...
2021-03-23   2061.850098
2021-03-24   2089.050049
2021-03-25   2047.300049
2021-03-26   1992.750000
2021-03-30   1994.250000
Name: Close, Length: 122, dtype: float64
```

```
adfuller_test(rel_6m2['Close First Difference'].dropna())
rel_6m2['Close First Difference'].plot(figsize=(12,8))
plt.xlabel("Year")
plt.ylabel("Price")
```

```
ADF Test Statistic : -10.425061256450515
p-value : 1.6660057704672985e-18
#Lags Used : 0
Number of Observations Used : 120
strong evidence against the null hypothesis(Ho), reject the null hypothesis. rel_6m has no unit root and is stationary
```

Fig- 21: Differencing Test for Reliance Industries 6-month dataset

```
relll_1yr['Close First Difference'] = relll_1yr['Close'] - relll_1yr['Close'].shift(1)
relll_1yr['Close'].shift(1)
```

```
Date
2020-03-31           NaN
2020-04-01   1101.963745
2020-04-03   1070.463501
2020-04-07   1068.036621
2020-04-08   1195.028076
                 ...
2021-03-23   2061.850098
2021-03-24   2089.050049
2021-03-25   2047.300049
2021-03-26   1992.750000
2021-03-30   1994.250000
Name: Close, Length: 248, dtype: float64
```

```
adfuller_test(relll_1yr['Close First Difference'].dropna())
relll_1yr['Close First Difference'].plot(figsize=(12,8))
plt.xlabel("Year")
plt.ylabel("Price")
```

```
ADF Test Statistic : -15.347206935312283
p-value : 3.761981818357672e-28
#Lags Used : 0
Number of Observations Used : 246
strong evidence against the null hypothesis(Ho), reject the null hypothesis. rel_1yr has no unit root and is stationary
```

Fig- 22: Differencing Test for Reliance Industries 1 year dataset

After differencing is performed, a graph is plotted to confirm the stationarity of the datasets. For the graph, it can be concluded that since the intervals are regular, differencing worked and datasets are now stationary. Figure 23 and 24 depict the visualization for ICICI Bank. Figure 25 and 26 depict the visualization for Reliance industries.

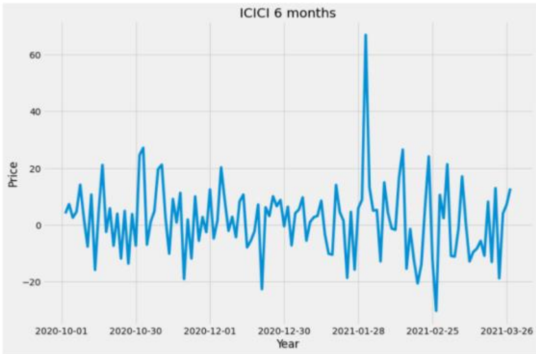

Fig- 23: Stationarity Visualization for ICICI Bank 6-month dataset

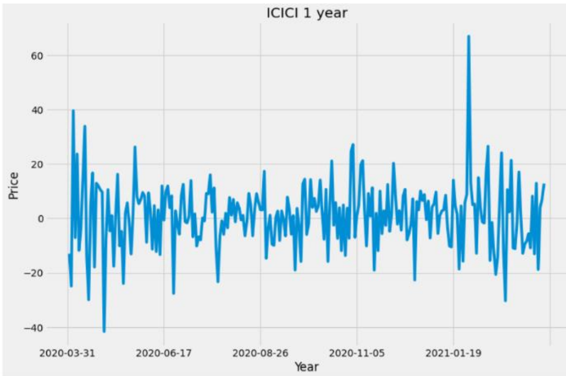

Fig- 24: Stationarity Visualization for ICICI Bank 1 year dataset

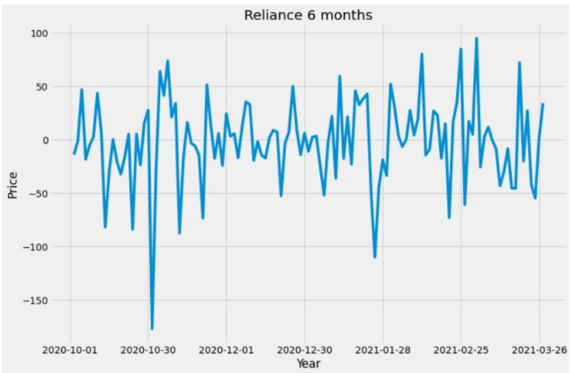

Fig- 25: Stationarity Visualization for Reliance Industries 6-month dataset

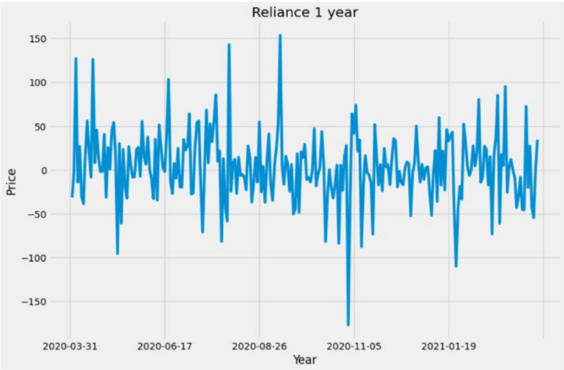

Fig- 26: Stationarity Visualization for Reliance Industries 1 year dataset

1.4 Auto ARIMA

Now that the datasets for both ICICI Bank and Reliance industries have become stationary, we proceed to the next phase. The next phase involves finding the optimum values of p,d and q for the ARIMA model. This is carried out by the auto_arima () function. This function calculates the suitable values of p, d and q for the best results of forecasting for the ARIMA model. The optimum model for forecasting is the model whose AIC value is the lowest. After computing the optimum model values using auto_arima (), we find that ARIMA (0, 2, 1) (0, 0, 0) [0] is the best model for the 1 year dataset and ARIMA (1, 0, 0) (0, 0, 0) [0] is the best model for 6 months dataset of Reliance Industries. Figure 27 and 28 shows the code snippet for Reliance Industries (6 month and 1 year) dataset respectively. While the ARIMA(0,1,0)(0,0,0)[0] is the best model for ICICI Bank dataset(1 year and 6 month) and Reliance Industries(6 month) dataset. Figure 29 and 30 shows the code snippet for ICICI Bank (6 month and 1 year) dataset respectively.

```
stepwise_fit = auto_arima(rel_6m2['Close'], trace=True,suppress_warnings=True)

Performing stepwise search to minimize aic
 ARIMA(2,0,2)(0,0,0)[0] intercept   : AIC=1253.252, Time=0.67 sec
 ARIMA(0,0,0)(0,0,0)[0] intercept   : AIC=1472.967, Time=0.03 sec
 ARIMA(1,0,0)(0,0,0)[0] intercept   : AIC=1248.435, Time=0.13 sec
 ARIMA(0,0,1)(0,0,0)[0] intercept   : AIC=1367.784, Time=0.30 sec
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=2206.751, Time=0.01 sec
 ARIMA(2,0,0)(0,0,0)[0] intercept   : AIC=1249.311, Time=0.79 sec
 ARIMA(1,0,1)(0,0,0)[0] intercept   : AIC=1249.394, Time=0.25 sec
 ARIMA(2,0,1)(0,0,0)[0] intercept   : AIC=1251.275, Time=0.73 sec
 ARIMA(1,0,0)(0,0,0)[0]             : AIC=inf, Time=0.06 sec

Best model:  ARIMA(1,0,0)(0,0,0)[0] intercept
Total fit time: 3.008 seconds
```

Fig- 27: Computing Optimum values of p,d and q using auto_arima() function for Reliance Industries 6-month dataset.

```
stepwise_fit = auto_arima(relll_1yr['Close'], trace=True,suppress_warnings=True)

Performing stepwise search to minimize aic
 ARIMA(2,2,2)(0,0,0)[0]             : AIC=inf, Time=1.58 sec
 ARIMA(0,2,0)(0,0,0)[0]             : AIC=2687.578, Time=0.03 sec
 ARIMA(1,2,0)(0,0,0)[0]             : AIC=2626.838, Time=0.13 sec
 ARIMA(0,2,1)(0,0,0)[0]             : AIC=2528.586, Time=0.15 sec
 ARIMA(1,2,1)(0,0,0)[0]             : AIC=2530.563, Time=0.33 sec
 ARIMA(0,2,2)(0,0,0)[0]             : AIC=2530.561, Time=0.33 sec
 ARIMA(1,2,2)(0,0,0)[0]             : AIC=inf, Time=0.85 sec
 ARIMA(0,2,1)(0,0,0)[0] intercept   : AIC=inf, Time=0.51 sec

Best model:  ARIMA(0,2,1)(0,0,0)[0]
Total fit time: 3.941 seconds
```

Fig- 28: Computing Optimum values of p,d and q using auto_arima() function for Reliance Industries 1 year dataset.

```
stepwise_fit = auto_arima(ic_6m2['Close'], trace=True,suppress_warnings=True)

Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=964.326, Time=0.41 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=961.050, Time=0.01 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=962.912, Time=0.06 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=962.911, Time=0.11 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=961.582, Time=0.01 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=964.510, Time=0.26 sec

Best model:  ARIMA(0,1,0)(0,0,0)[0] intercept
Total fit time: 0.898 seconds
```

Fig- 29: Computing Optimum values of p,d and q using auto_arima() function for ICICI Bank 6-month dataset.

```
stepwise_fit = auto_arima(ic_1yr2['Close'], trace=True,suppress_warnings=True)

Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=inf, Time=2.50 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=1943.341, Time=0.05 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=1945.304, Time=0.31 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=1945.298, Time=0.34 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=1943.250, Time=0.05 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=1943.829, Time=0.90 sec

Best model:  ARIMA(0,1,0)(0,0,0)[0]
Total fit time: 4.165 seconds
```

Fig- 30: Computing Optimum values of p,d and q using auto_arima() function for ICICI Bank 1 year dataset.

1.5 ARIMA Model

The training and testing of the ARIMA model is the next phase after finding the optimum p, d and q values of ARIMA model. The training and testing data is split in a ratio of 70:30, where in the 70% of data is trained and remaining 30% of the data is used for testing in the model. The model is fitted and a model_prediction object is created for further forecasting process. After the implementation of ARIMA model, with the optimum values, the prediction values are depicted using a visualization plot, with the Actual price(Red in colour) and Predicted price(Blue in colour) of the stock. Figure 31 and 32 depicts the visualization for ICICI Bank (6 month and 1 year) dataset respectively. Figure 33 and 34 depicts the visualization for Reliance Industries (6 month and 1 year) dataset respectively.

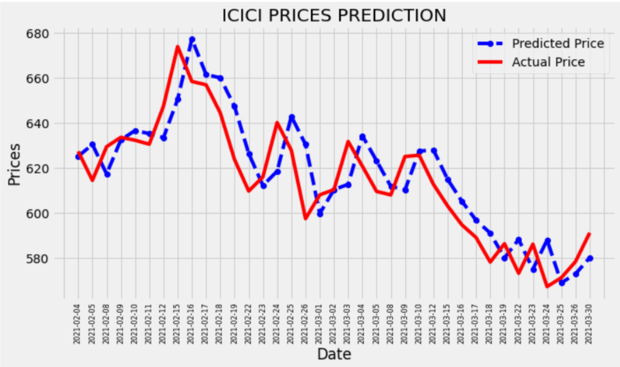

Fig- 31: Forecasting of stock price of ICICI Bank using 6-month dataset.

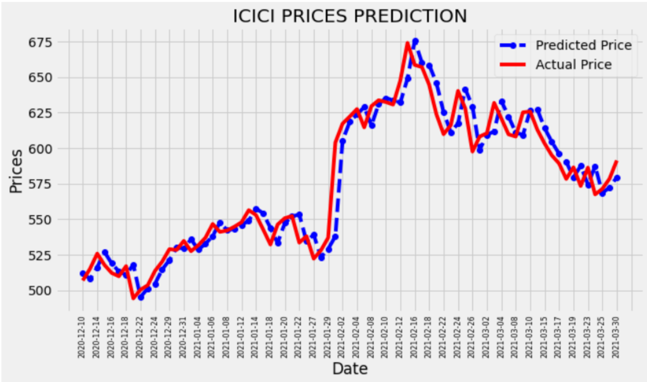

Fig- 32: Forecasting of stock price of ICICI Bank using 1 year dataset.

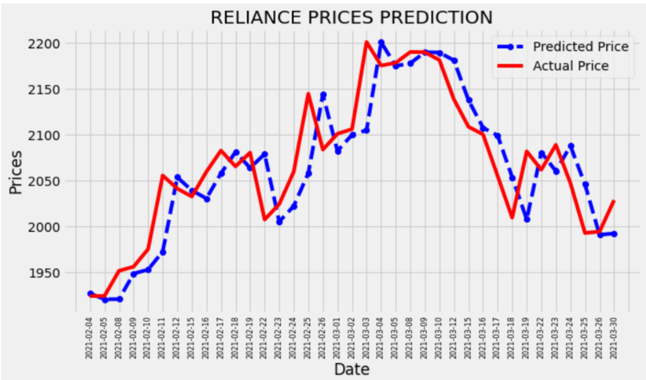

Fig- 33: Forecasting of stock price of Reliance Industries using 6-month dataset.

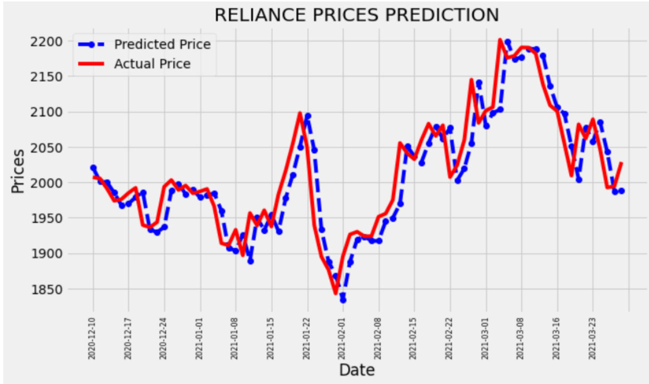

Fig- 34: Forecasting of stock price of Reliance Industries using 1 year dataset.

1.6 Future Price Prediction

Now that the ARIMA model is trained and tested for future predictions, the next process is to predict the stock prices of the companies for the next 30 days using the model_fit.predict () function with the parameters involving the start value of length of the present dataset and end value with 30 day increment from the final length value of the dataset. Figure 35 and 36 shows the future prediction values for ICICI Bank using the 6 month and 1 year dataset respectively. Figure 37 and 38 shows the future prediction values for Reliance Industries using the 6 month and 1 year dataset respectively.

```
pred=model_fit.predict(start=len(ic_6m2),end=len(ic_6m2)+30,typ='levels')
print(pred)
```

```
[580.29498795 582.0399879  583.78498785 585.5299878  587.27498775
 589.0199877  590.76498765 592.5099876  594.25498755 595.9999875
 597.74498745 599.4899874  601.23498735 602.9799873  604.72498725
 606.4699872  608.21498715 609.9599871  611.70498705 613.449987
 615.19498695 616.9399869  618.68498685 620.4299868  622.17498675
 623.9199867  625.66498665 627.4099866  629.15498655 630.8999865
 632.64498645 634.3899864 ]
```

Fig- 35: Future Prediction of stock prices for ICICI Bank using 6-month dataset.

```
pred=model_fit.predict(start=len(ic_1yr2),end=len(ic_1yr2)+30,typ='levels')
print(pred)
```

```
[579.58271153 580.61543506 581.64815859 582.68088211 583.71360564
 584.74632917 585.7790527  586.81177623 587.84449976 588.87722328
 589.90994681 590.94267034 591.97539387 593.0081174  594.04084093
 595.07356446 596.10628798 597.13901151 598.17173504 599.20445857
 600.2371821  601.26990563 602.30262915 603.33535268 604.36807621
 605.40079974 606.43352327 607.4662468  608.49897033 609.53169385
 610.56441738 611.59714091]
```

Fig- 36: Future Prediction of stock prices for ICICI Bank using 1 year dataset.

```
pred=model_fit.predict(start=len(rel_6m2),end=len(rel_6m2)+30,typ='levels')
print(pred)
```

```
[1992.32666626 1990.40333252 1988.47999877 1986.55666503 1984.63333129
 1982.70999755 1980.78666381 1978.86333007 1976.93999632 1975.01666258
 1973.09332884 1971.1699951  1969.24666136 1967.32332762 1965.39999387
 1963.47666013 1961.55332639 1959.62999265 1957.70665891 1955.78332517
 1953.85999142 1951.93665768 1950.01332394 1948.0899902  1946.16665646
 1944.24332272 1942.31998897 1940.39665523 1938.47332149 1936.54998775
 1934.62665401 1932.70332027]
```

Fig- 37: Future Prediction of stock prices for Reliance Industries using 6-month dataset.

```
pred=model_fit.predict(start=len(relll_1yr),end=len(relll_1yr)+30,typ='levels')
print(pred)
```

```
[1989.05807528 1983.79495649 1978.46064362 1973.05513668 1967.57843566
 1962.03054057 1956.4114514  1950.72116816 1944.95969084 1939.12701945
 1933.22315398 1927.24809443 1921.20184081 1915.08439311 1908.89575134
 1902.63591549 1896.30488557 1889.90266158 1883.4292435  1876.88463135
 1870.26882513 1863.58182483 1856.82363046 1849.99424201 1843.09365948
 1836.12188288 1829.07891221 1821.96474745 1814.77938863 1807.52283573
 1800.19508875 1792.7961477 ]
```

Fig- 38: Future Prediction of stock prices for Reliance Industries using 1 year dataset.

1.7 Model Performance

The final step in the forecasting process is to check the model performance. This can be computed using various evaluation techniques such as Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and the Mean Absolute Percentage Error (MAPE). The conclusions using the above-mentioned techniques are shown in Figures 39-42.

```
# model performance
mse = mean_squared_error(test_ic_6m, model_predictions)
print('MSE: '+str(mse))
mae = mean_absolute_error(test_ic_6m,model_predictions)
print('MAE: '+str(mae))
rmse = math.sqrt(mean_squared_error(test_ic_6m, model_predictions))
print('RMSE: '+str(rmse))
mape = np.mean(np.abs((model_predictions - test_ic_6m))/np.abs(test_ic_6m))
print('MAPE: '+str(mape))
```

```
MSE: 193.13385116212487
MAE: 11.761947243706198
RMSE: 13.897260563223417
MAPE: 0.047784582504879644
```

Fig- 39: Model performance of 6-month dataset of ICICI bank.

```
# model performance
mse = mean_squared_error(test_ic_1yr, model_predictions)
print('MSE: '+str(mse))
mae = mean_absolute_error(test_ic_1yr,model_predictions)
print('MAE: '+str(mae))
rmse = math.sqrt(mean_squared_error(test_ic_1yr, model_predictions))
print('RMSE: '+str(rmse))
mape = np.mean(np.abs(model_predictions - test_ic_1yr)/np.abs(test_ic_1yr))
print('MAPE: '+str(mape))
```

```
MSE: 187.38243166485807
MAE: 10.081561186646116
RMSE: 13.688770275845018
MAPE: 0.09521617913038301
```

Fig- 40: Model performance of 1 year dataset of ICICI bank.

```
# model performance
mse = mean_squared_error(test_rel_6m, model_predictions)
print('MSE: '+str(mse))
mae = mean_absolute_error(test_rel_6m,model_predictions)
print('MAE: '+str(mae))
rmse = math.sqrt(mean_squared_error(test_rel_6m, model_predictions))
print('RMSE: '+str(rmse))
mape = np.mean(np.abs(model_predictions - test_rel_6m)/np.abs(test_rel_6m))
print('MAPE: '+str(mape))
```

```
MSE: 1576.643300682668
MAE: 30.386258200085297
RMSE: 39.706967910968324
MAPE: 0.041785657266990656
```

Fig- 41: Model performance of 6-month dataset of Reliance Industries

```
# model performance
mse = mean_squared_error(test_rel_1yr, model_predictions)
print('MSE: '+str(mse))
mae = mean_absolute_error(test_rel_1yr,model_predictions)
print('MAE: '+str(mae))
rmse = math.sqrt(mean_squared_error(test_rel_1yr, model_predictions))
print('RMSE: '+str(rmse))
mape = np.mean(np.abs(model_predictions - test_rel_1yr)/np.abs(test_rel_1yr))
print('MAPE: '+str(mape))
```

```
MSE: 1411.2140989549916
MAE: 28.84399351209135
RMSE: 37.566129677609744
MAPE: 0.04608703539200299
```

Fig- 42: Model performance of 1 year dataset of Reliance Industries

2. CONCLUSION

The volatile nature of stock prices makes them difficult to predict. The experimental analysis in this research work suggests that a forecasting model specifically the ARIMA model can be used effectively with a reasonably high accuracy in predicting the future stock prices. The specific instances of ICICI Bank and Reliance Industries have been used for verifying the hypothesis. The only drawback of this analysis is that ARIMA model holds higher accuracy for short-term predictions.

REFERENCES

[1] SHEIKH MOHAMMAD IDREES, M. AFSHAR ALAM, PARUL AGARWAL," A Prediction Approach for Stock Market Volatility Based on Time Series Data", IEEE Access, vol. 7, pp. 17287-17298, 2019.

[2] Wint Nyein Chan," Time Series Data Mining: Comparative Study of ARIMA and Prophet Methods for Forecasting Closing Prices of Myanmar Stock Exchange", Journal of Computer Applications and Research, vol. 1, pp.75-80, 2020.R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[3] Reliance Industries Historical Stock Data, https://finance.yahoo.com/quote/RELIANCE.BO?p=RELIANCE.BO&.tsrc=fin-srch , [Online; Accessed March 2020]

[4] ICICI Bank Historical Stock Data, https://finance.yahoo.com/quote/ICICIBANK.BO/history?p=ICICIBANK.BO , [Online; Accessed March 2020]

[5] Closing Price, https://www.investor.gov/introduction-investing/investing-basics/glossary/closing-price , [Online; Accessed March 2020]

[6] A. A. Ariyo, A. O. Adewumi and C. K. Ayo, "Stock Price Prediction Using the ARIMA Model," 2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation, Cambridge, UK, 2014, pp. 106-112, doi: 10.1109/UKSim.2014.67.

[7] Leybourne, Stephen, Tae-Hwan Kim, and Paul Newbold. "Examination of some more powerful modifications of the Dickey–Fuller test." Journal of Time Series Analysis 26.3 (2005): 355-369.

BIOGRAPHIES

Aravind Ganesan.
Graduated with a bachelor’s degree from SCSVMV. Currently pursuing master’s degree at University of Texas at Dallas.

Adarsh Kannan Iyengar.
Graduated with a bachelor’s degree from SCSVMV. Currently pursuing master’s degree at Dalhousie University.