A CASE STUDY ON STOCK MARKET FORECASTING BY USING ARIMA METHOD AND CHECKING THE VOLATILITY OF THE DATA BY FITTING ARCH-GARCH MODEL

By

ARKAPRAVA BANERJEE

REGN NO. – 100063 of 2020-2021

ROLL NO. – 96/STS NO. 200002

M.Sc. Final Semester Dissertation

DEPARTMENT OF STATISTICS

2025

Declaration

I hereby declare that this dissertation entitled **"A case study on Stock Market Forecasting by using ARIMA method and checking the volatility of the data by fitting ARCH-GARCH model"** has been prepared by me at the University Of Kalyani, Nadia 741235, under the guidance of **Dr. Chiranjib Neogi** ***(Guest Faculty, Department of Statistics, University of Kalyani and Retired Associate Scientist, Economic Research Unit, Indian Statistical Institute)***, in the partial fulfillment of the requirement for the award of degree of 5-Year Integrated M.Sc. in Statistics.

This dissertation has not been submitted in part or full to any other university or towards any other degree before this below mentioned date.

Date : ARKAPRAVA BANERJEE

Certificate

It is certified that Arkaprava Banerjee has carried out the project work presented in this dissertation entitled "A Case Study on Stock Market forecasting by using ARIMA method and checking the volatility of the data using ARCH-GARCH model" for the award of Master of Science in Statistics from University of Kalyani, Nadia-741235, India. It is further certified that he has completed the project work under the supervision of Dr. Chiranjib Neogi. This work done in this project is the result of candidate's own effort and has not been submitted elsewhere for the award of degree/diploma in any other Institute/University. The project work has been carried out from June 2025 to August 2025.

_____

Dr. Chandranath Pal (Head of the Department)

_____

Dr. Chiranjib Neogi (Supervisor)

Acknowledgement

We would like to express our sincere gratitude to our project supervisor Dr. Chiranjib Neogi for his patience, guidance, and support. We have benefited greatly from his wealth of knowledge and meticulous efforts in assistance he has provided. We are extremely grateful that he took us on for the project and continued to have faith in us. His encouraging words and thoughtful, detailed feedback have been very important to us.

Thank you to the Head of our Department, Dr. Chandranath Pal for giving us the opportunity to work on this project. Working on this project helped us to gain hands-on experience in whatever knowledge we have gained during this course.

Last but not the least thank you to each team member for working hard and being supportive and cooperative throughout the project.

Abstract

A correct prediction of stocks or Index can lead to huge profits for the seller and the broker. Frequently, it is brought out that prediction is chaotic rather than random, which means it can be predicted by carefully analyzing the history of respective stock market. In Stock Market Prediction, the aim is to predict the future value of the financial stocks of a company. We can predict the prices using ARIMA model of Time series analysis. Machine learning is an efficient way to represent such processes. It predicts a market value close to the tangible value, thereby increasing the accuracy. The recent trend in stock market prediction technologies is the use of machine learning which makes predictions based on the values of current stock market indices by training on their previous values. Machine learning itself employs different models to make prediction easier and authentic. The paper focuses on the use of ARIMA based Machine learning to predict stock values. Factors considered are Date, open, close, low and high.

Content

Introduction

The stock market is where investors buy and sell shares of companies. It's a set of exchanges where companies issue shares and other securities for trading. Companies list shares of their stock on an exchange through a process called an initial public offering, or IPO. Investors purchase those shares, which allow the company to raise money to grow its business. Investors can then buy and sell these stocks among themselves. The stock market is a place where anyone can buy and sell fractional ownership in a publicly traded company. It distributes control of some of the world's largest companies among hundreds of millions of individual investors and the buying and selling decisions of those investors determine the value of those companies.

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. All the companies enlisted in the stock exchange use to provide their information on a number of aspects of the company to inform the buyers and sellers of shares of the companies. Although, the stock trend is difficult to predict, investors seem to find new techniques in order to minimize the risk of investment and increase the probability of profiting from the investments. 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 analyze trends of stock prices with adequate information for decision making, it is proposed to use the ARIMA model for stock price prediction.

The ARCH-GARCH (Autoregressive Conditional Heteroskedasticity-Generalized ARCH) model is widely used in financial econometrics to forecast stock market volatility. Developed by Robert Engle (ARCH) and Tim Bollerslev (GARCH), it addresses the observation that financial time series, like stock returns, often exhibit time-varying volatility and clustering periods of high volatility followed by more high volatility.

The ARCH model assumes that current volatility depends on past squared errors (i.e. past shocks). However, it can be inefficient with many lags. To address this, the GARCH model generalizes ARCH by incorporating past forecasted variances, making it more parsimonious and flexible. In a typical GARCH (1,1) model, current volatility is a function of a long-term average, yesterday's squared return, and yesterday's variance.

Data Description

We have taken BANK NIFTY stock index daily data for June 2015-June 2025 from the official NSE (National Stock Exchange) website. The datasets have the following components: Date, Open, High, Low, Close. The picture displays the sample datasets for BANK NIFTY. In our analysis we have considered two 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 the dataset, 'Date' will be the predictor variable and 'Close' will be the target variable. We here describe the terminologies used in the data set:

BANK NIFTY- Bank Nifty, or the Nifty Bank Index, is a benchmark index on India's National Stock Exchange (NSE) that represents the performance of the 12 most liquid and large-cap banking stocks from the private and public sectors. It serves as a barometer for the banking sector's health and plays a key role in India's financial markets. The index is widely used for trading, especially in derivatives like futures and options, due to its high volatility and liquidity. Bank Nifty is rebalanced periodically and reflects investor sentiment, interest rate movements, and broader economic indicators affecting the banking industry.

Close Price: The final price at which the stock is traded on a given particular trading day. The close is a reference to the end of a trading session in the financial markets when the markets close for the day. The close can also refer to the process of exiting a trade or the final procedure in a financial transaction in which contract documents are signed and recorded.

Open price- It is the price at which the financial security opens in the market when trading begins. It may or may not be different from the previous day's closing price. The security may open at a higher price than the closing price due to excess demand of the security.

High- Today's high refers to a security's intraday highest trading price. It is represented by the highest point on a day's stock chart.

Low price- Low is a security's intraday low trading price. It is the lowest price at which a stock trades over the course of a trading day.

```
df
```

	Index Name	Date	Open	High	Low	Close
0	NIFTY BANK	16 Jun 2025	55554.10	55999.85	55381.45	55944.90
1	NIFTY BANK	13 Jun 2025	55149.30	55688.00	55149.30	55527.35
2	NIFTY BANK	12 Jun 2025	56480.90	56611.05	55968.50	56082.55
3	NIFTY BANK	11 Jun 2025	56639.85	56782.50	56399.40	56459.75
4	NIFTY BANK	10 Jun 2025	56993.90	57015.40	56564.15	56629.10
...	...	...	...	...	...	...
2473	NIFTY BANK	22 Jun 2015	17956.55	18420.95	17950.60	18334.50
2474	NIFTY BANK	19 Jun 2015	17758.95	17935.30	17730.70	17880.85
2475	NIFTY BANK	18 Jun 2015	17661.90	17797.35	17630.65	17733.55
2476	NIFTY BANK	17 Jun 2015	17681.20	17692.55	17533.45	17584.05
2477	NIFTY BANK	16 Jun 2015	17368.45	17651.90	17293.45	17602.45

```
[2478 rows x 6 columns]
```

The entire data

Pre-Processing the data

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, etc.

First we have to check for null values (NaN) in the dataset using the function isnull(). sum(). If null values are present then they have to be dropped using dropna() function. We use this on our data set and get the result as-No null values are present in the dataset.

The picture below shows the absence of null values in the data.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2478 entries, 0 to 2477
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Index Name  2478 non-null   object
 1   Date        2478 non-null   object
 2   Open        2478 non-null   float64
 3   High        2478 non-null   float64
 4   Low         2478 non-null   float64
 5   Close       2478 non-null   float64
dtypes: float64(4), object(2)
memory usage: 116.3+ KB

df.isnull().sum()

Index Name    0
Date          0
Open          0
High          0
Low           0
Close         0
dtype: int64
```

There are no missing values in the data set

Next, we have 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. The issue with the datatype of

'Date' in the dataset is that the datatype is String. Thus, to do that, the 'Date' attribute is converted into dateTime from String with the help of to_datetime() function.

Further we convert the dates to index formats for our ease of use. The data on BANK NIFTY is daily data but it has gaps since the market is closed on weekends and public holidays. So by converting it to index we can predict values for a certain number of days based on the previous data which is independent of date.

```
df['Date'] = pd.to_datetime(df['Date'],format='%Y-%m-%d')

df.set_index('Date',inplace=True)
df['Date']=df.index

df.dtypes

Index Name              object
Open                   float64
High                   float64
Low                    float64
Close                  float64
Date          datetime64[ns]
dtype: object

df.index

DatetimeIndex(['2025-06-16', '2025-06-13', '2025-06-12', '2025-06-11',
               '2025-06-10', '2025-06-09', '2025-06-06', '2025-06-05',
               '2025-06-04', '2025-06-03',
               ...
               '2015-06-29', '2015-06-26', '2015-06-25', '2015-06-24',
               '2015-06-23', '2015-06-22', '2015-06-19', '2015-06-18',
               '2015-06-17', '2015-06-16'],
              dtype='datetime64[ns]', name='Date', length=2478,
freq=None)
```

Code to change date to index

To finish off the pre-processing, df.drop() function is used on the data-frame. This function is used to drop the columns and keep relevant attributes from the dataset, that is, 'Index' and 'Close'.

```
df1=df.drop(axis=1,labels=['Index Name','Open','High','Low','Date'])
DF=df1.reset_index()
DF1=DF.drop(axis=1,labels=['Date'])
DF1

          Close
0      55944.90
1      55527.35
2      56082.55
3      56459.75
4      56629.10
...         ...
2473   18334.50
2474   17880.85
2475   17733.55
2476   17584.05
2477   17602.45

[2478 rows x 1 columns]
```

Data set with only Index and Close

Visualization of the data

Data Visualization is used to communicate information clearly and efficiently to users by the usage of information graphics such as tables and charts. It helps users in analyzing a large amount of data in a simpler way. It makes complex data more accessible, understandable, and usable.

Data visualization is actually a set of data points and information that are represented graphically to make it easy and quick for user to understand. Data visualization is good if it has a clear meaning, purpose, and is very easy to interpret, without requiring context. Tools of data visualization provide an accessible way to see and understand trends, outliers, and patterns in data by using visual effects or elements such as a chart, graphs, and maps.

Here we use two types of graphs for data visualization, one being the line charts which shows us the BANK NIFTY daily closing price against the time which shows all the details including everyday volatility and the other being box plot which shows the yearly close price graph with each year's volatility which makes our data easy to understand.

Line Chart

A line chart or line graph displays the evolution of one or several numeric variables. Data points are connected by straight line segments. It is similar to a scatter plot except that the measurement points are ordered (typically by their x-axis value) and joined with straight line segments. A line chart is often used to visualize a trend in data over intervals of time a time series thus the line is often drawn chronologically.

Line Graphs are used to display quantitative values over a continuous interval or time period. A Line Graph is most frequently used to show trends and analyze how the data has changed over time.

Line Graphs are drawn by first plotting data points on a Cartesian coordinate grid, then connecting a line between all of these points. Typically, the y-axis has a quantitative value, while the x-axis is a timescale or a sequence of intervals. Negative values can be displayed below the x-axis.

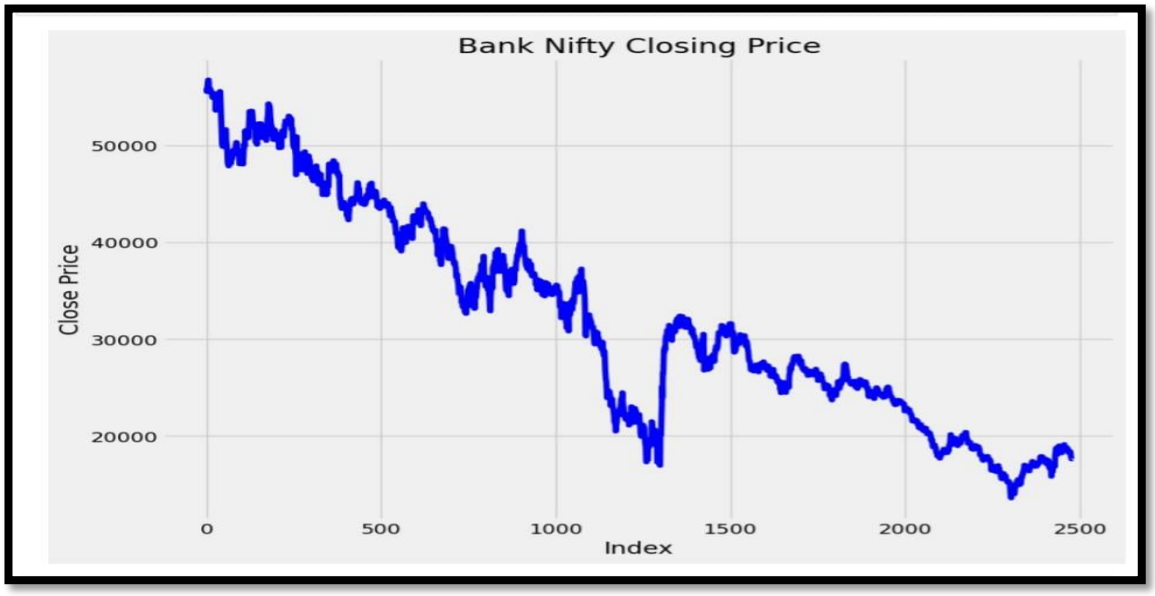
The direction of the lines on the graph works as a nice metaphor for the data: an upward slope indicates where values have increased and a downward slope indicates where values have decreased. The line's journey across the graph can create patterns that reveal trends in a dataset.

We plot the daily closing price of BANK NIFTY versus index (i.e 2478 days).

```
plt.figure(figsize=(10,8))
plt.grid(True)
plt.xlabel('Index')
plt.ylabel('Close Price')
```

```
plt.plot(DF1['Close'],color="Blue")
plt.title('Bank Nifty Closing Price')
plt.tick_params(axis='x',labelsize=14)
plt.tick_params(axis='y',labelsize=14)
plt.style.use('fivethirtyeight')

plt.show()
```

Code for plotting the close price vs the index



Line graph showing BANK NIFTY closing price vs the Index

Box Plot

A box plot visualization allows us to examine the distribution of data. One box plot appears for each attribute element. Each box plot displays the minimum, first quartile, median, third quartile, and maximum values. In addition, you can choose to display the mean and standard deviation as dashed lines. Outliers appear as points in the visualization. You can adjust the spacing between points (that is, jitter) to avoid overlap. A box plot must include at least one metric and at least one attribute.

Box Plot is the visual representation of the depicting groups of numerical data through their quartiles. Boxplot is also used for detecting the outlier in data set. It captures the summary of the data efficiently with a simple box and whiskers and allows us to compare easily across groups. Boxplot summarizes a sample data using 25th, 50th and 75th percentiles. These percentiles are also known as the lower quartile, median and upper quartile.
A box plot consists of 5 things.

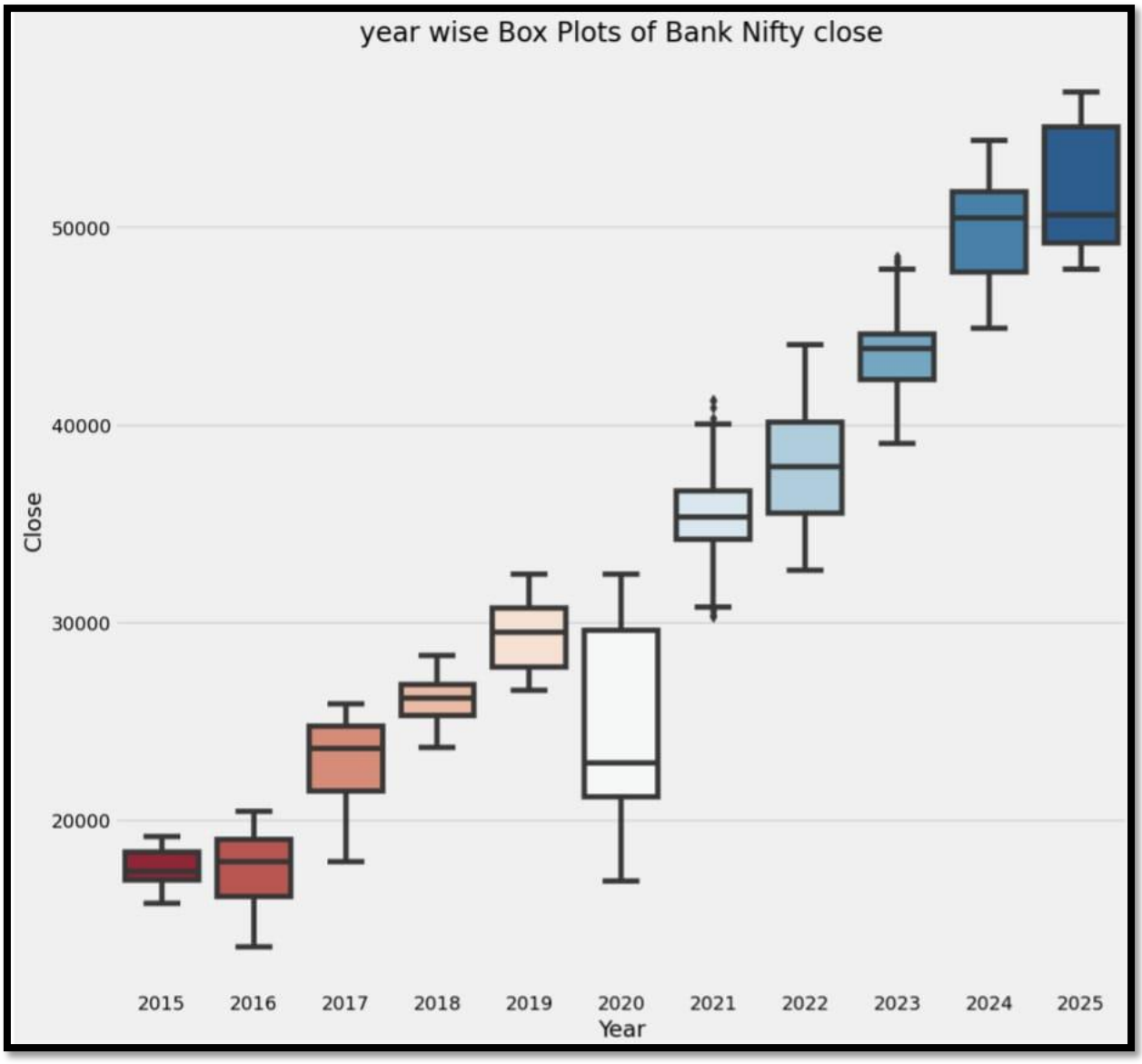
- Minimum
- First Quartile or 25%
- Median (Second Quartile) or 50%
- Third Quartile or 75%
- Maximum

Box plot showing the closing price :

```
#convert 'Date' to datetime and extract Year
monthly_data['Date'] = pd.to_datetime(monthly_data['Date'])
monthly_data['Year'] = monthly_data['Date'].dt.year

plt.figure(figsize=(12,12))
ax=sns.boxplot(x=monthly_data['Year'],y=monthly_data['Close'],palette=
'RdBu')
ax.set_title('year wise Box Plots of Bank Nifty close')
plt.tick_params(axis='x',labelsize=14)
plt.tick_params(axis='y',labelsize=14)
```

```
plt.style.context('fivethirtyeight')
plt.show()
```

Code for plotting box plot of year wise closing data



Box Plots of year wise BANK NIFTY closing price data

Here in the above year-wise Box Plots we can clearly summarize that the years 2020 and 2025 have very high volatility when compared to other years. In the above figure, we can see that the years 2015 and 2018 have the least volatility with the difference in the opening and the closing prices of the respective years being very less.

Stationarity Test

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 to show the next phase in the process is to check the stationarity of the data a constant variance and autocorrelation structure over time.

Strong stationarity

Strong stationarity requires the shift-invariance (in time) of the finite-dimensional distributions of a stochastic process. This means that the distribution of a finite sub-sequence of random variables of the stochastic process remains the same as we shift it along the time index axis. For example, all i.i.d. stochastic processes are stationary.

Formally, the discrete stochastic process $\boldsymbol{X}=\{x_i ; i\in\mathbb{Z}\}$ is stationary if

$$F_X(x_{t_{1+\tau}}, \ldots, x_{t_{n+\tau}}) = F_X(x_{t_1}, \ldots, x_{t_n})$$

for $T\subset\mathbb{Z}$ with $n\in\mathbb{N}$ and any $\tau\in\mathbb{Z}$. For continuous stochastic processes the condition is similar, with $T\subset\mathbb{R}$, $n\in\mathbb{N}$ and any $\tau\in\mathbb{R}$ instead.

This is the most common definition of stationarity, and it is commonly referred to simply as stationarity. It is sometimes also referred to as strict-sense stationarity or strong-sense stationarity.

Weak Stationary

Weak stationarity only requires the shift-invariance (in time) of the first moment and the cross moment (the auto-covariance). This means the process has the same mean at all time points, and that the covariance between the values at any two time

points, t and t−k, depend only on k, the difference between the two times, and not on the location of the points along the time axis.

Formally, the process $\{x_i ; \ i\in\mathbb{Z}\}$ is weakly stationary if:

1. The first moment of x_i is constant; i.e. $\forall t, E[x_i]=\mu$
2. The second moment of x_i is finite for all t; i.e. $\forall t, E[x_i^2]<\infty$ (which also implies of course $E[(x_i-\mu)^2]<\infty$; i.e. that variance is finite for all t)
3. The cross moment — i.e. the auto-covariance — depends only on the difference u-v; i.e. $\forall$ u,v,a; $cov(x_u, x_v)=cov(x_{u+a}, x_{v+a})$

The third condition implies that every lag $\tau\in\mathbb{N}$ has a constant covariance value associated with it:

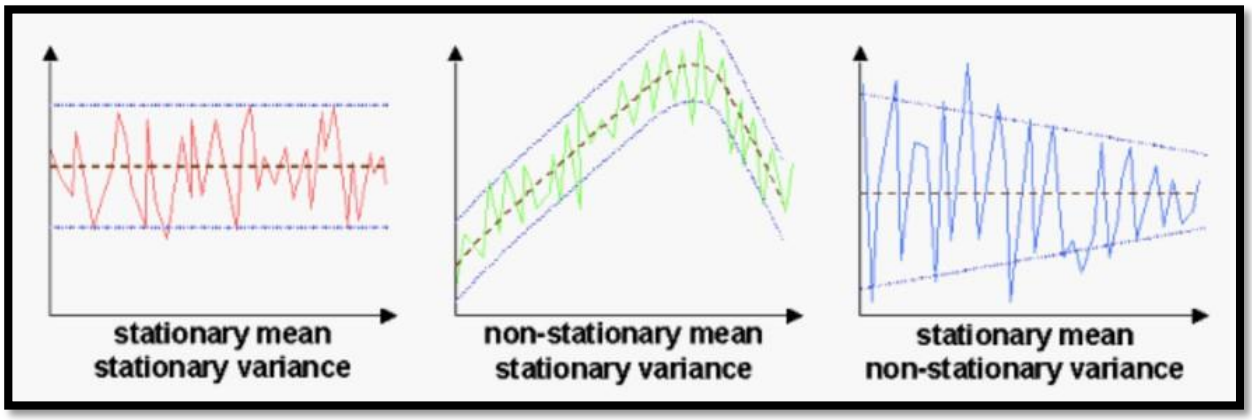
$$cov(X_{t_1}, X_{t_2}) = K_{XX}(t_1, t_2) = K_{XX}(t_2 - t_1, 0) = K_{XX}(\tau)$$

Note that this directly implies that the variance of the process is also constant, since we get that for all $t\in\mathbb{N}$

$$Var(X_t) = cov(X_t, X_t) = K_{XX}(t, t) = K_{XX}(0) = d$$

This paints a specific picture of weakly stationary processes as those with constant mean and variance. Their properties are contrasted nicely with those of their counterparts in Figure below.

Strong stationarity does not imply weak stationarity, nor does the latter imply the former.



In order to test for stationarity we plot the Auto Correlation Function and Partial Auto correlation functions of the BANK NIFTY daily indices.

ACF & PACF

The autocorrelation function (ACF) reveals how the correlation between any two values of the signal changes as their separation changes. It is a time domain measure of the stochastic process memory, and does not reveal any information about the frequency content of the process. Generally, for an error signal, the ACF is defined as,

$$\rho_k = \frac{\mathrm{Cov}(e_t, e_{t+k})}{\sqrt{Var(e_t)\, Var(e_{t+k})}}$$

For a stationary stochastic process of variance σ^2, the previous expression for the ACF reduces to

$$\rho_k = \frac{Cov(e_t, e_{t+k})}{\sigma^2}$$

which is time-independent.

ACF plot or correlogram is a bar chart of coefficients of correlation between a time series and it lagged values in vertical axis and the corresponding lags in horizontal axis. Simply stated: ACF explains how the present value of a given time series is correlated with the past (1-unit past, 2-unit past,..., n-unit past) values. In the ACF plot, the y-axis expresses the correlation coefficient whereas the x-axis mentions the number of lags. Assume that; y(t), y(t- 1),...., y(t-n) are values of a time series at time t, t-1,...,t-n, then the lag-1 value is the correlation coefficient between y(t) and y(t- 1), lag-2 is the correlation coefficient between y(t) and y(t-2) and so on.

If the lags or the spikes lie within the blue band then we can say the data is stationary else the data is non-stationary. If the coefficients are high at certain lags that are above the blue band we can say that high positive correlation exists for those points of the data which implies non-stationarity.

A partial correlation is a conditional correlation. It is the correlation between two variables under the assumption that we

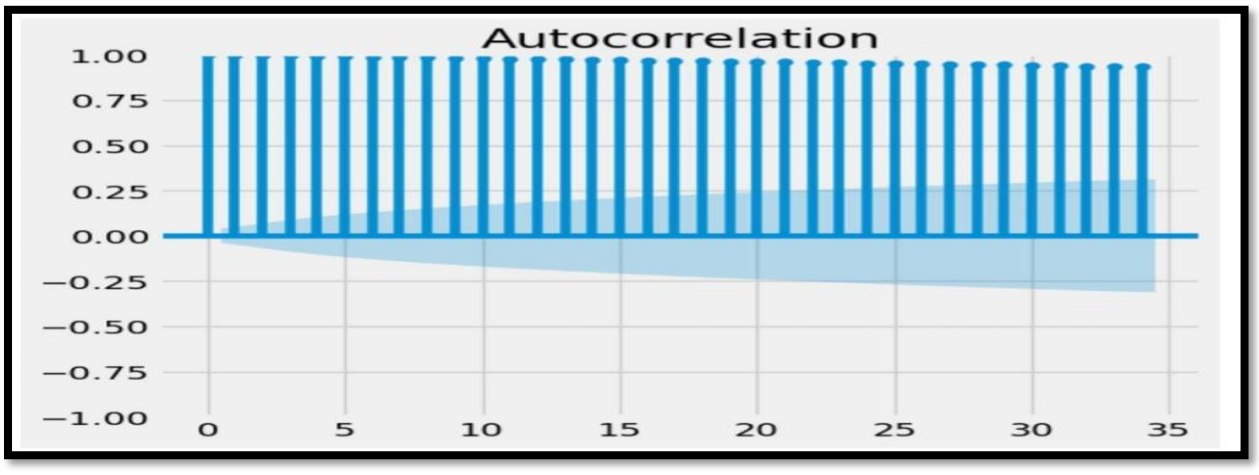
know and take into account the values of some other set of variables. For instance, consider a regression context in which y is the response variable and x1, x2 and x3 are predictor variables. The partial correlation between y and x3 is the correlation between the variables determined taking into account how both y and x3 are related to x1 and x2. The partial correlation is given by,

$$\frac{\text{Covariance}(y, x_3|x_1, x_2)}{\sqrt{\text{Variance}(y|x_1, x_2)\ \text{Variance}(x_3|x_1, x_2)}}$$

PACF is the partial autocorrelation function that explains the partial correlation between the series and lags itself. In simple terms, PACF can be explained using a linear regression where we predict y(t) from y(t-1), y(t-2), and y(t-3). In PACF, we correlate the "parts" of y(t) and y(t-3) that are not predicted by y(t-1) and y(t-2).

The ACF and PACF for the BANK NIFTY daily indices is given by in the following graphs

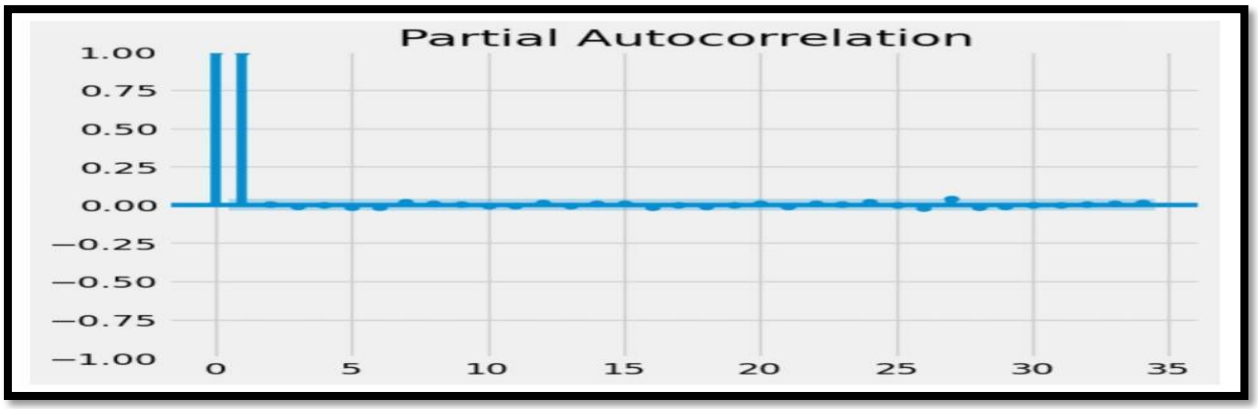
```
from statsmodels.graphics.tsaplots import plot_acf
plt.figure(figsize=(12,12))
plot_acf(DF1['Close'].dropna())
plt.tick_params(axis='x',labelsize=14)
plt.tick_params(axis='y',labelsize=14)
plt.show()

<Figure size 1200x1200 with 0 Axes>
```



ACF Plot

```
from statsmodels.graphics.tsaplots import plot_pacf
plt.figure(figsize=(12,12))
plot_pacf(DF1['Close'].dropna())
plt.tick_params(axis='x',labelsize=14)
plt.tick_params(axis='y',labelsize=14)
plt.show()
```

```
<Figure size 1200x1200 with 0 Axes>
```



PACF Plot

From both ACF and PACF plots we can conclude that the data is Non-Stationary.

To further support the findings about the stationarity we performed a formal test. Dickey Fuller Test and Augmented Dickey Fuller tests which are based on Null Hypothesis of non-stationarity, are the most acceptable method to find the stationary of a series.

Dickey-Fuller Test

The Dickey-Fuller (DF) test was developed and popularized by Dickey and Fuller (1979). The null hypothesis of DF test is that there is a unit root in an AR model, which implies that the series is not stationary. The alternative hypothesis is generally stationarity or trend stationarity but can be different depending on the version of the test is being used.
A Dickey-Fuller test is a unit root test that tests the null hypothesis that $\alpha=1$ in the following model equation, alpha is the coefficient of the first lag on Y.

Null Hypothesis(H_0): alpha=1

where,

- y(t-1) = lag 1 of time series
- delta y(t-1) = first difference of the series at time (t-1)

Fundamentally, it has a similar null hypothesis as the unit root test. That is, the coefficient of y(t-1) is 1, implying the presence of a unit root. If not rejected, the series is taken to be non-stationary.

Augmented Dickey–Fuller Test

ADF (Augmented Dickey-Fuller) test is a statistical significance test which means the test will give results in hypothesis tests with null and alternative hypothesis and is an expansion of the Dickey Fuller Test.

The Augmented Dickey-Fuller test is a type of statistical test called a unit root test.

In probability theory and statistics, a unit root is a feature of some stochastic processes (such as random walks) that can cause problems in statistical inference involving time series models. In a simple term, the unit root is non-stationary but does not always have a trend component.

ADF test is conducted with the following assumptions.

Null Hypothesis (H_O): Series is non-stationary or series has a unit root.

Alternate Hypothesis (H_A): Series is stationary or series has no unit root.

If the null hypothesis is failed to be rejected, this test may provide evidence that the series is non-stationary.

Conditions to Reject Null Hypothesis (H_O)

> If Test statistic < Critical Value and p-value < 0.05 – Reject Null Hypothesis (H_O), i.e. time series does not have a unit root, meaning it is stationary. It does not have a time-dependent structure.

The ADF test expands the Dickey-Fuller test equation to include high order regressive process in the model.

$$y_t = c + \beta t + \alpha y_{t-1} + \phi_1 \Delta Y_{t-1} + \phi_2 \Delta Y_{t-2} . . + \phi_p \Delta Y_{t-p} + e_t$$

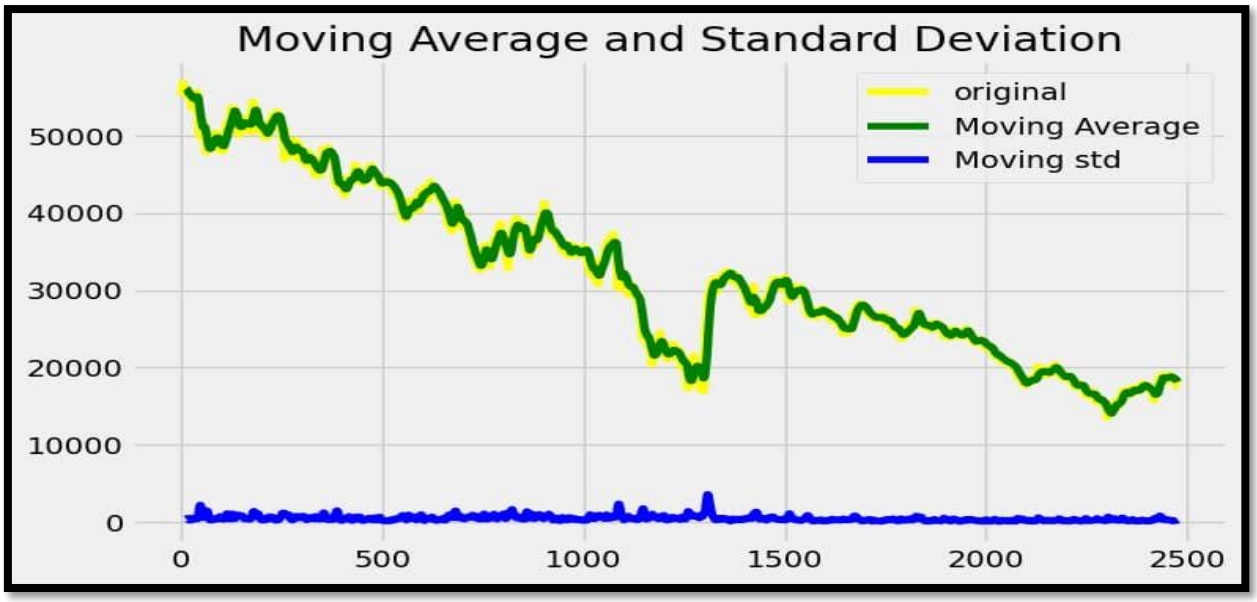
Thus for a dataset to be stationary, p-value must be less than 5%. But in this case, the p-value is greater than 5%.

The following lines of code help us to conduct the Augmented Dickey Fuller Test-

```
def test_stationarity(timeseries):
    plt.figure(figsize=(8,5))
    plt.grid(True)
    plt.tick_params(axis='x',labelsize=14)
    plt.tick_params(axis='y',labelsize=14)
    #Determing Rolling Statistics
    rolmean = timeseries.rolling(12).mean()
    rolstd = timeseries.rolling(12).std()
    #Plot rolling statistics
    plt.plot(timeseries,color='yellow',label='original')
    plt.plot(rolmean,color='green',label='Moving Average')
    plt.plot(rolstd,color='blue',label='Moving std')
    plt.legend(loc='best')
    plt.title('Moving Average and Standard Deviation')
    plt.show(block=False)
    print("Results of Dickey Fuller Test")
    from statsmodels.tsa.stattools import adfuller
    adft = adfuller(timeseries,autolag='AIC')
    output = pd.Series(adft[0:4],index=['Test Statistics','P-
value','No. of lags used','No. of observations used'])
    for key ,values in adft[4].items():
        output['critical value (%s)'%key] = values
    print(output)

test_stationarity(DF1['Close'])
```

Code of the Augmented Dickey Fuller Test

The result for the above code is given below:



Results of Dickey Fuller Test	
Test Statistics	-1.805116
P-value	0.377925
No. of lags used	6.000000
No. of observations used	2471.000000
critical value (1%)	-3.432999
critical value (5%)	-2.862710
critical value (10%)	-2.567393
dtype: float64	

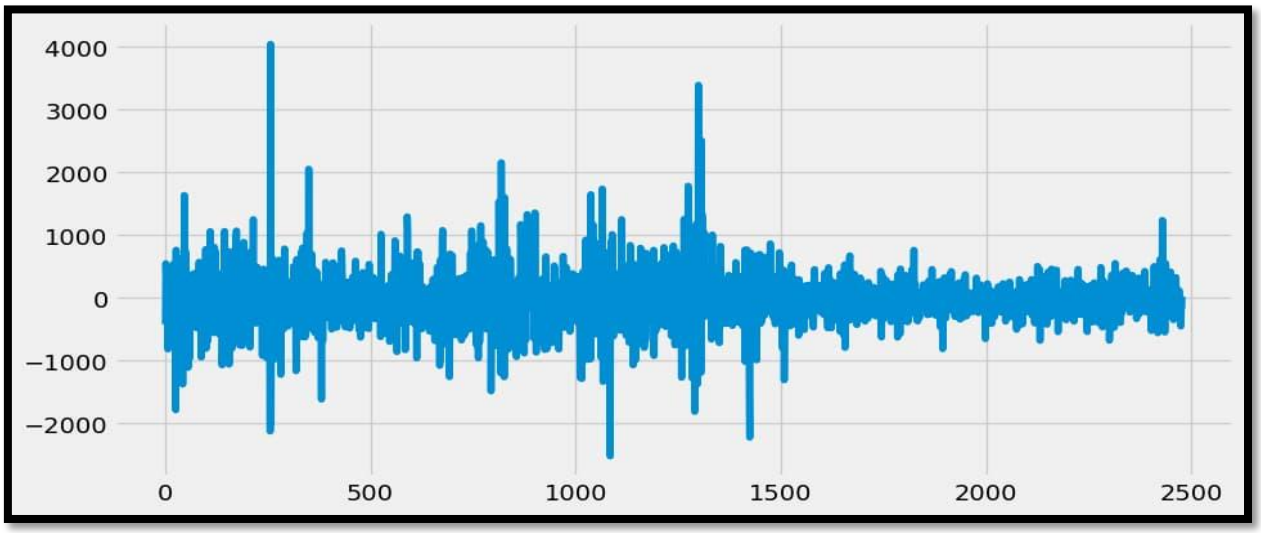
Result of the Augmented Dickey Fuller Test

Here p value is greater than 0.05 thus we can say that the data is non-stationary as we fail to reject the null hypothesis.

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. 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.

The code snippet in the following figure shows the differencing and the plot after differencing.

```
DF1['Close'] = DF1['Close'] - DF1['Close'].shift(1)
plt.figure(figsize=(10,5))
plt.tick_params(axis='x',labelsize=14)
plt.tick_params(axis='y',labelsize=14)
DF1['Close'].dropna().plot()
```

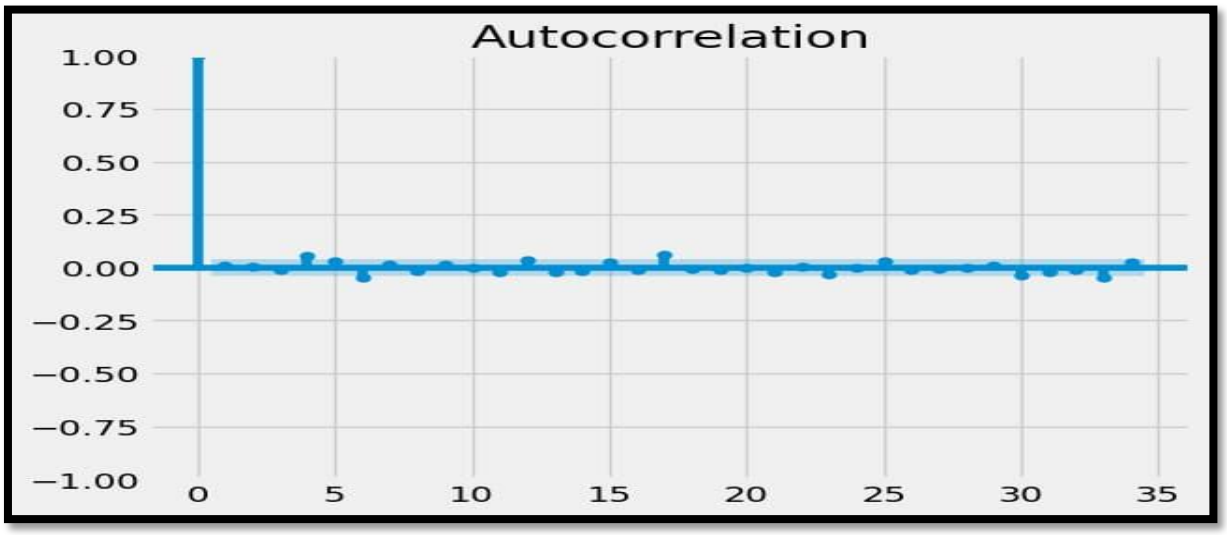


AFTER FIRST DIFFERENCING

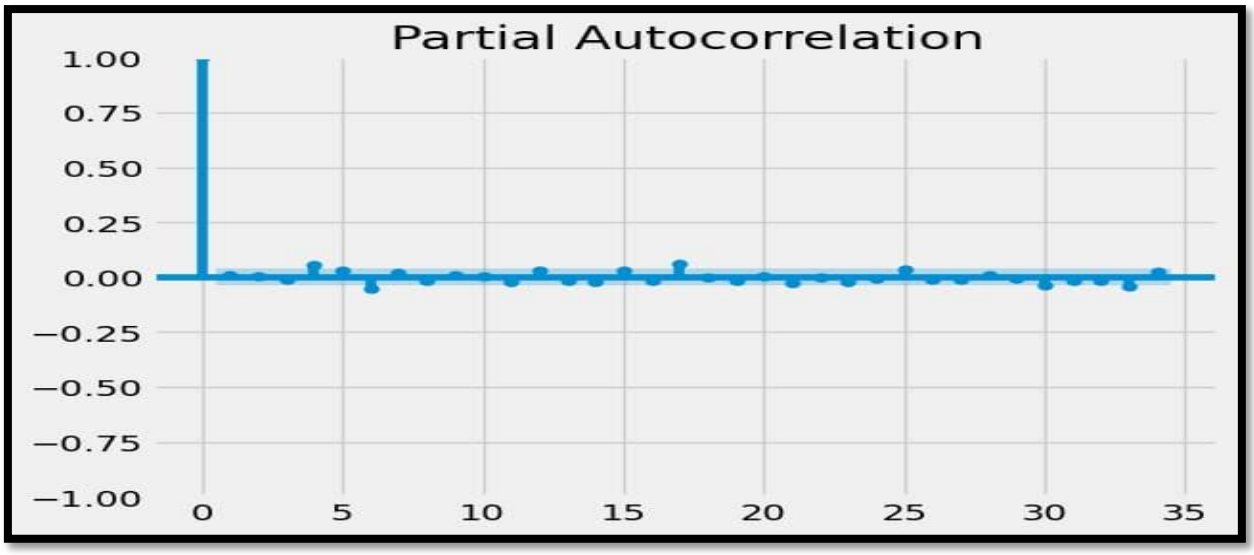
From the graph plotted we can conclude that the data is stationary after first differencing.

We again check for stationarity of the first differenced data set using the above methods- ACF, PACF plots and Dickey Fuller Test.

The ACF plot and PACF plots are as follows-

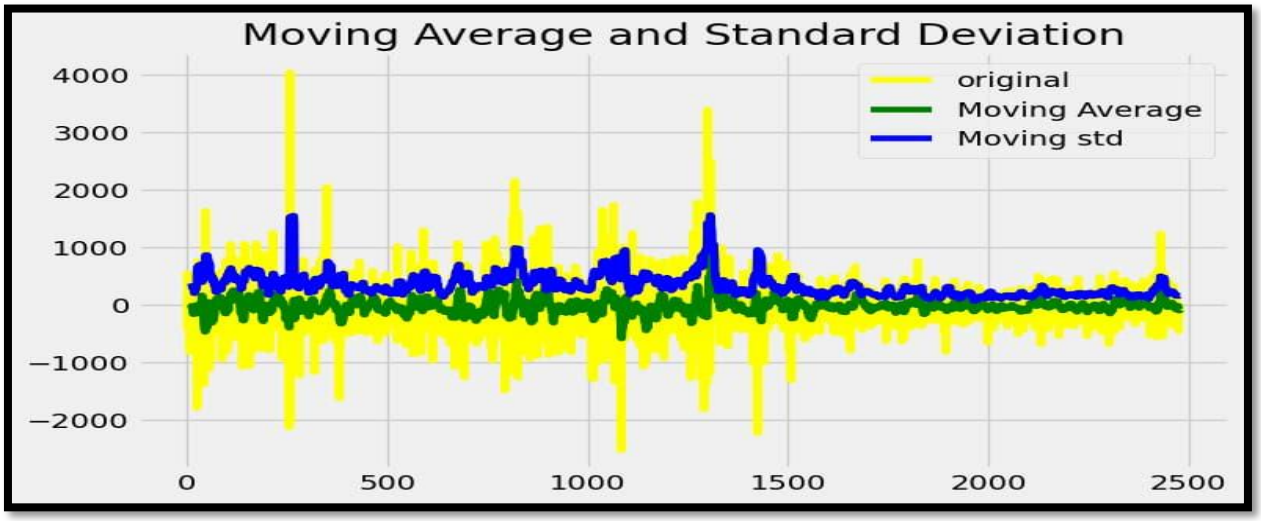


ACF Plot after First Differencing



PACF Plot after First Differencing

The results of Augmented Dicky Fuller is given below:




```
Results of Dickey Fuller Test
Test Statistics                -20.145852
P-value                          0.000000
No. of lags used                 5.000000
No. of observations used      2471.000000
critical value (1%)             -3.432999
critical value (5%)             -2.862710
critical value (10%)            -2.567393
dtype: float64
```

Result of the Dicky Fuller Test after First Differencing

Here p-value is less than 0.05 which means null hypothesis is rejected.

From the above three tests we can conclude that the data after first differencing is stationary.

Methodology

ARIMA

An autoregressive integrated moving average, or ARIMA, is a statistical analysis model that uses time series data to either better understand the data set or to predict future trends. A statistical model is autoregressive if it predicts future values based on past values.

Each component in ARIMA functions as a parameter with a standard notation. For ARIMA models, a standard notation would be ARIMA with p, d, and q, where integer values substitute for the parameters to indicate the type of ARIMA model used. The parameters can be defined as:

p: the number of lag observations in the model; also known as the lag order.

d: the number of times that the raw observations are differenced; also known as the degree of differencing.

q: the size of the moving average window; also known as the order of the moving average.

The AR part of ARIMA indicates that the evolving variable of interest is regressed on its own lagged values. The MA part indicates that the regression error is actually a linear combination of error terms whose values occurred contemporaneously and at various times in the past. The I (for "integrated") indicates that the data values have been replaced with the difference between their values and the previous values (and this differencing process may have been performed more than once).

$\{X_t\}$ is said to follow an Auto Regressive Integrated Moving Average (ARIMA) model of order (p, d, q) if,

$$Z_t = \nabla d X_t = (1 - B)\, d X_t \sim ARMA(p, q)$$

Where $\nabla = (1-B)$ and B is the difference operator and B is the back-shift operator, so it follows,

$$Z_t = \phi_1 Z_{t-1} + \cdots + \phi_p Z_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q}$$

Where $Z_t = \nabla d X_t$; i.e. d^{th} difference of X_t and $\phi_1, \phi_2, \ldots\ldots, \phi_p$ are the MA parameters of the model, and $\theta_1, \theta_2, \ldots\ldots, \theta_q$ are the AR parameters, and $\epsilon_t, \epsilon_{t-1}, \ldots\ldots, \epsilon_{t-q}$ are white noise error terms.

This can as well be written in terms of back-shift operator as,

$$\phi(B) Z_t = \theta(B) \epsilon_t$$

$$\Rightarrow \phi(B) \nabla d X_t = \theta(B) \epsilon_t$$

$$\Rightarrow \phi(B)(1 - B)\, d X_t = \theta(B) \epsilon_t$$

$$\phi^*(B) X_t = \theta(B) \epsilon_t$$

Where, $\phi(B) = 1 - \phi_1 B - \ldots - \phi_p B^p$ and $\theta(B) = 1 - \theta_1 B_1 + \ldots + \theta_q B_q$

Thus, the model for $\{X_t\}$ is $\phi^*(B) X_t = \theta(B) \epsilon_t$

$\phi^*(B) = \phi(B)(1 - B)\, d \Rightarrow X_t \sim ARMA(p + d, q)$

So, ARIMA(p,d,q) $\Rightarrow$ ARMA(p+d,q)

Dividing the model into training and testing data

The model is now divided into training and testing data. 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.

In machine learning, datasets are split into two subsets. The first subset is known as the training data– it's a portion of our actual dataset that is fed into the machine learning model to discover and learn patterns. In this way, it trains our model. The other subset is known as the testing data. Training data is typically larger than

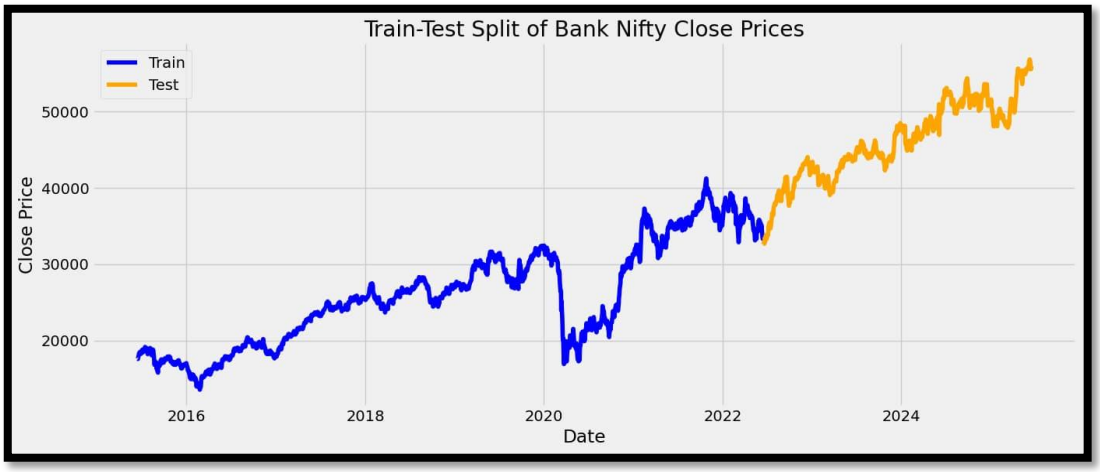
testing data. This is because we want to feed the model with as much data as possible to find and learn meaningful patterns. Once data from our datasets are fed to a machine learning algorithm, it learns patterns from the data and makes decisions.

Once our machine learning model is built (with our training data), we need unseen data to test your model. This data is called testing data, and you can use it to evaluate the performance and progress of your algorithms' training and adjust or optimize it for improved results. The code below divides the model into training and testing data:

```
# Train-test split (70% train, 30% test)
split_idx = int(len(df_filtered) * 0.7)
train_df = df_filtered.iloc[:split_idx]
test_df = df_filtered.iloc[split_idx:]

# Plotting
plt.figure(figsize=(14, 6))
plt.plot(train_df['Date'], train_df['Close'], label='Train',
color='blue')
plt.plot(test_df['Date'], test_df['Close'], label='Test',
color='orange')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.title('Train-Test Split of Bank Nifty Close Prices')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

Code snippet dividing the data into training and testing

The plot shown below shows how the model is divided into training and testing data-



Graph showing the closing price divided into training and testing data

Auto-ARIMA

Now that the daily indices dataset for BANK NIFTY has 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, 1, 1) is the best model.

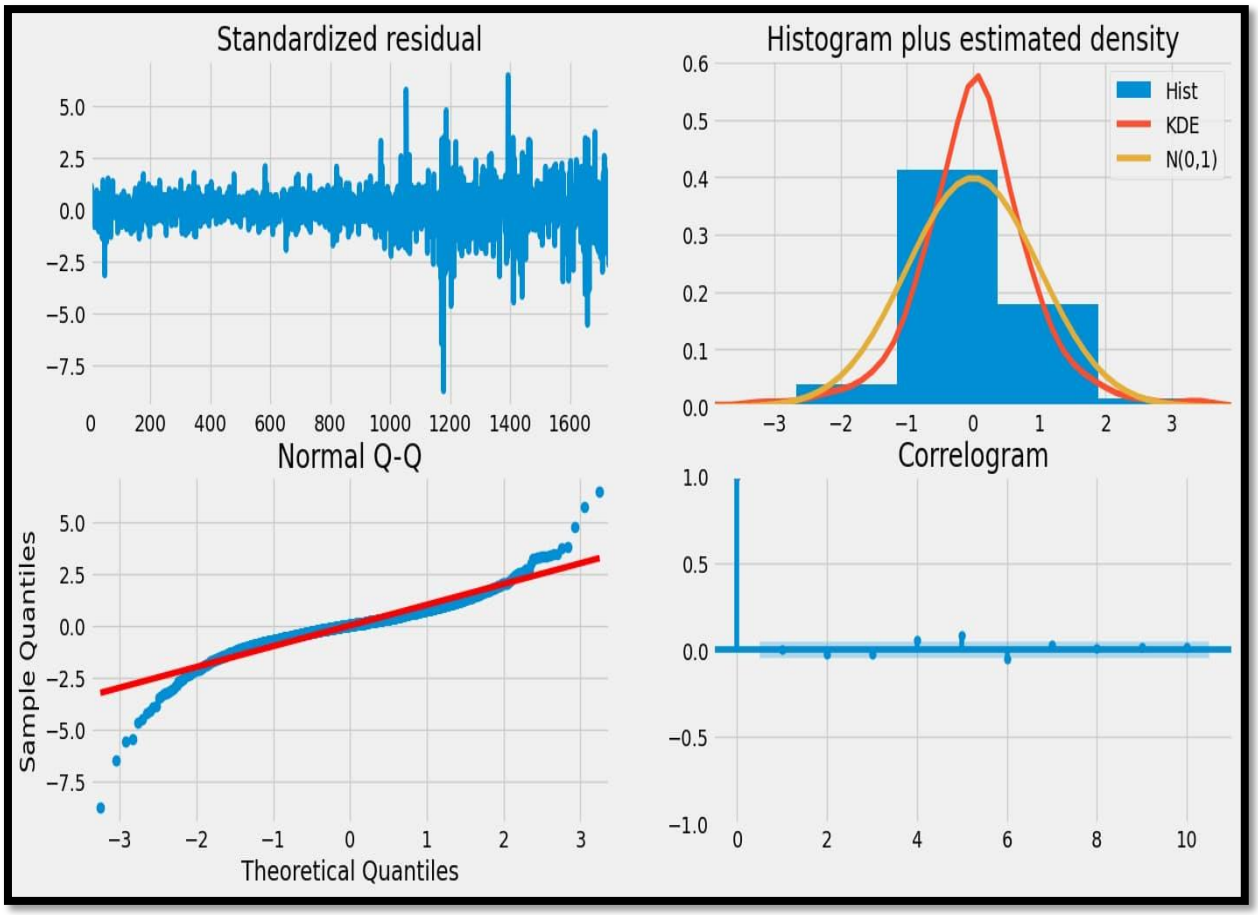
The code snippet of the auto-ARIMA is given below along with the output-

```
from statsmodels.tsa.arima_model import ARIMA
from pmdarima.arima import auto_arima

model_autoARIMA = auto_arima(train_df['Close'],start_p=0,start_q=0,
                      test='adf',
                      max_p=3,max_q=3,
                      m=1,
                      d=None,
                      seasonal=False,
                      start_P=0,
                      D=0,
                      trace=True,
                      error_action='ignore',
                      supress_warnings=True,
                      stepwise=True)
print(model_autoARIMA.summary())
model_autoARIMA.plot_diagnostics(figsize=(15,8))
plt.show()
```

Code snippet of Auto-ARIMA

```
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=25599.025, Time=0.05 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=25597.597, Time=0.06 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=25597.431, Time=0.06 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=25597.965, Time=0.01 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=25599.118, Time=0.16 sec
 ARIMA(0,1,2)(0,0,0)[0] intercept   : AIC=25598.694, Time=0.14 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=25600.574, Time=0.36 sec
 ARIMA(0,1,1)(0,0,0)[0]             : AIC=25596.277, Time=0.04 sec
 ARIMA(1,1,1)(0,0,0)[0]             : AIC=25597.946, Time=0.07 sec
 ARIMA(0,1,2)(0,0,0)[0]             : AIC=25597.581, Time=0.06 sec
 ARIMA(1,1,0)(0,0,0)[0]             : AIC=25596.445, Time=0.03 sec
 ARIMA(1,1,2)(0,0,0)[0]             : AIC=25599.179, Time=0.19 sec

Best model:  ARIMA(0,1,1)(0,0,0)[0]
Total fit time: 1.230 seconds
                                SARIMAX Results
```

```
Dep. Variable:                           y   No. Observations:
1734
Model:                 SARIMAX(0, 1, 1)   Log Likelihood                -
12796.138
Date:                  Tue, 17 Jun 2025   AIC
25596.277
Time:                          21:21:43   BIC
25607.192
Sample:                                0   HQIC
25600.314
                                - 1734

Covariance Type:                     opg

==============================================================================
========
                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
--------
ma.L1          0.0467      0.015      3.110      0.002       0.017
0.076
sigma2      1.519e+05   2235.429     67.960      0.000    1.48e+05
1.56e+05
==============================================================================
=============
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
5773.80
Prob(Q):                              0.99   Prob(JB):
0.00
Heteroskedasticity (H):               7.46   Skew:
-0.58
Prob(H) (two-sided):                  0.00   Kurtosis:
11.87
```



Graphs showing the model to be fitted

Fitting of the ARIMA 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 later

```
import warnings
import statsmodels.api as sm
warnings.filterwarnings('ignore','statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)
model=sm.tsa.arima.ARIMA(DF1,order=(0,1,1))
fitted=model.fit()

print(fitted.summary())
```

Fitting ARIMA of order (0,1,1)

```
                               SARIMAX Results
==========================================================================
========
Dep. Variable:                   Close   No. Observations:
2478
Model:                 ARIMA(0, 1, 1)   Log Likelihood                -
18431.607
Date:                Tue, 17 Jun 2025   AIC
36867.213
Time:                        21:28:40   BIC
36878.843
Sample:                             0   HQIC
36871.437
                                - 2478
Covariance Type:                   opg
==========================================================================
========
                 coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------
--------
ma.L1         -0.9992      0.004   -260.378      0.000      -1.007
-0.992
sigma2      1.692e+05   2156.676     78.431      0.000    1.65e+05
1.73e+05
==========================================================================
=============
Ljung-Box (L1) (Q):                   0.32   Jarque-Bera (JB):
9738.07
Prob(Q):                              0.57   Prob(JB):
0.00
Heteroskedasticity (H):               0.21   Skew:
0.70
Prob(H) (two-sided):                  0.00   Kurtosis:
12.61
```

Summary of the ARIMA model fitted

Prediction of future values

Now that the ARIMA model is trained for future predictions, the next process is to predict the stock prices of the companies for the training data using fitted.forecast() function. We fit the ARIMA model of order (0,1,1) to the training data and now use the model to predict the values for testing data. The following code snippet shows how the values of the test data are predicted along with its output:

```
model=sm.tsa.arima.ARIMA(train_df['Close'],order=(0,1,1))
fitted=model.fit()
# Get forecast and confidence interval
fc, confint = model_autoARIMA.predict(n_periods=744,
return_conf_int=True)

# Convert to Series
fc_series = pd.Series(fc, index=test_df.index)
lower_series = pd.Series(confint[:, 0], index=test_df.index)
upper_series = pd.Series(confint[:, 1], index=test_df.index)

#plot
plt.figure(figsize=(10,5))
plt.plot(train_df['Close'],label='Training Data')
plt.plot(test_df['Close'],color='green',label='Actual Stock Price')
plt.plot(fc_series,color='orange',label='Predicted Stock Price')
#plt.fill_between(lower_series.index,lower_series,upper_series,color='
k',alpha=0.10)
plt.title('Bank Nifty Stock Price')
plt.xlabel('Date')
plt.ylabel('Bank Nifty Stock Price')
plt.legend(loc='upper left',fontsize=8)
plt.show()
```

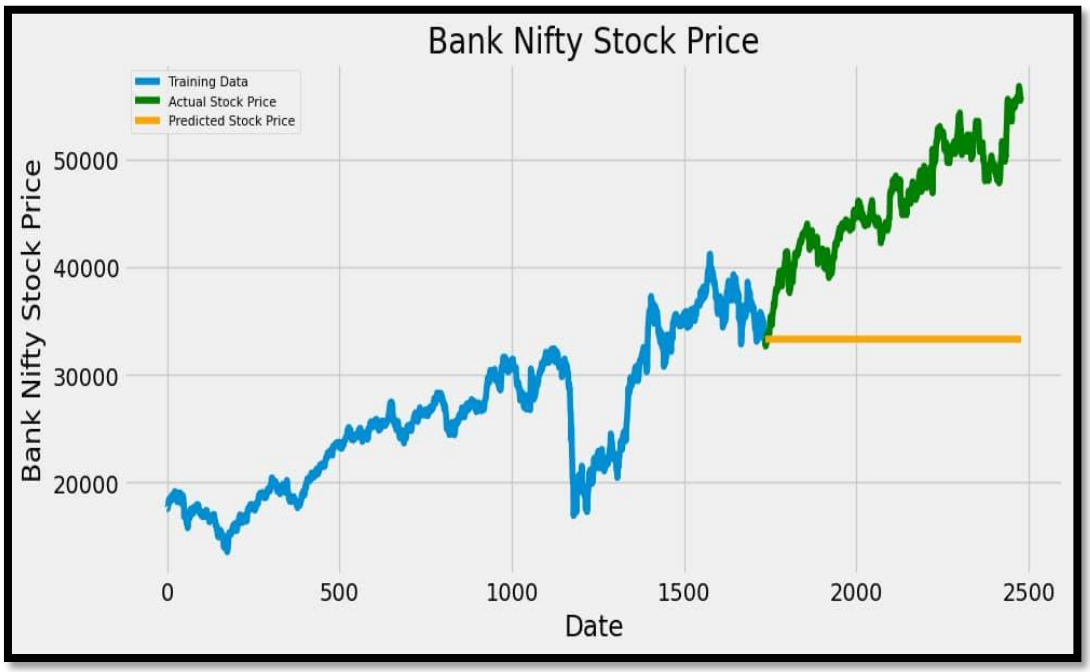
Code to predict values using ARIMA

The overview of predicted values as predicted by the model are-

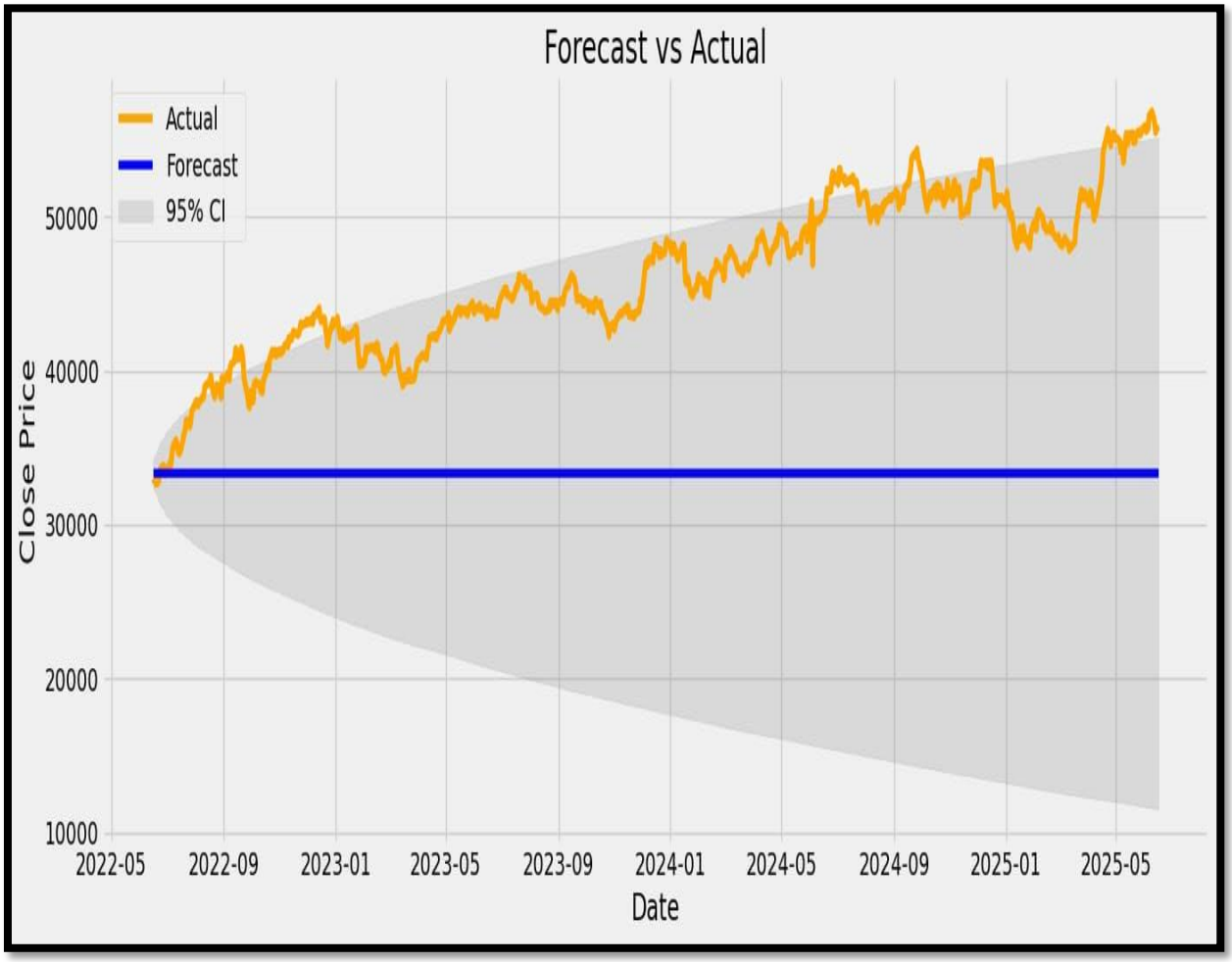
```
print("Forecasted Values:\n", fc_series.head(10))

Forecasted Values:
 1734    33340.389634
1735    33340.389634
1736    33340.389634
1737    33340.389634
1738    33340.389634
1739    33340.389634
1740    33340.389634
1741    33340.389634
1742    33340.389634
1743    33340.389634
dtype: float64
```

The values predicted by ARIMA

We plot a graph of the training data and the real v/s predicted testing data.



Graph showing predicted values using ARIMA



Root Mean Square Error (RMSE)

Root Mean Square Error (RMSE) is a standard way to measure the error of a model in predicting quantitative data. Formally it is defined as follows:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

$\hat{y}_1$, $\hat{y}_2$,..., $\hat{y}_n$ are predicted values.

y_1 , y_2 , ..., y_n are observed values.

N is the number of observations.

From the value of the RMSE we can say how accurate a model is in prediction.

The RMSE of the ARIMA Model is calculated as follows:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error

mae = mean_absolute_error(test_df['Close'], fc_series)
rmse = mean_squared_error(test_df['Close'], fc_series, squared=False)

print(f"MAE: {mae:.2f}")
print(f"RMSE: {rmse:.2f}")

MAE: 12800.87
RMSE: 13779.01
```

Where df[“Close”] is the actual value and fc is the predicted value and the variable MSE holds the mean of the square of differences of real and predicted values.

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. However, the only drawback of this analysis is that ARIMA model holds lower accuracy for long-term predictions and cannot predict properly in presence of high volatility.

Thus to detect the volatility we use the ARCH MODEL.

Volatility

Volatility is the rate at which the price of a stock increases or decreases over a particular period.

Higher stock price volatility often means higher risk and helps an investor to estimate the fluctuations that may happen in the future.

Volatility is the standard deviation of a stock's annualized returns over a given period and shows the range in which its price may increase or decrease.

If the price of a stock fluctuates rapidly in a short period, hitting new highs and lows, it is said to have high volatility. If the stock price moves higher or lower more slowly, or stays relatively stable, it is said to have low volatility.

Historic volatility is calculated using a series of past market prices, while implied volatility looks at expected future volatility, using the market price of a market-traded derivative like an option.

ARCH Model

Stock prices prediction often emphasizes on developing a model to predict behaviour of the data. However, to achieve best results the stock market return should be characterised as a combination of drift and volatility. As risk uncertainty considerations of stock price pose a major concern, a new type of time series modelling technique has emerged to take account of conditional variances along with the mean behaviour. Autoregressive Conditional Heteroskedasticity model (ARCH) model, provides us with a framework to model the time varying variances. In financial time series it has been observed that in volatility (time varying variance) clustering large changes tend to follow large changes and small changes tend to follow small changes. This phenomenon is called conditional heteroskedasticity and can be modelled by ARCH model, a later generalization of it GARCH (Generalized ARCH) model was proposed. While conventional time series econometric models drive under the assumption of constant mean, the ARCH process introduced allows the conditional variance to change over time as a function of past error leaving the unconditional variance constant. Several studies have further pointed out that the Autoregressive Integrated Moving Average (ARIMA) with ARCH errors is found to be successful in modelling some macroeconomic time series.

In this section we will briefly discuss ARCH model.

The ARCH-type model is a non- linear model which we will try to define in terms of the distributions of the errors of a dynamic linear regression model. Suppose that we are modelling the variance of a series y_t. y_t can be modelled as $Y_t = x'_t \zeta + \epsilon_t$, t = 1, 2, ….,T.

Where x_t is a k × 1 vector of exogenous variables which may or may not include lagged values of dependent variable ζ is a k × 1 vector of regression parameters. The distribution of stochastic error ϵt is conditioned on the realized values of set of variables $\Psi_{t-1} = \{y_{t-1}, x_{t-1}, y_{t-2}, x_{t-2},...\}$ Engle assumed,

$$\epsilon_t \,|\Psi_{t-1} \sim N(0, h_t)$$

where, $h_t = \alpha_0 + \alpha_1 \epsilon^2_{t-1} + ... + \alpha_q \epsilon^2_{t-q}$, $\alpha_0 > 0$

and $\alpha_i \geq 0$, i = 1(1)q so that the conditional variance is positive since, $\epsilon_{t-i} = y_{t-i} - x'_{t-i}\,\zeta$, i = 1(1)q , h_t clearly is a function of Ψ_{t-1}.

In ARCH regression model the variance of error ϵt is conditions on the realised value of lagged errors ϵ_{t-i} , i = 1,2,. . . ,q.

Since variance is expected squared deviation, a linear combination of lagged squares is a natural measure of the recent trend in variance to translate to the current conditional variance h_t. The current error ϵt is an increasing function of magnitude of the lagged errors irrespective of their signs. So large errors of either sign tend to be followed by a large error of either sign and similarly small error of either sign tend to be followed by a small error of either sign. From the order of lag q the length of time for which shock persists in conditioning the variance of subsequent errors is determined. Larger value of q indicates longer episodes of volatility.

ARIMA-ARCH Model

The ARIMA-ARCH model is one model in which the variance of the error term of the ARIMA model follows an ARCH process. We consider the squared residuals after fitting ARIMA(0,1,1) to our data, i.e. we consider $r^2_t = (y_t - \hat{y}_t)^2$ for t = 1,2,.....,n. We will apply Engle's Lagrange Multiplier Test to test for the existence of ARCH-Effect in the series $r^2{}_t$.

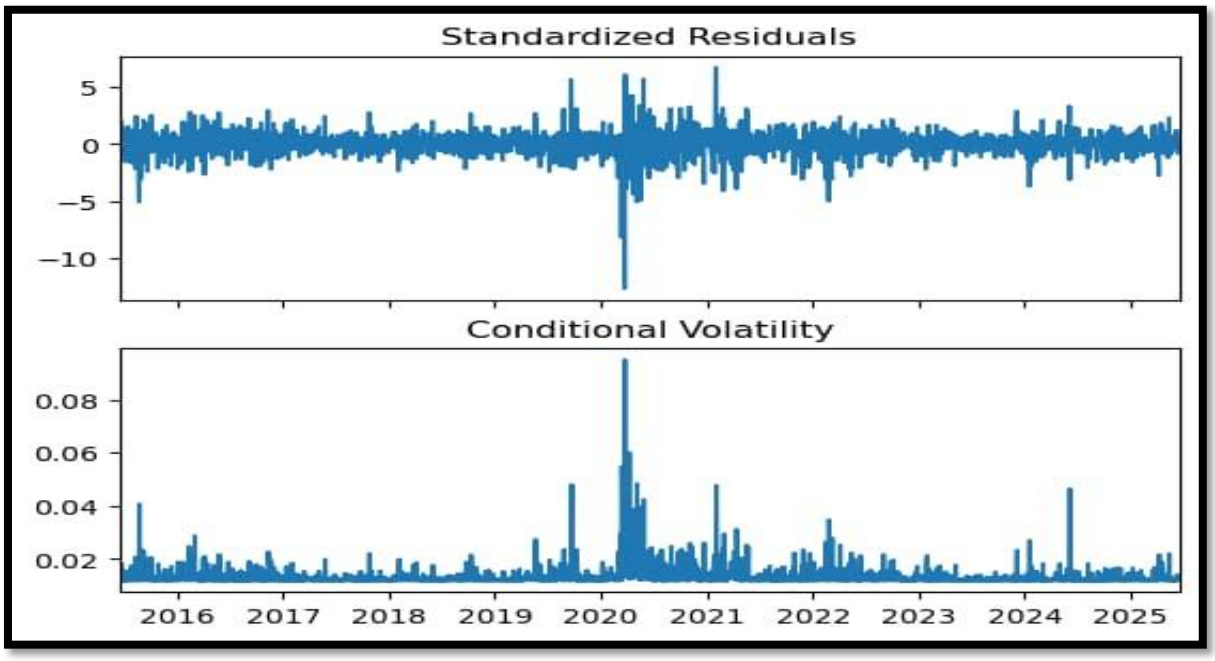
Since we have seen that we don't get accurate results by fitting ARIMA due to the presence of volatility in the data, we calculate the returns of the BANK NIFTY daily stock indices

Suppose that the data is given by, z_t ,t=1,...,n.

Now consider the return series, relative first order difference given by :

$$y_t = \frac{y_t - y_{t-1}}{y_t} * 100$$

We fit the ARIMA of order (0,0,0) to the returns dataset and plot its residuals. The residual plot is given below:



Plot showing the residuals

We plot the residuals to check whether it is stationary in order to fit the ARCH Model. From the above plot, we can say that the residuals are stationary and volatility exists as the variance of the residuals are increasing and decreasing in nature.

The detection of the ARCH effect in a time series is a test of serial independence applied to the serially uncorrelated fitting error of some model, in our case ARIMA model. We have assumed that linear serial dependence inside the original series is removed with an efficient model. Hence, any further serial dependence must be due to some nonlinear mechanism which has not been

detected by the model. Here, the nonlinear mechanism we are concerned with is the conditional heteroskedasticity.

Lagrange Multiplier Test in this testing process,

H_0 : ARCH-Effect is not present

H_1 : ARCH-Effect is present

This procedure simply involves obtaining the squares of the residual from fitted model r^2_t and regress them on a constant and p lagged values, where i is the ARCH lags. Let us consider the equation:

$$r^2_t = \alpha_0 + \sum_{i=1}^{p} . \alpha_i r^2_{t-i}$$

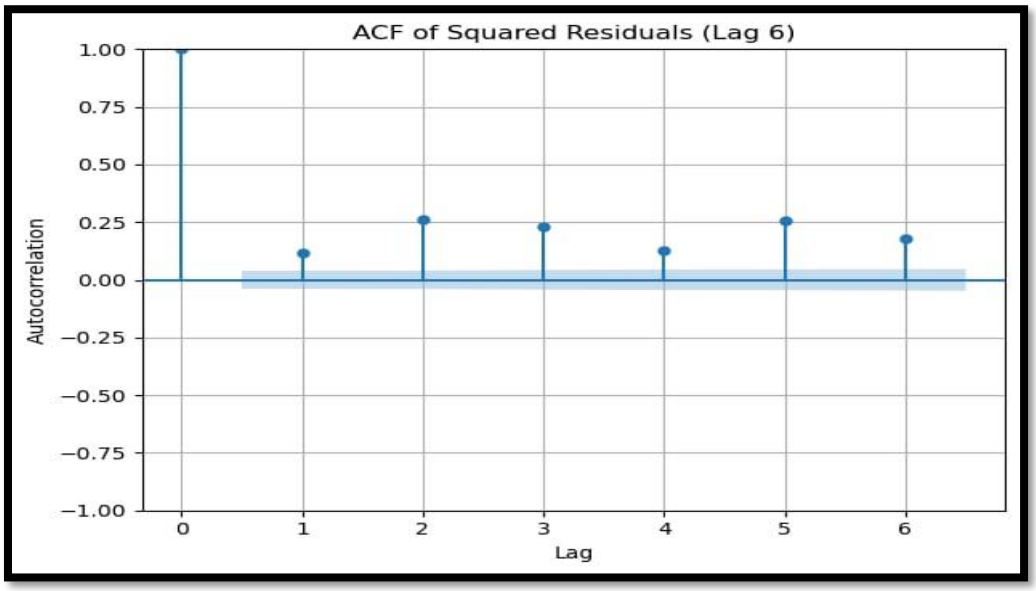
The hypothesis is that, in the absence of ARCH components, we have $\alpha_i = 0$ for all i=1,2,.....,p, against the alternative that, in the presence of ARCH components, at least one of the estimated α_i must be significant. The test statistic is given by nR^2 . Here R is the sample multiple correlation coefficient computed from the above regression of r^2_t on r^2_{t-1} , ..., r^2_{t-p} . Under the null hypothesis H_0 (ARCH effect is not present), the nR^2 asymptotically follows a χ^2 distribution with degrees of freedom p.

Thus here the p-value is 6.318e-04< 0.05. Hence at 5% level of significance we reject the null hypothesis and conclude that the squared residual series contains ARCH-effect.

Now we plot the ACF plot of the squared residuals with lag =6. We plot the ACF plot to check if any correlation is out of the range, if there are significant spikes we can say that correlation of squared residual exists however if there are no spikes we can say that no correlations are significant hence the series is white noise.

For the above residuals the correlogram containing the ACF is given below:

```
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_acf

# Square the residuals
squared_residuals = residuals**2

# Plot ACF of squared residuals up to lag 2
plot_acf(squared_residuals, lags=6, alpha=0.05)
plt.title("ACF of Squared Residuals (Lag 6)")
plt.xlabel("Lag")
plt.ylabel("Autocorrelation")
plt.grid(True)
plt.tight_layout()
plt.show()
```

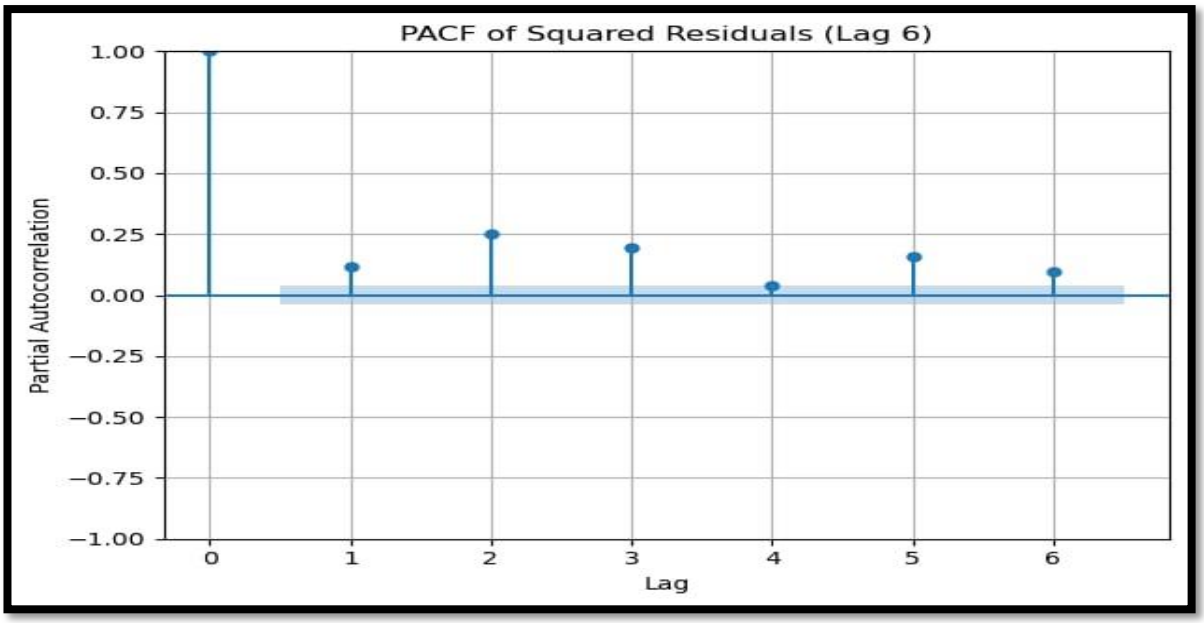


ARIMA(0,0,0) ACF Plot

ACF plot of squared residuals that there are significant number of spikes (upto order 1), at lag 6, hence we can say that correlation of squared residuals exists, therefore volatility exists.

Again we plot the PACF plot for the squared residuals. The correlogram containing the same is given below-

```
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_pacf

# Plot PACF of squared residuals
plot_pacf(squared_residuals, lags=6, alpha=0.05, method='ywm')
plt.title("PACF of Squared Residuals (Lag 6)")
plt.xlabel("Lag")
plt.ylabel("Partial Autocorrelation")
plt.grid(True)
plt.tight_layout()
plt.show()
```



ARIMA(0,0,0) PACF plot

In the PACF plot, spikes up to order 1.

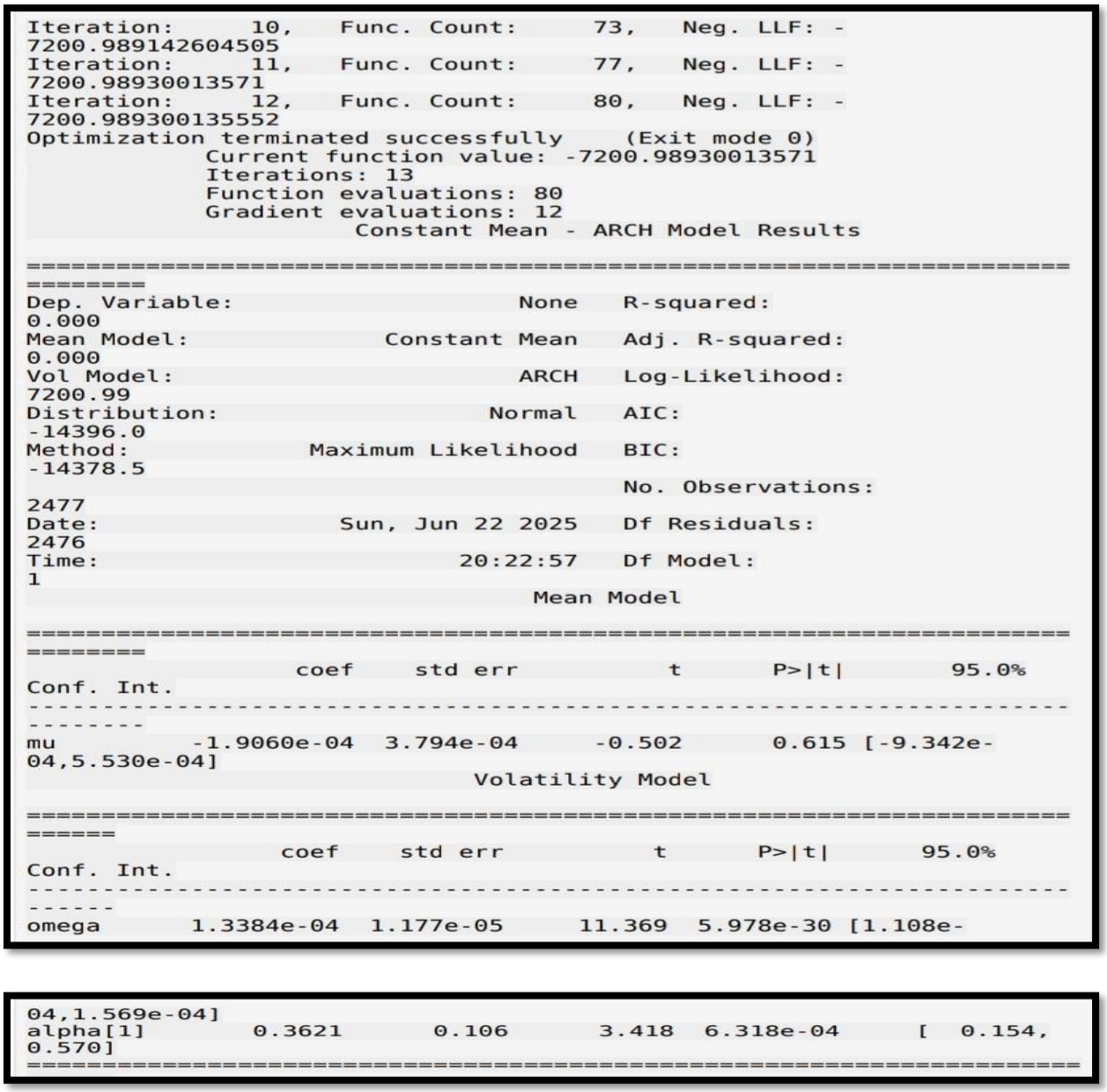
Based on these ACF and PACF plots we will fit a ARCH(1) model and check for the significance of the coefficients of appropriate lags.

```
# Fit ARCH(1) model
arch_mod = arch_model(residuals, vol='ARCH', p=1)
arch_result = arch_mod.fit()
print(arch_result.summary())
```

Code snippet for fitting ARCH model

Here we take p=1 so as to check the significance of alpha 1 also

```
Iteration:      1,   Func. Count:      5,   Neg. LLF:
33766106427.13344
Iteration:      2,   Func. Count:     16,   Neg. LLF:
768894231.8338449
Iteration:      3,   Func. Count:     27,   Neg. LLF: -
1797.4975570060653
Iteration:      4,   Func. Count:     35,   Neg. LLF:
911283.4351407352
Iteration:      5,   Func. Count:     43,   Neg. LLF: -
4777.5744237875715
Iteration:      6,   Func. Count:     50,   Neg. LLF:
963.6514876856762
Iteration:      7,   Func. Count:     58,   Neg. LLF: -
7136.7244190467945
Iteration:      8,   Func. Count:     63,   Neg. LLF: -
7200.004216970702
Iteration:      9,   Func. Count:     68,   Neg. LLF: -
7035.510837931333
```



```
Iteration:     10,   Func. Count:     73,   Neg. LLF: -
7200.989142604505
Iteration:     11,   Func. Count:     77,   Neg. LLF: -
7200.98930013571
Iteration:     12,   Func. Count:     80,   Neg. LLF: -
7200.989300135552
Optimization terminated successfully    (Exit mode 0)
            Current function value: -7200.98930013571
            Iterations: 13
            Function evaluations: 80
            Gradient evaluations: 12
                     Constant Mean - ARCH Model Results

==============================================================================
========
Dep. Variable:                       None   R-squared:
0.000
Mean Model:               Constant Mean   Adj. R-squared:
0.000
Vol Model:                         ARCH   Log-Likelihood:
7200.99
Distribution:                    Normal   AIC:
-14396.0
Method:             Maximum Likelihood   BIC:
-14378.5
                                         No. Observations:
2477
Date:                  Sun, Jun 22 2025   Df Residuals:
2476
Time:                          20:22:57   Df Model:
1
                                  Mean Model

==============================================================================
========
                 coef    std err          t      P>|t|        95.0%
Conf. Int.
------------------------------------------------------------------------------
--------
mu        -1.9060e-04  3.794e-04     -0.502      0.615 [-9.342e-
04,5.530e-04]
                              Volatility Model

==============================================================================
======
                 coef    std err          t      P>|t|       95.0%
Conf. Int.
------------------------------------------------------------------------------
------
omega      1.3384e-04  1.177e-05     11.369  5.978e-30 [1.108e-
04,1.569e-04]
alpha[1]       0.3621      0.106      3.418  6.318e-04      [  0.154,
0.570]
==============================================================================
```

Output showing significance of alpha

Here we can see that the confidence interval and p values are significant up to order 1 thus we can say the ARCH(1) is the appropriate model to be fitted.

Hence the coefficients are significant and we can reject the null hypothesis that the ARCH effect doesn't exist. Thus by accepting the alternate hypothesis we conclude that volatility is present (alpha= 0.3621) in the daily BANK NIFTY stock indices. Here the volatility persistence is 0.36, which indicates low volatility and thus we go for GARCH(1,1) model.

GARCH Model

The GARCH (Generalized Autoregressive Conditional Heteroskedasticity) model is used to model and forecast time-varying volatility in financial time series, such as stock returns. It captures volatility clustering, where periods of high volatility tend to be followed by high volatility, and vice versa.

1. Basic Setup

Let r_t be the return at time t, modelled as: r_t=μ+ϵ_t

Where:

- μ is the mean return (can be constant or time-varying),
- ϵ_t is the residual (shock) at time t,
- ϵ_t= $z_t\sqrt{h_t}$,
- z_t∼N(0,1) [standard normal],
- h_t is the conditional variance at time t.

2. GARCH(p, q) Model

The conditional variance h_t evolves as:

$$h_t = \omega + \sum_{i=1}^{q} \alpha_i {\epsilon_{t-i}}^2 + \sum_{j=1}^{p} \beta_j h_{t-j}$$

Where:

- ω>0, α_i≥0, β_j≥0,
- α_i measures the impact of past shocks (ARCH terms),
- β_j measures the persistence of past variances (GARCH terms).

3. GARCH(1,1): The Most Common Form

$$h_t = \omega + \alpha_1 {\epsilon_{t-1}}^2 + \beta_1 h_{t-1}$$

This simple form explains current volatility h_t as a function of:

- A constant ω,
- The previous squared return shock ${\epsilon_{t-1}}^2$,
- The previous variance h_{t-1}.

4. Conditions for Stationarity

To ensure that variance is finite and the process is stationary:

$\alpha_1+\beta_1<1$

If $\alpha_1+\beta_1\approx1$, the process is highly persistent and reacts slowly to shocks.

5. Application in Stock Forecasting

- Direct price prediction is not the goal; GARCH models forecast volatility, which is crucial for:
  - Option pricing,
  - Value at Risk (VaR),
  - Risk management,
  - Trading strategy development.

By accurately modelling the changing variance of returns, GARCH model allows better anticipation of risk and market behaviour.

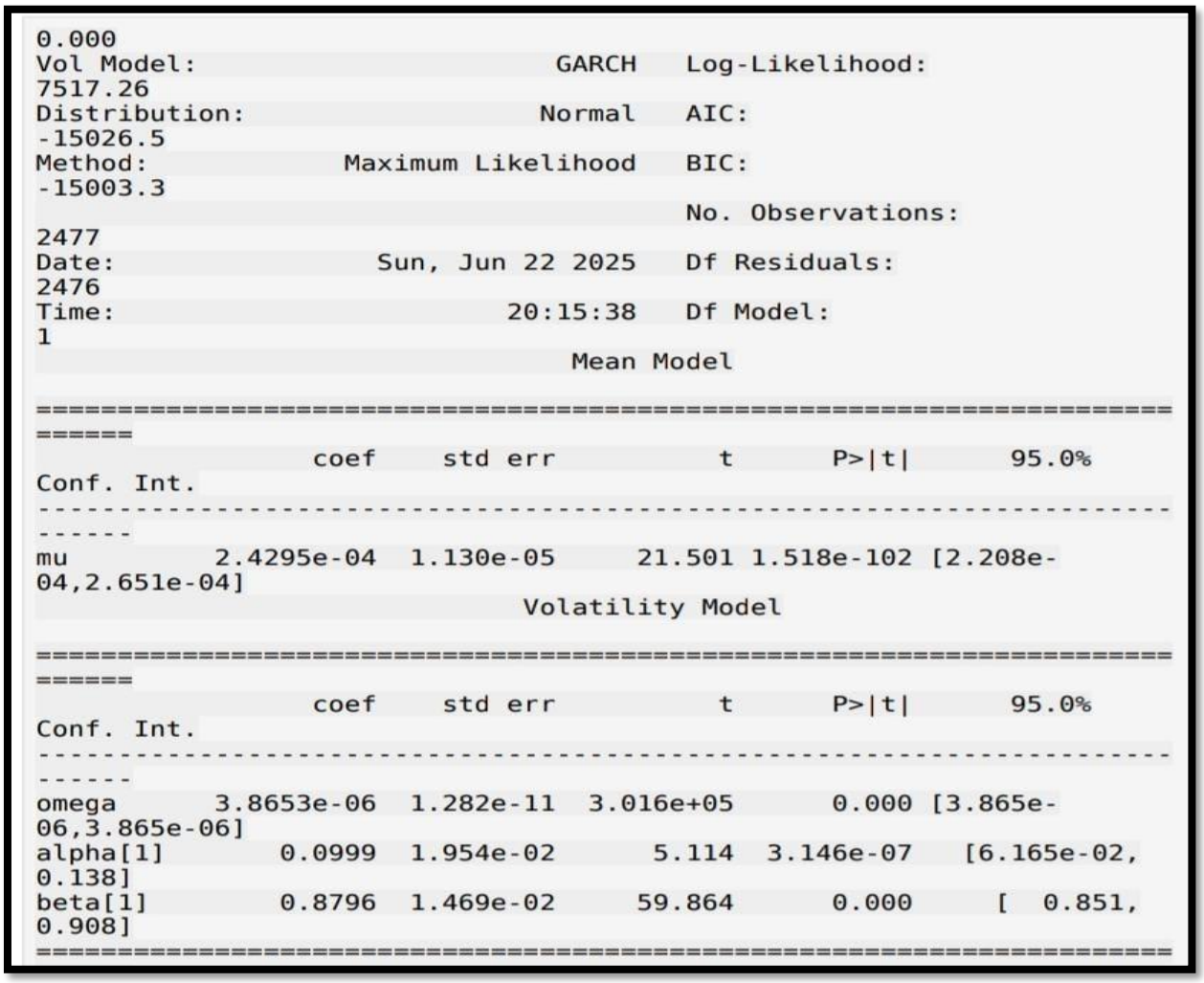
ARIMA-GARCH Model

The ARIMA-GARCH model is one model in which the variance of the error term of the ARIMA model follows a GARCH process. We consider the squared residuals after fitting ARIMA(0,0,0) to our data , i.e. we consider the GARCH(1,1) model as ARCH model has only volatility persistence 0.36.

```
from arch import arch_model

garch_model = arch_model(residuals, vol='GARCH', p=1, q=1)
garch_result = garch_model.fit()
print(garch_result.summary())
```

Code snippet for fitting GARCH model

```
Iteration:      1,   Func. Count:      6,   Neg. LLF:
13558246769.530647
Iteration:      2,   Func. Count:     19,   Neg. LLF:
1418514630.3995838
Iteration:      3,   Func. Count:     28,   Neg. LLF:
1058616.3149895747
Optimization terminated successfully    (Exit mode 0)
            Current function value: -7517.256746918818
            Iterations: 3
            Function evaluations: 36
            Gradient evaluations: 3
                      Constant Mean - GARCH Model Results

==============================================================
========
Dep. Variable:                       None   R-squared:
0.000
Mean Model:                 Constant Mean   Adj. R-squared:
```



```
0.000
Vol Model:                           GARCH   Log-Likelihood:
7517.26
Distribution:                       Normal   AIC:
-15026.5
Method:              Maximum Likelihood      BIC:
-15003.3
                                             No. Observations:
2477
Date:                  Sun, Jun 22 2025      Df Residuals:
2476
Time:                         20:15:38       Df Model:
1
                                  Mean Model

============================================================================
======
                 coef    std err          t      P>|t|      95.0%
Conf. Int.
----------------------------------------------------------------------------
------
mu         2.4295e-04  1.130e-05     21.501 1.518e-102 [2.208e-
04,2.651e-04]
                               Volatility Model

============================================================================
======
                 coef    std err          t      P>|t|      95.0%
Conf. Int.
----------------------------------------------------------------------------
------
omega      3.8653e-06  1.282e-11  3.016e+05      0.000 [3.865e-
06,3.865e-06]
alpha[1]       0.0999  1.954e-02      5.114  3.146e-07   [6.165e-02,
0.138]
beta[1]        0.8796  1.469e-02     59.864      0.000      [  0.851,
0.908]
============================================================================
```

Output showing significance of alpha & beta

We plot the residuals to check whether it is stationary in order to fit the GARCH Model. From the above plot, we can say that the residuals are stationary and volatility exists as the variance of the residuals are increasing and decreasing in nature.

The detection of the GARCH effect in a time series is a test of serial independence applied to the serially uncorrelated fitting error of some model, in our case ARIMA model. We have assumed that linear serial dependence inside the original series is removed with an efficient model. Hence, any further serial dependence must be due to some nonlinear mechanism which has not been

Lagrange Multiplier Test In this testing process,

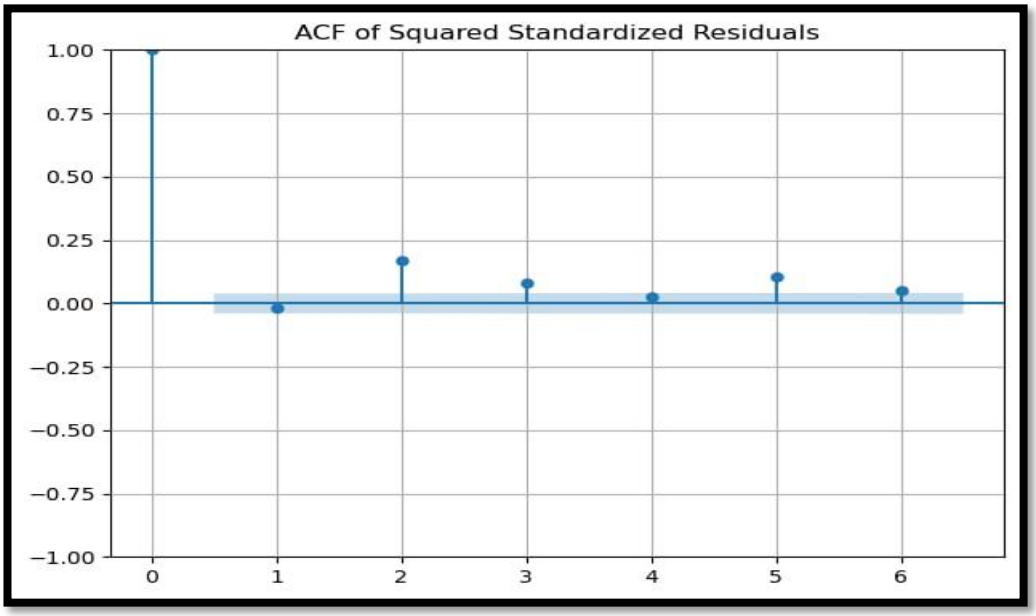
H_0 : GARCH-Effect is not present

H_1 : GARCH-Effect is present

Thus here the p-value is 3.146e-07< 0.05. Hence at 5% level of significance we reject the null hypothesis and conclude that the squared residual series contains GARCH-effect.

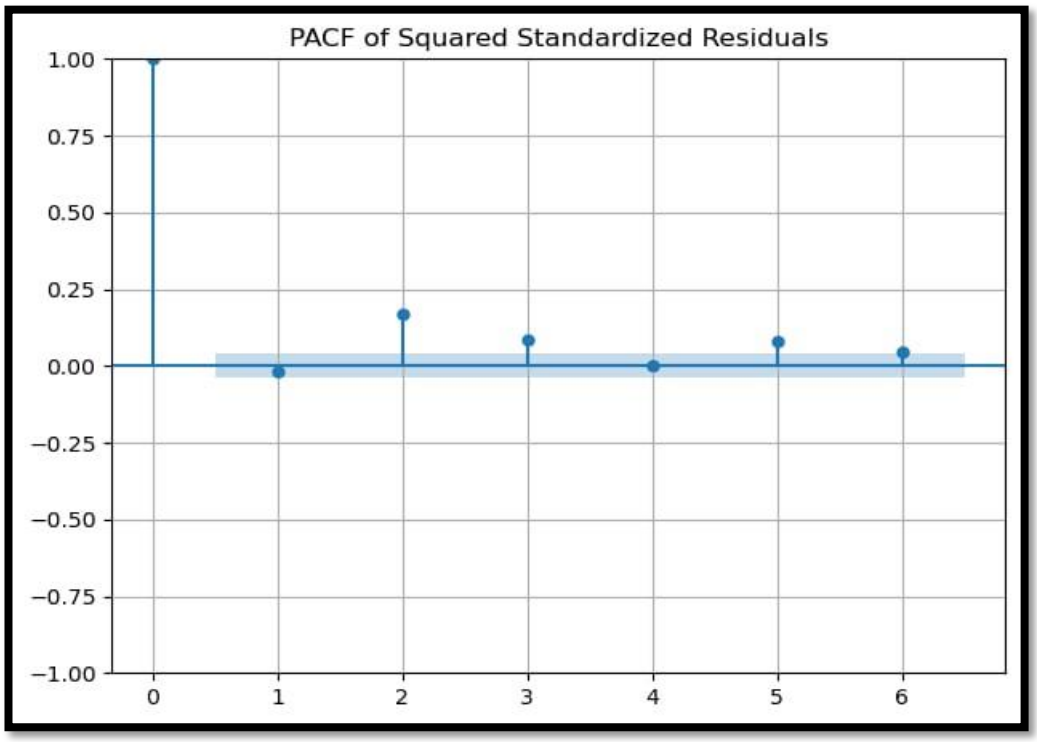
Now we plot the ACF plot of the squared residuals with lag =6. We plot the ACF plot to check if any correlation is out of the range, if there are significant spikes we can say that correlation of squared residual exists however if there are no spikes we can say that no correlations are significant hence the series is white noise.
For the above residuals the correlogram containing the ACF is given below:

```
from arch import arch_model
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import matplotlib.pyplot as plt
garch_model = arch_model(residuals, vol='GARCH', p=1, q=1)
garch_result = garch_model.fit()

# Get standardized residuals
std_resid = residuals / arch_result.conditional_volatility
squared_resid = std_resid ** 2

# Plot ACF & PACF of squared residuals
plot_acf(squared_resid, lags=6)
plt.title('ACF of Squared Standardized Residuals')
plt.grid(True)
plt.tight_layout()
plt.show()
```



ACF plot of squared residuals that there are significant number of spikes (upto order 1), at lag 6, hence we can say that correlation of squared residuals exists, therefore volatility exists.
Again we plot the PACF plot for the squared residuals. The correlogram containing the same is given below-

```
plot_pacf(squared_resid, lags=6)
plt.title('PACF of Squared Standardized Residuals')
plt.grid(True)
plt.tight_layout()
plt.show()
```

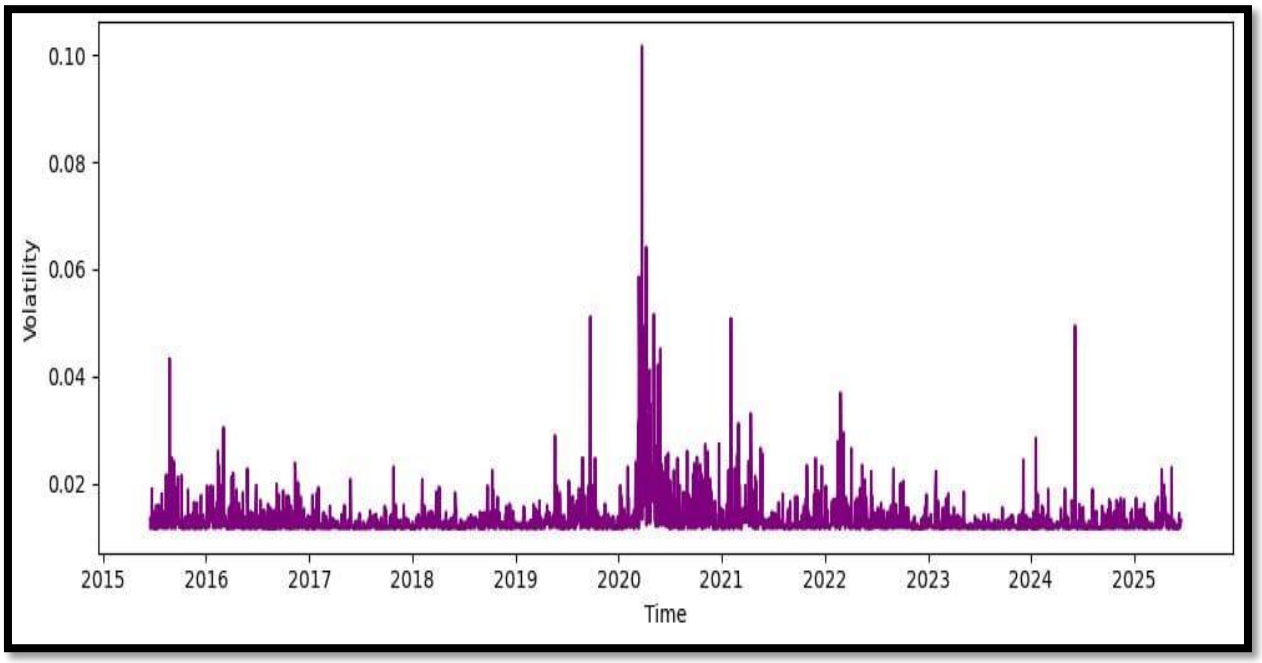


In the PACF plot, spikes up to order 1.

Based on these ACF and PACF plots we will fit a GARCH(1,1) model and check for the significance of the coefficients of appropriate lags.
Here we take p=1and q=1 so as to check the significance of alpha 1 and beta 1.

Here we can see that the confidence interval and p values are significant up to order 1 thus we can say the GARCH(1,1) is the appropriate model to be fitted.

Hence the coefficients are significant and we can reject the null hypothesis that the GARCH effect doesn't exist. Thus by failing to reject the null hypothesis we conclude that volatility is present ($\boldsymbol{\alpha_1}$= 0.0999, $\boldsymbol{\beta_1}$=0.8796, $\boldsymbol{\alpha_1 + \beta_1 = 0.98}$) in the daily BANK NIFTY stock indices. Here the volatility persistence is 0.98, which indicates high volatility and thus we can conclude that the GARCH(1,1) model is a better fit than the ARCH model where volatility persistence was 0.36. The GARCH model also had a Higher log-likelihood, Lower AIC and a realistic volatility dynamics as compared to an ARCH model. This indicates in total that the BANK NIFTY stock data is highly volatile in nature.



Volatility of GARCH(1,1) model

Conclusion

In this work our central purpose was to predict the closing stock prices of BANK NIFTY Stock index. Using appropriate methods we have successfully transformed the non-stationary raw data into a stationary one. After that, under the assumption of constant error variance we have identified that the ARIMA(0,1,1) model is best to describe the mean nature of the stock price. Later, we detected the presence of ARCH-effect in the ARIMA(0,1,1) residuals through Engle's Lagrange Multiplier test. We saw that the data has low volatility as the value of alpha was 0.36. But afterwards we tested the GARCH(1,1) model where we found a higher volatility persistence (alpha=0.98) which concluded that there is a high volatility present in the BANK NIFTY stock data.

References

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

2. 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.

3. Osemwenkhae J.E., Eguasa E.B. and Iduseri A, "Hybrid of ARIMA-ARCH Modelling of Daily Share Price Data of Okomuc Oil Plc in Nigeria Hybrid of ARIMA-ARCH Modelling of Daily Share Price Data of Okomu Oil Plc in Nigeria", Journal of the Nigerian Association of Mathematical Physics Volume36, No. 2 (July, 2016), pp 163 – 168.

4. 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.

5. The Use of ARCH and GARCH Models for Estimating and Forecasting Volatility-ruzgar-kale.