Time Series Analysis of Stock Market

Project report – ISEN 613: Engineering Data Analysis

Submitted by- Team 9

Name	UIN	Contribution
Aditya Sharma	426001491	20%
Chetan Shelke	826001716	20%
Kiran Ramakrishnan	125008669	20%
Laxmi Prasad Sai Yallasari	226002267	20%
Kriti Verma	226000292	20%

Contents

Project Background and Motivation

'Beating the stock market' has been some of the most interesting applications of data analytics. Malkiel theorized that "a blindfolded monkey throwing darts at a newspaper's financial pages could select a portfolio that would do just as well as one carefully selected by experts." Malkiel suggests that it is not possible predict the volatility of the stock market before-hand [Efficient Market Hypothesis] and hence Random walk model is the most ideal fit.

Though Efficient Market Hypothesis finds favor among many noted academicians, many critics have pointed out that there are lots of instances where analysts have performed far better than the random walk model [considered base model]. In fact, there is a whole industry built around it. Prediction methods fall under two major categories: Fundamental Analysis and Technical Analysis.

Fundamental Analysis: It considers the intrinsic value of the stock, rather than the value at which it is currently traded. The fundamentals of company include Quantitative: Measurable numeric characteristics such as profits, revenues, cash and other assets and Qualitative parameters: Quality of management, employee morale etc., it is important to combine both the parameters to make an assessment.

Technical Analysis: In this type of analysis we only consider past trends and disregard other parameters to predict the stock market. The stock market forecasting is a Time Series problem. Three broad classes of models in Time series are Auto-regressive models(AR), moving average models (MV) and Integrated models. Combinations of these models give rise to ARIMA and ARMA models. There are another class of models for non-linearity (heteroscedastic models) these include GARCH, EGARCH, TARCH etc.,

Problem Statement

Given the past stock data, can we buy a stock and make profit on it based on a time series forecast?

Numerous attempts have been made to answer this question. But, despite years of stock data and research in technical analysis a clear winning approach has not been identified yet.

Akgiray is the first person to use GARCH to model stock market. He noted that GARCH models are superior to other forecasting models like Random Walk [ARIMA (0 1 0)], Moving Average [ARIMA (0 0 x)] and Exponential Smoothing [ARIMA (0 1 1)]. This is also supported by Brailsford and FAff (1996) who used a variation of GARCH model to predict Australian Stock market, Franses and Van Dijk who used GARCH model to predict stock market in Spain, Italy, Germany and Sweden also showed GARCH model produced the best forecast. However, Tse(1996) shows superiority of moving average model over GARCH model. Pan and Zheng also prefer to use simplistic MA models to predict chinese stock market. Hence, to forecast the stock market, simple models like Moving average to more complex GARCH model is continued to be used.

The goal of our project is to investigate ARIMA and GARCH based models and figure out which gives the best results for short term forecasting.

Scope

As we mentioned we restricted ourselves to technical analysis of the past stock prices of company 3M listed in S&P 500 index. We elected to use two popular Time Series analysis technique to carry out the analysis i.e. GARCH and ARIMA.

Objectives

- Implement Time Series Analysis to Predict Stock Market and recommend the best model to carry out forecasting.
- The literature is divided regards to the best time series model to use on the dataset. So our objective was to implement two of the most popular approaches to solve time series problems that is GARCH and ARIMA models in one of the Stocks in S&P data so that we can take a stance about which researcher's conclusion is most accurate.
- To Develop Strong understanding about time series analysis techniques.
- Be able to effectively discriminate between different models, and build an intuition about which model to use.
- Get familiarized with Stock market data analysis in terms of its scope, limitations and complexities.

Literature Review

Stock Price prediction using ARIMA model Ayodele A. Adebiyi et al.

The Author discusses ARIMA analysis on NYSE data and Nigerian Stock exchange. The Author took up Nokia stock for analysis. Plotting the ACF we can observe that the ACF dies very slowly which means that the data is non-stationary. Differencing was used to make it stationary. Dickey Fuller test was used to confirm that the data is stationary. Several ARIMA models were experimented and ideal ARIMA was chosen to be ARIMA(2,1,0) . The model was tested for white noise and then determined if the model is good. ARIMA is considered to be robust and efficient in time series forecasting applications.. The best ARIMA model was be chosen based on:

- Relatively small Baysian Information Criteria
- Relatively small error in regression
- Relatively small adjested R square
- ACF and PACF shows no pattern

The Author concludes that ARIMA outperforms more complex models in short term forecasting. ARIMA can compete well with emerging techniques.

Predicting Stock market returns using GARCH model Arowolo W.B. et al.,

This work studies GARCH model applied to daily closing prices of Zenith Bank Plc trade in Nigerian Stock Exchange from April 2005 to December 2009. GARCH model has the ability to capture volatility in stock market. In the model the P and Q values were chosen and error terms were identified using AIC and BIC and Diagnostic tests were done to assess goodness of fit. The author argues that GARCH is the best of all the available models to capture the time varying volatility by applying it to nigerian stock market. The stock market is leptokurtic. He finds that negative shocks have more influence on volatility than positive influence. The optimal values of P and Q in GARCH(p,q) model will depend on the location, type of data and model order selected techniques used.

Use of GARCH and ARCH models in Applied Econometrics, Robert Engle

Generally we used residuals to check for fit of a model. But now days variance has become an important metric in prediction. While there are a lot of models to use mean, ARCH models were the first ones to use variance. Engle proposed to use weights to use past variance to predict future variance. In GARCH model the reducing weight of any term never goes to zero. It is noted that instead of using equal weights, declining weights are more accurate. In ARCH(P,Q), P refers to the number of autoregressive lag terms in the equation.

Project Approach

Step 1

Load relevant libraries

As the problem in consideration for this project is Time-Series in nature, install all the libraries facilitating analysis of such data-form

Step 2

Read the data:

Read or import the data from excel using read.excel() or using read.csv() from CSV into R Studio. command was used to import the data into the "R-Studio"

Step 3

Data overview

Explore the data using R commands such as *summary()*, *str()*, *FinTS.stats(), and jarque.bera.test().*

Step 4

Data Cleaning:

Replace blank fields with zeros and convert the Date column into date format. We will use *tsclean() and ts()* function of R to create a time series objects.

tsclean() is a convenient method for outlier removal and inputting missing values

Step 5.

Plot the data against time:

Create time series objects and plot factors against time.

Observe the graph and look whether the factor displays any discernable trend

Step 6

Check for Stationarity:

Run the Dickey-Fuller test to determine the stationarity and conclude whether the data is stationary or not using the p-Value

Step 7

Make the data stationery:

Apply various orders of differencing and transformations like log and square-root on the data to make the series stationary

Step 8

Plot Correlogram:

Examine the correlogram and partial correlogram of the stationary time series using the *acf()* and *pacf()* functions in R to determine the parameters (p,d,q) for ARIMA model

Step 9

Select the candidate model:

Run different ARIMA models and compute AICs

Compare AIC values to determine the best ARIMA model.

Step 10

Check for an Arch effect:

Run ArchTest to check if data has any arch effecti.e. if p-value is less than or more than 0.05

p-value is less than 0.05 and thus has an arch effect

Step 11

GARCH Modeling:

Fit the GARCH model using garchFit() function in R for different p and q parameters

Select the best GARCH model using the AIC and SIC values

Best model has least AIC. Also, SIC is same as AIC.

Step 12

Run diagnostic plots:

Plot acf and pacf of residuals for selected best GARCH model

Residuals shows no discernible pattern and model captures all the information in the data

Step 13

Predict:

Predict the output using predict function

Step 14

Compare and conclude:

Shared above was the sequence of steps exercised, to arrive at the best ARIMA mode. Though lot of trial and error is

Time Series Analysis – A brief Overview

ARIMA model: ARIMA (Autoregressive Integrated Moving Average model) is one of the most extensively used time series analysis technique. ARIMA models are used for forecasting a time series that can be made stationary. Exponential Smoothing and Random-walk are special cases of ARIMA models. In ARIMA (p,d,q) p is the number of autoregressive terms, d is the number of non-seasonal differences needed for stationarity and q is the number of lagged forecast errors in the prediction equation.

ARIMA (0,1,0) model (random walk) It is the simplest model. The prediction equation of this model is

$$\hat{Y}_t = \mu + Y_{t-1}$$

$Y_{(t-1)}$ is the previous value

$Y_{(t)}$ is the current value

e is the error is the random error

ARIMA (1,1,0) model Differenced first order auto-regression model: We use this model if errors in the random walk model are auto correlated.

$$\hat{Y}_t - Y_{t-1} = \mu + \phi_1(Y_{t-1} - Y_{t-2})$$

$$\hat{Y}_t - Y_{t-1} = \mu$$

which can be rearranged to

$$\hat{Y}_t = \mu + Y_{t-1} + \phi_1 (Y_{t-1} - Y_{t-2})$$

ARIMA(0,1,1) model simple exponential smoothing model: In this model it uses the average of the previous values instead of using just one value, so that noise is averaged out

$$\hat{Y}_t = \hat{Y}_{t-1} + \alpha e_{t-1}$$

Because $e_{t-1} = Y_{t-1} - \hat{Y}_{t-1}$ by definition, this can be rewritten as:

$$\hat{Y}_t = Y_{t-1} - (1-\alpha)e_{t-1}$$

$$= Y_{t-1} - \theta_1 e_{t-1}$$

ARIMA(0,2,1) model Linear exponential smoothing: It uses two non-seasonal differences in conjunction with MA terms.

$$\hat{Y}_t - 2Y_{t-1} + Y_{t-2} = - \theta_1 e_{t-1} - \theta_2 e_{t-2}$$

which can be rearranged as:

$$\hat{Y}_t = 2 Y_{t-1} - Y_{t-2} - \theta_1 e_{t-1} - \theta_2 e_{t-2}$$

Theta1 and theta2 are the MA(1) and MA(2) coefficients

GARCH model: Generalized Autoregressive Conditional Heteroscedastic model is used to model dataset with volatile variance. In ARCH process the variance is a function of variances in the previous time periods.

If an autoregressive moving average model (ARMA) model is assumed for the error variance, the model is a generalized autoregressive conditional heteroskedasticity (GARCH) model.

In that case, the GARCH (p, q) model (where p is the order of the GARCH terms σ^2 and q is the order of the ARCH terms $\boldsymbol{\varepsilon^2}$, following the notation of original paper, is given by

$$y_t = x_t' b + \epsilon_t$$

$$\epsilon_t | \psi_t \sim \mathcal{N}(0, \sigma_t^2)$$

$$\sigma_t^2 = \omega + \alpha_1 \epsilon_{t-1}^2 + \cdots + \alpha_q \epsilon_{t-q}^2 + \beta_1 \sigma_{t-1}^2 + \cdots + \beta_p \sigma_{t-p}^2 = \omega + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{i=1}^{p} \beta_i \sigma_{t-i}^2$$

GARCH (*p*, *q*) model specification

The lag length *p* of a GARCH (*p*, *q*) process is established in three steps:

1. Estimate the best fitting AR (*q*) model

$$y_t = a_0 + a_1 y_{t-1} + \cdots + a_q y_{t-q} + \epsilon_t = a_0 + \sum_{i=1}^{q} a_i y_{t-i} + \epsilon_t.$$

2. Compute and plot autocorrelation of $\boldsymbol{\varepsilon^2}$ by

$$\rho = \frac{\sum_{t=i+1}^{T} (\hat{\epsilon}_t^2 - \hat{\sigma}_t^2)(\hat{\epsilon}_{t-1}^2 - \hat{\sigma}_{t-1}^2)}{\sum_{t=1}^{T} (\hat{\epsilon}_t^2 - \hat{\sigma}_t^2)^2}$$

3. The asymptotic that is for large samples, standard deviation of is $\rho(i)$ is $1/\sqrt{T}$. Individual values that are larger than this indicate GARCH errors. To estimate the total number of lags, use the Ljung-Box test until the value of these are less than, say, 10% significant. The Ljung-Box Q-statistic follows χ^2 distribution with n degrees of freedom if the squared residuals $\boldsymbol{\varepsilon_t^2}$ are uncorrelated. It is recommended to consider up to T/4 values of n. The null hypothesis states that there are no ARCH or GARCH errors. Rejecting the null thus means that such errors exist in the conditional variance.

Important Diagnostic Tests and Plots

As already mentioned before, that it is an important requirement that, we make the data stationary, either by transforming, or, differencing original or transformed data to order convenient, to achieve constant mean and variance. Though many rely on visual judgement of data plotted against time/time index, to decide whether or not the time-series data, to be analyzed has been stationarized or not. However, we have statistical means of confirming, if the data, we are inferring to be stationary, is statistically stationary or not. Dickey-Fuller test, explained below is one such test.

Dickey-Fuller test: It is used to test for stationarity, testing the null hypotheses for presence unit root in autoregressive model. The logic behind it is that if the root is less than one it will converge back to a specific value thereby making it stationary.

As, explained above, that ARIMA, model needs following listed parameters, described using following alphabets, in most literature, referred to:

P : Number of lagged observations included in the model

q : Number of error terms or the size of the moving average window.

d : Order of differencing to make data stationary.

A conventional method of finding parameters (p,q) is to examine the ACF (**A**uto-**C**orrelation **F**unction) and PACF (**P**artial **A**uto-**C**orrelation **F**unction), to identify the "q" and "p" respectively.

ACF: It is the similarity of the observations as a function of timelag. It is used to identify the presence of periodic signals. It is basically correlations coefficient of the same variable at two different times.

PACF: This function gives the partial Autocorrelations with its own lagged values. It cuts the influence of auto-correlations between the lagged values.

Briefly, ACF yields us correlation between X_t and X_{t-k} , where X_t is value of time series variable at time instance marked "t", and, X_{t-k} is value "k" time instances before "t". Of-course, all the values between X_t and X_{t-k}, contribute to value that X_t has. PACF is influence of X_{t-k} on X_t, assuming that all the X_i, for $t-k+1 < i < t$, also influence X_t.

To estimate ***q*** and ***p,*** we just count the number of lag, which go beyond the threshold level of significance, in the direction of 1st lag. One of the PACF and ACF, has been quoted as example further in report, to demonstrate.

Implementation Plan

Steps in Univariate Time Series Analysis (For R)

The implementation of one of the ARIMA and GARCH model that we are implementing on the dataset has been discussed below. Since the implementation procedure for all the variants are similar.

Step 1. Load relevant libraries.

As the problem that was considered for this project was Time-Series in nature, all the libraries facilitating analysis of such data-form have been included. Listing few relevant time series packages:

a) "tseries" - Most commonly used time-series package, Has all the necessary commands for basic time series analysis, with ARIMA.
b) "rugarch" – For GARCH modelling.
c) "FinTS" – It is extensive econometric package, developed for performing tests like ARCH test, Jaque-Bera test
d) "forecast" – This is forecasting

Step 2. Read the data:

Read the data in Excel using read_excel command. As the data was available in ".csv" format ***read.csv(.)*** command was used to import the data into the "R-Studio"

Step 3. Data overview:

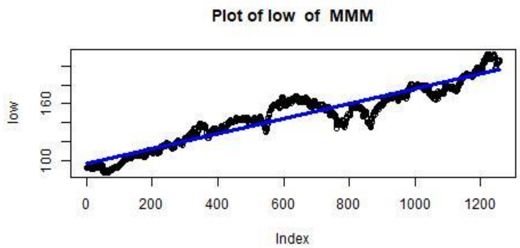
Use the ***summary()*** and ***str()*** commands to explore the data. Also ***FinTS.stats(),*** command was also another command, essentially with more attributes, summarizing data being analyzed.

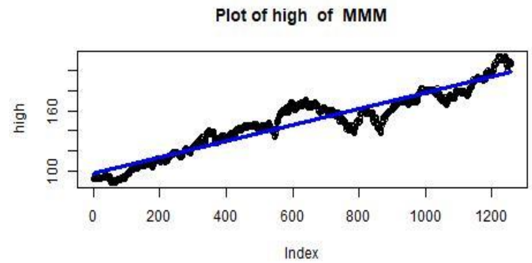
Step 4. Data Cleaning:

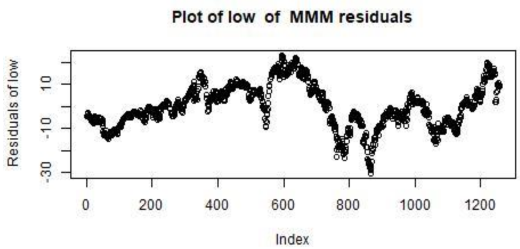
Selected the numeric columns and removed any rows containing blank fields

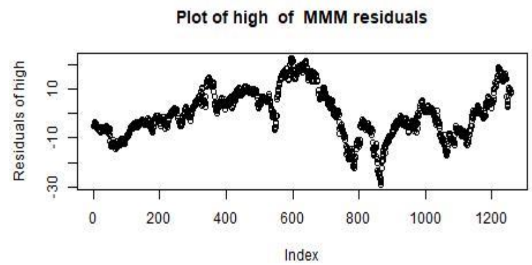
```
numeric_column = sapply(input, is.numeric)
rows_non_numeric = which(is.na(input[,numeric_column]))
input = input[-c(rows_non_numeric),]
```

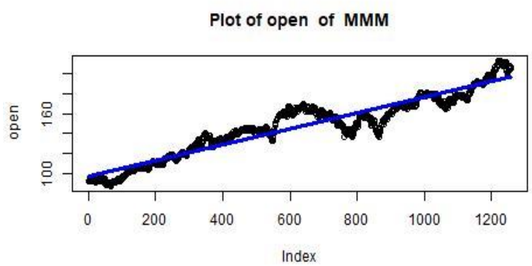
Step 5. Plot the data against time: Create time series objects and plot factors against time. Observe the graph and look whether the factor displays any discernable trend.

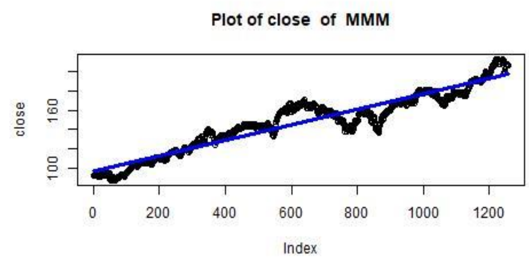


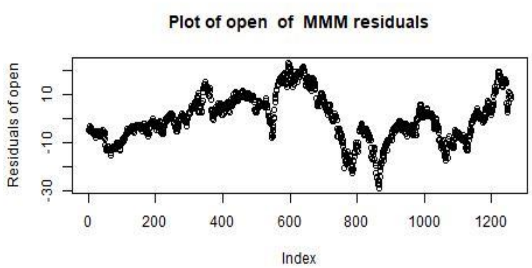


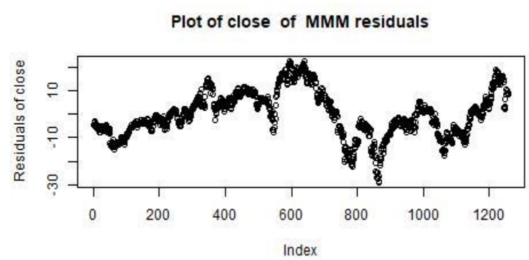


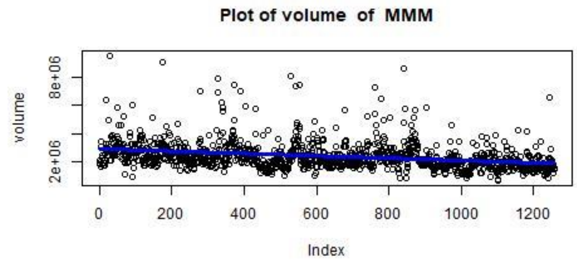


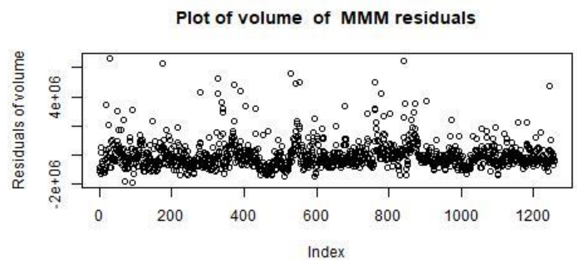












The above drawn plots are that of low, high, close, and open, of stock "MMM" that has been given to us. It is evident from the plot that the data has a discernable positive trend against time, whereas the volume doesn't exhibit any trend. Our hypothesis that low, high, close, and open are not stationery is further supported by the Dickey-Fuller test.

Step 6. Check for Stationarity:

Run the Dickey-Fuller test to determine the stationarity and conclude that data is not stationary if the p-Value is more than 0.05.

```
> adf.test_p_value
     [,1]           [,2]
[1,] "MMM _ open"   "0.407396459357783"
[2,] "MMM _ high"   "0.527402345359433"
[3,] "MMM _ low"    "0.418853473443304"
[4,] "MMM _ close"  "0.348741302888482"
[5,] "MMM _ volume" "0.01"
```

The p-value < 0.05 of volume supports the alternative hypothesis that **volume is stationery**. The p-value > 0.05 calls us to reject the alternative hypothesis, **low, high, open, and close are not stationery**.

Step 7. Make the data stationery:

We can apply various orders of differencing and transforms like log and square-root on the data to make the series stationary as observed in Dickey-Fuller test.

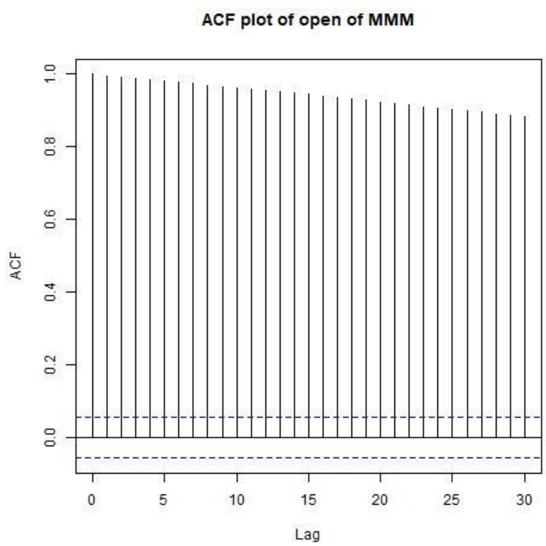
We were able to achieve stationarity with single differencing for the dataset we are analyzing.

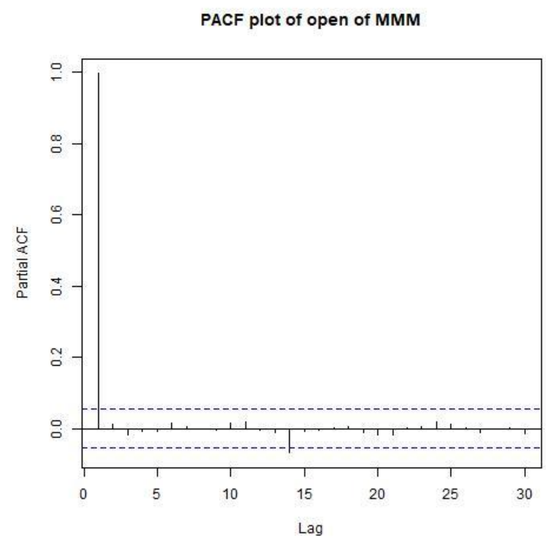
```
#Stationarizing the data by differencing and Plotting the same
#**************************************************************
order_of_differencing = 1
model_data_diff = Differencing(model_data,order_of_differencing)
```

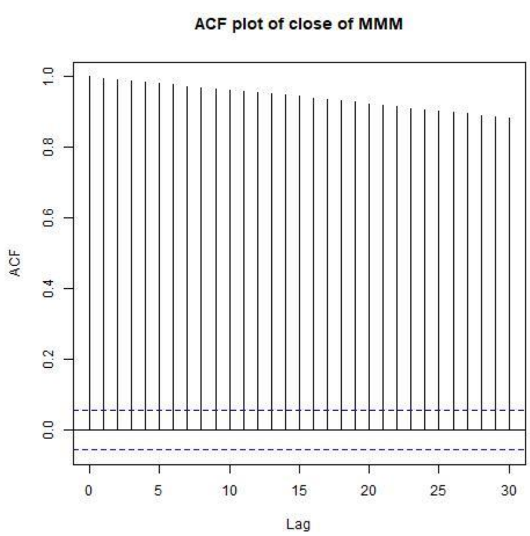
Step 8. Plot Correlogram:

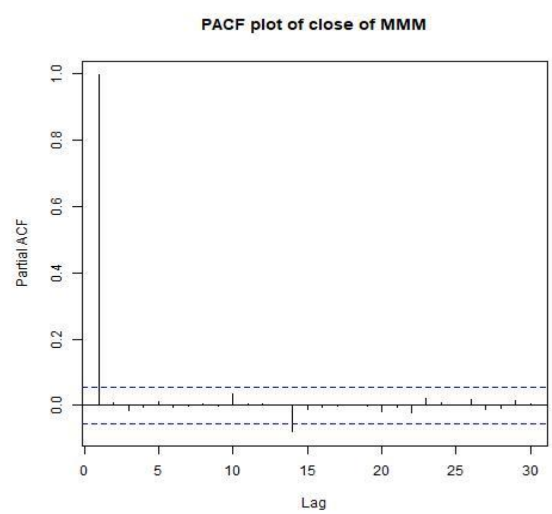
We examined the correlogram and partial correlogram of the stationary time series using the ***acf()*** and ***pacf()*** functions in R, respectively. This helps to determine the best ARIMA model parameters (p,d,q).

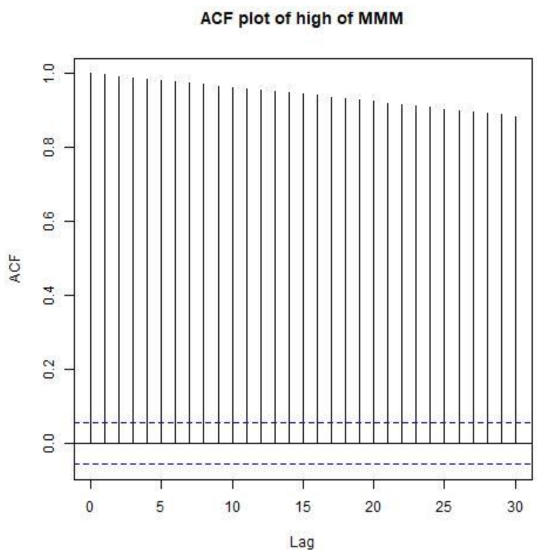
Below attached are the plots, concerning the data available in the data set. We have attached only the plots of "high", "close", "open", "low", and "Volume".

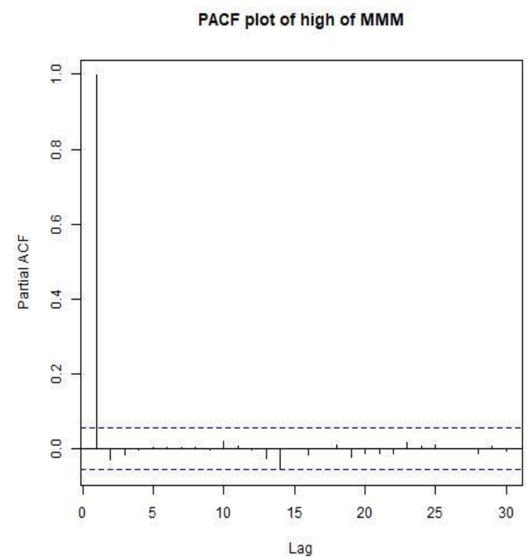


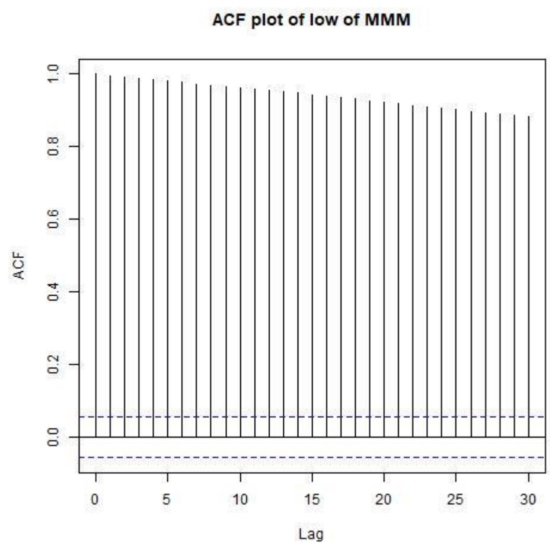


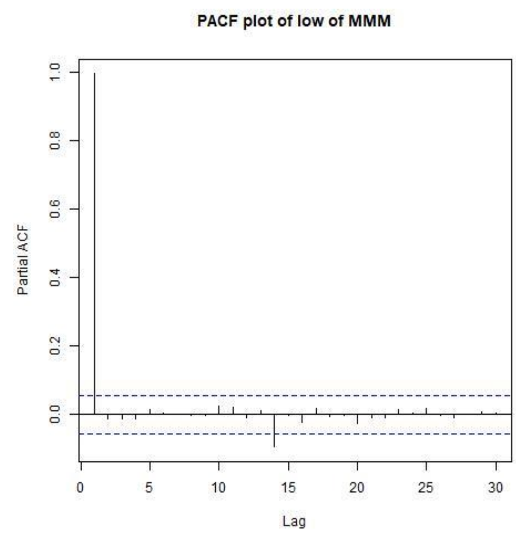


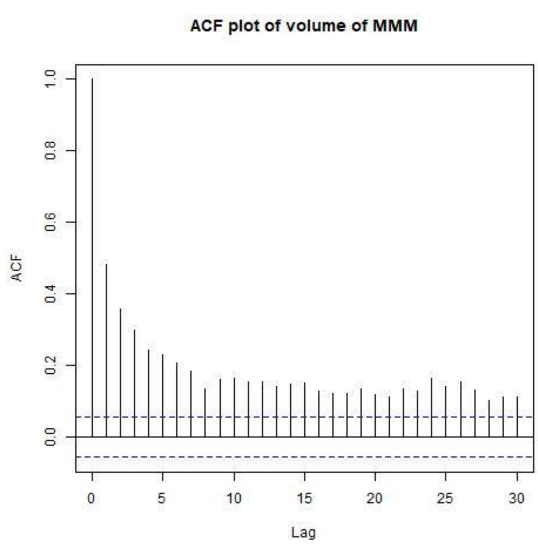


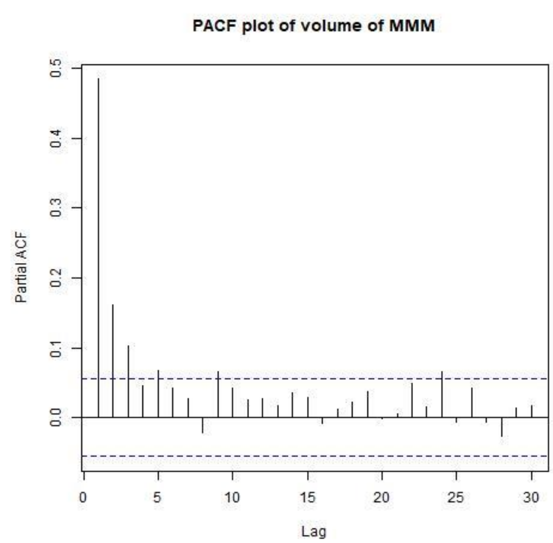

ACF plot of high of MMM
ACF
Lag


PACF plot of high of MMM
Partial ACF
Lag


ACF plot of low of MMM
ACF
Lag


PACF plot of low of MMM
Partial ACF
Lag


ACF plot of volume of MMM
ACF
Lag


PACF plot of volume of MMM
Partial ACF
Lag

From the above ACF plots we can see that for high, low, open, and close there is a gradual decrease in the auto correlation. This implies that the series are not stationary. The ACF plot of Volume shows sharp decline and hence it can be interpreted as stationery.

After inspecting the PACF terms we figure out which lag factors are contributing to the present value of the variable. The plots for low, high, open, and close signify that it needs to be differenced once. Whereas the volume term suggests that differencing more than first order maybe required.

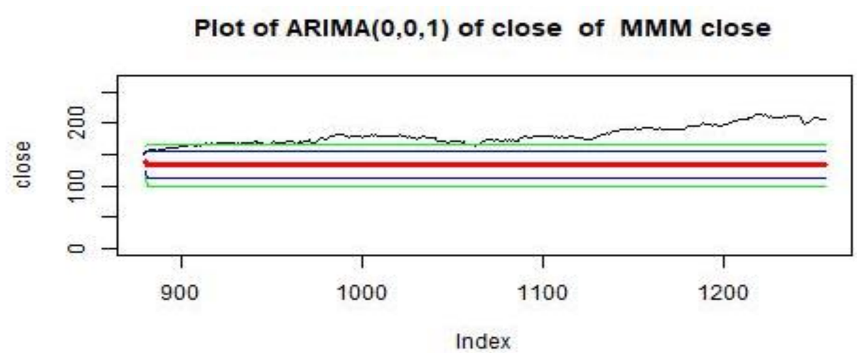
After going through some finance literature and a technical paper (*Stock Price Prediction Using the ARIMA Model Ayodele A. Adebiyi et al.,*), it was concluded that we should base our analysis on the close values only. As the price fluctuations in the market are based on speculation, prediction based on the closing price of the stock gives a comparatively better estimate of the overall stock price movement.

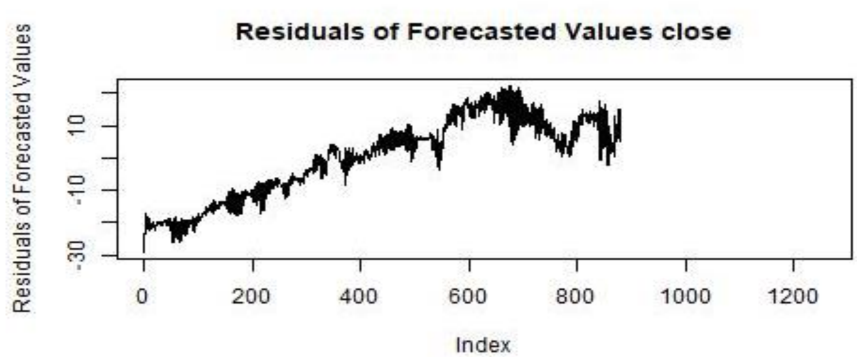
Step 9. Select the best ARIMA model:

Run different ARIMA models to compare the AIC value to determine the best model. Below are the forecasted plots of various permutations of ***(p,d,q)***, which have been earlier defined. In the following the graphs, the colors represent following:

a) Red color : Forecasted Value
b) Blue : Locus of values for 80% confidence interval
c) Green : Locus of values at 95% confidence interval
d) Black: Actual Values

The plots below, with parameters (0,0,1) can be interpreted as moving average of order “1”. As we can see from the plot and the residuals, that forecast is moving away as the time instance from the beginning increases. The other graphs, have been plotted to gain peek into behaviors of (p,d,q), and estimate to get better idea of choosing (p,d,q). The graphs given on table below are representative of some special cases of ARIMA.





After the model is developed it is verified if the residuals are uncorrelated, having normal distribution, and whether the variance is constant. Described below are some statistical diagnostic test used to verify the model**.**

Running Diagnostic Tests

a) Box-L-jung Test (Statistics for ARIMA residuals):
The residuals are plotted to check for the randomness. Following is the snippet of command that has been used

b) Plot ACG of residuals:
Plot the ACF of residuals to conclude whether the ARIMA model is adequate or not.

Compare results

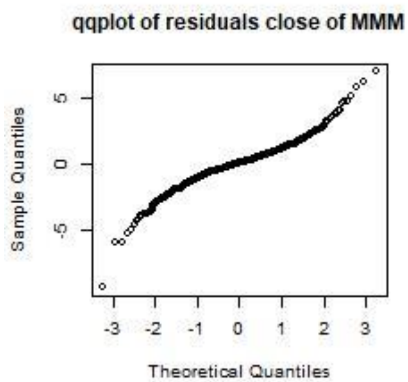
Shared above was the sequence of steps exercised, to arrive at the best ARIMA mode. Though lot of trial and error is required, there are simple commands at the best model directly (Auto.Arima()).

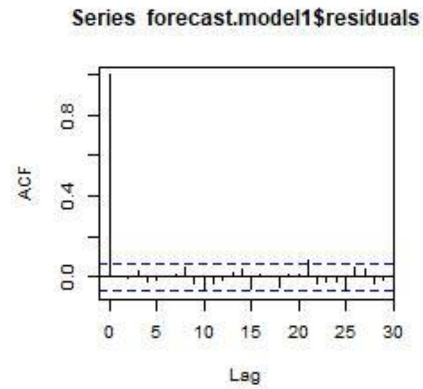
Below is the best model and its diagnostic.

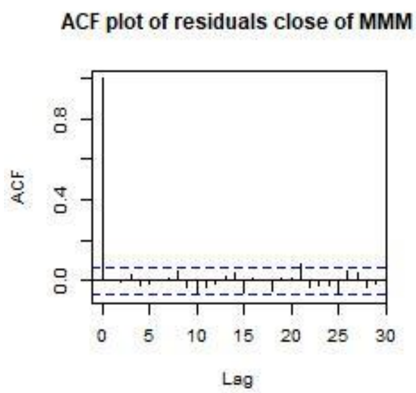
```
>   Box.test(forecast.model1$residuals^2,lag=20,type = "Ljung-Box")

        Box-Ljung test

data:  forecast.model1$residuals^2
X-squared = 60.359, df = 20, p-value = 6.266e-06
```

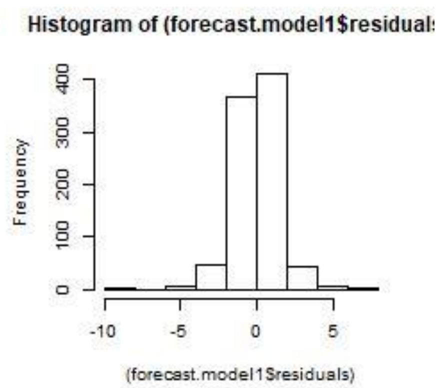
```
>   summary(arima.model_1)
Series: model_data[train, j]
ARIMA(0,1,2) with drift

Coefficients:
          ma1     ma2   drift
      -0.0647  0.0536  0.0684
s.e.   0.0338  0.0348  0.0472

sigma^2 estimated as 2.01:  log likelihood=-1550.79
AIC=3109.59   AICc=3109.63   BIC=3128.7

Training set error measures:
                       ME     RMSE       MAE          MPE      MAPE      MASE        ACF1
Training set 0.0001296408 1.414491 0.9889401 -0.001479627 0.7389068 0.9931366 0.001506192
```

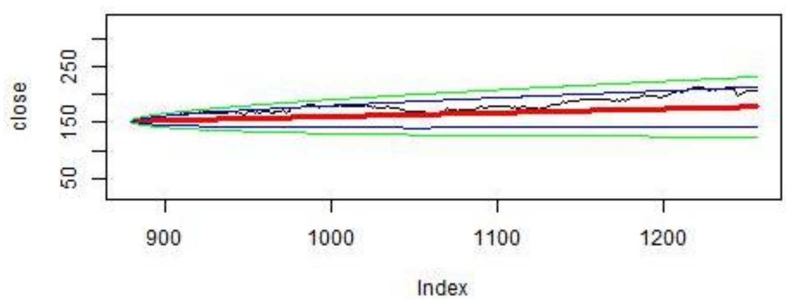
This model has the best AIC of all the models that we tried. Moreover, we can see from the plots titled "qqplot of residuals close of MMM", "Histogram" and "ACF plot of residuals close of MMM", that residual are uncorrelated and also, seem to follow normal distribution, which is a fundamental assumption when developing a model.

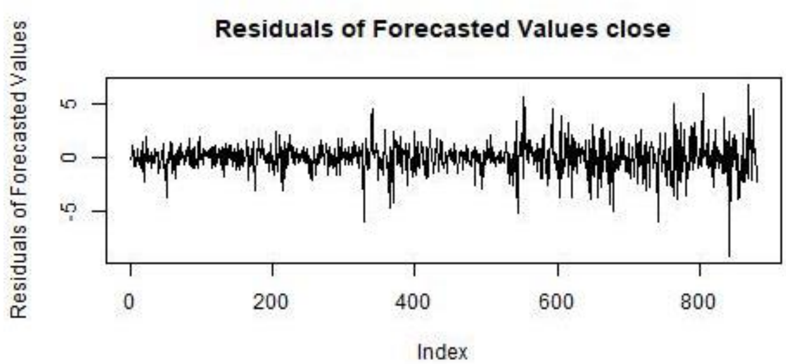

qqplot of residuals close of MMM
Sample Quantiles
Theoretical Quantiles


Series forecast.model1$residuals
ACF
Lag


ACF plot of residuals close of MMM
ACF
Lag


Histogram of (forecast.model1$residual:
Frequency
(forecast.model1$residuals)


close
Index


Residuals of Forecasted Values close
Residuals of Forecasted Values
Index

Comparison

Tabulated below are summary and diagnostic plots for the Naïve Model and Simple Exponential Smoothing (Special Cases of ARIMA models).

Comparing sample ARIMA models

<table>
<tr><th>Type of model</th><th>AIC values</th></tr>
<tr><td>ARIMA(0,1,0)
Naïve model
(Random walk)</td><td><pre>
> summary(arima.model2)

Call:
arima(x = model_data[train, j], order = c(0, 1, 0))

sigma^2 estimated as 2.021: log likelihood = -1554.65, aic = 3111.3

Training set error measures:
                     ME   RMSE   MAE   MPE   MAPE   MASE   ACF1
Training set 0.06842139 1.420724 0.9947468 0.05174541 0.7429639 0.9989679 -0.06401567
</pre></td></tr>
<tr><td>ARIMA(0,1,1)
Simple
Exponential
Smoothing</td><td><pre>
> summary(arima.model4)

Call:
arima(x = model_data[train, j], order = c(0, 1, 1))

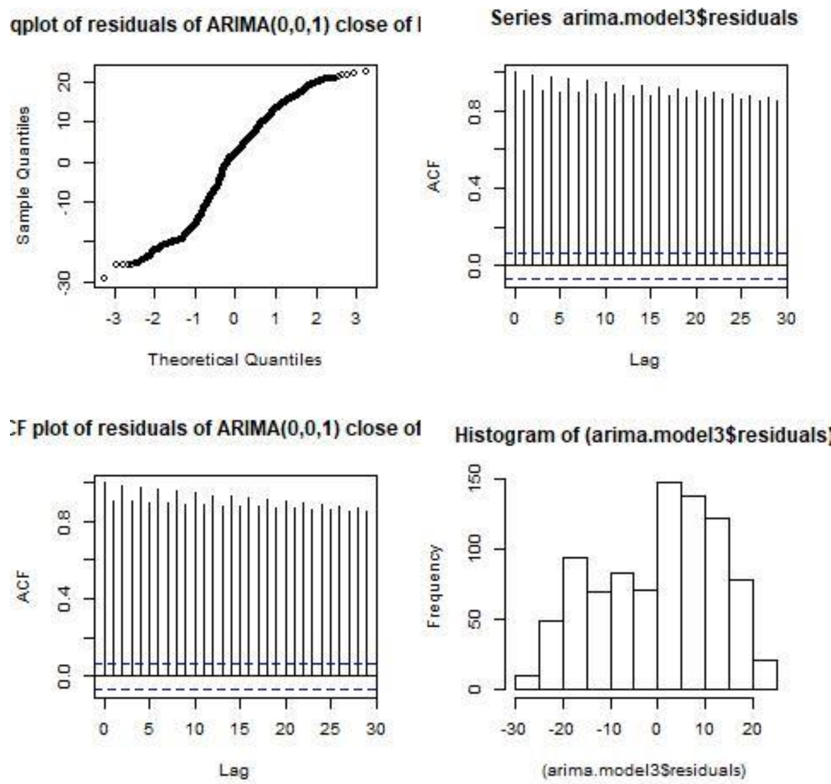
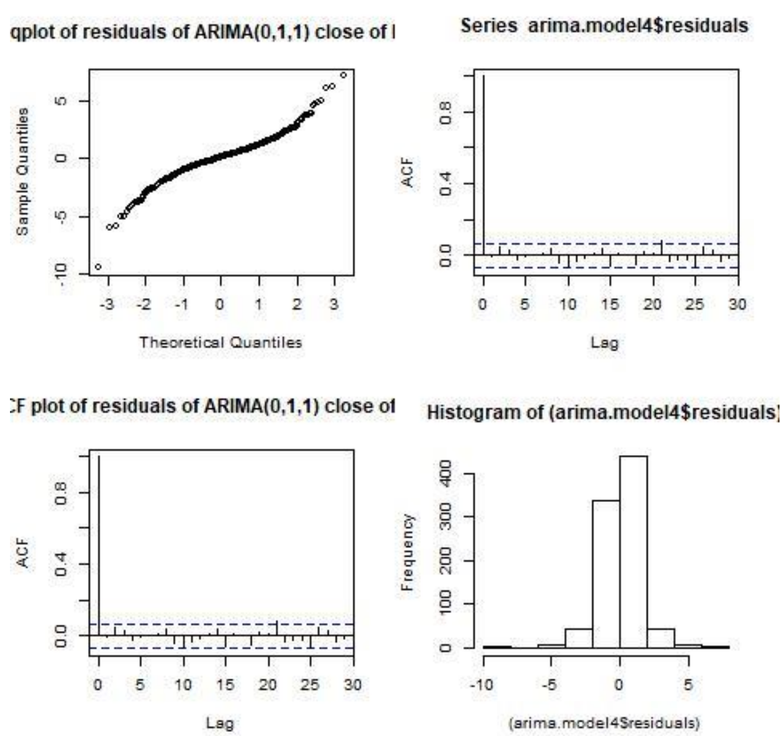
Coefficients:
          ma1
       -0.056
s.e.   0.032

sigma^2 estimated as 2.014: log likelihood = -1553.14, aic = 3110.27

Training set error measures:
                     ME   RMSE   MAE   MPE   MAPE   MASE   ACF1
Training set 0.07262217 1.418272 0.9955407 0.05493083 0.7436778 0.9997652 -0.005754611
</pre></td></tr>
</table>

The above two models are special cases of ARIMA models. The Random model can be considered as the base model to benchmark other models. As seen the AIC values are far higher than our best model. Simple Exponential Smoothing model's weight never goes to zero. These are fairly common and generic models used in time series analysis.

We can observe in diagnostic plot that the residual's autocorrlation slowly decreases implying that it is not stationary and also the q plot are not entirely linear. Hence these models are not well suited for our analysis.

Type of Model	Diagnostic Plots
ARIMA(0,1,0) Naïve model (Random walk)	
ARIMA(0,1,1) Simple Exponential Smoothing	

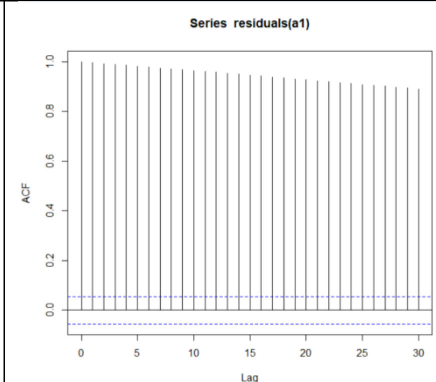
Implementation steps specific to GARCH model

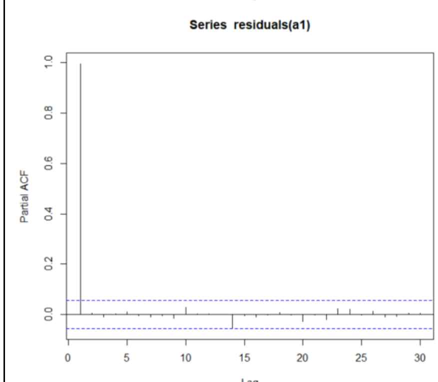
Please note we are only presenting the steps unique to GARCH model

1.	Look at the ACF and PACF of Close and Close^2. If the data includes only AR terms, then ARCH is suggested. But, if ACF and PACF of Close and Close^2 show ARMA pattern, then GARCH is suggested.
2.	Since, our data that is Closing price of the MMM stock has short periods of increased variation and ACF and PACF (figure no. 1& figure no 2) plots of Close and Close^2 show ARMA and not just AR terms, we will implement the GARCH.
3.	The ACF of the squared series follows an ARMA pattern because both the ACF and PACF taper. This suggests a GARCH model.

Comparing sample GARCH Models

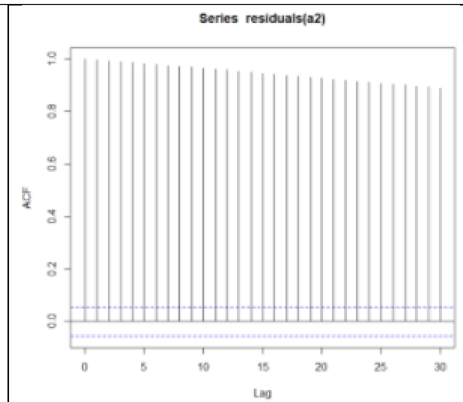
<table>
<tr><th>Type of model</th><th>AIC and SIC values</th><th>ACF and PACF plots of residuals</th></tr>
<tr><td>GARCH(1,1)</td><td><pre>
Standardised Residuals Tests:
                                Statistic p-Value
 Jarque-Bera Test   R   Chi^2 101.1953 0
 Shapiro-Wilk Test R   W   0.7717686 0
 Ljung-Box Test   R   Q(10) 10169.38 0
 Ljung-Box Test   R   Q(15) 14921.19 0
 Ljung-Box Test   R   Q(20) 19447.15 0
 Ljung-Box Test   R^2 Q(10) 14.8418   0.1379368
 Ljung-Box Test   R^2 Q(15) 29.72949 0.01293519
 Ljung-Box Test   R^2 Q(20) 33.7174   0.02810117
 LM Arch Test   R   TR^2   20.70037 0.05494399

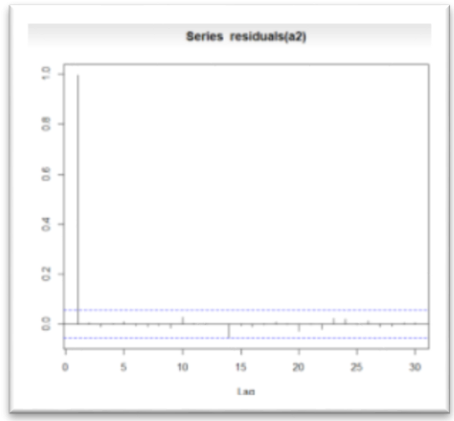
Information Criterion Statistics:
      AIC   BIC   SIC   HQIC
-1.509483 -1.493138 -1.509503 -1.503340
</pre></td><td>


</td></tr>
</table>

GARCH(1,2)

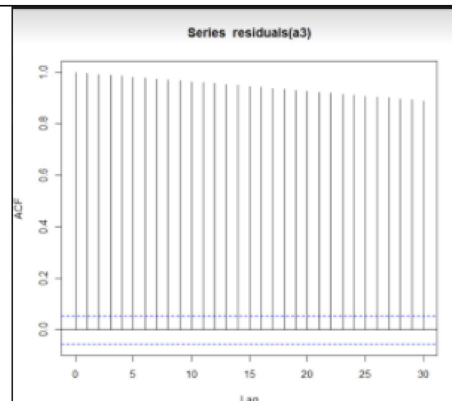
```
Standardised Residuals Tests:
                                Statistic p-Value
 Jarque-Bera Test   R    Chi^2  99.43464  0
 Shapiro-Wilk Test  R    W      0.7737016 0
 Ljung-Box Test     R    Q(10)  10129.11  0
 Ljung-Box Test     R    Q(15)  14859.76  0
 Ljung-Box Test     R    Q(20)  19365.55  0
 Ljung-Box Test     R^2  Q(10)  14.60591  0.1471038
 Ljung-Box Test     R^2  Q(15)  28.90684  0.01653467
 Ljung-Box Test     R^2  Q(20)  32.71124  0.03628479
 LM Arch Test       R    TR^2   20.51277  0.05798679

Information Criterion Statistics:
      AIC       BIC       SIC      HQIC
-1.505982 -1.485550 -1.506013 -1.498303
```

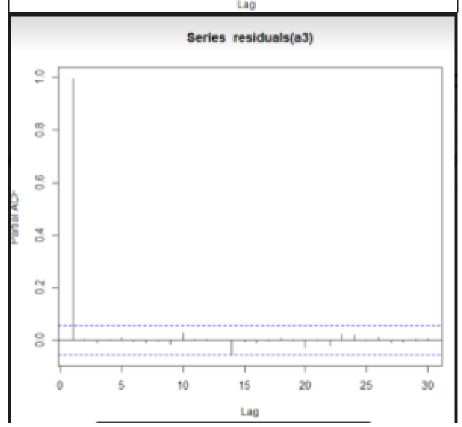


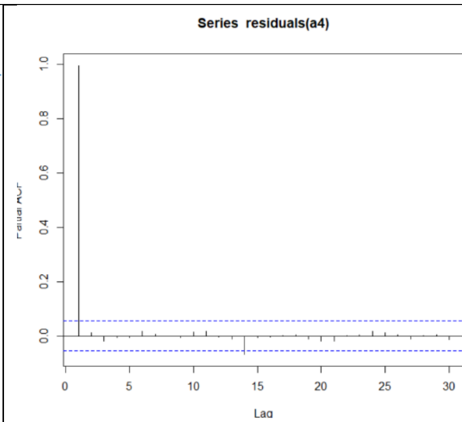
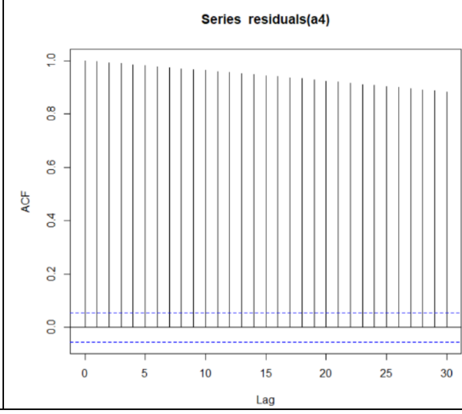


GARCH(2,1)

```
Standardised Residuals Tests:
                                Statistic p-Value
 Jarque-Bera Test   R    Chi^2  112.9731  0
 Shapiro-Wilk Test  R    W      0.7759069 0
 Ljung-Box Test     R    Q(10)  10117.3   0
 Ljung-Box Test     R    Q(15)  14843.87  0
 Ljung-Box Test     R    Q(20)  19319.84  0
 Ljung-Box Test     R^2  Q(10)  10.30178  0.4144276
 Ljung-Box Test     R^2  Q(15)  35.24567  0.002268445
 Ljung-Box Test     R^2  Q(20)  39.403    0.005938495
 LM Arch Test       R    TR^2   16.89719  0.1535059

Information Criterion Statistics:
      AIC       BIC       SIC      HQIC
-1.508918 -1.488486 -1.508949 -1.501239
```





GARCH(2,2)	Standardised Residuals Tests: Statistic p- Jarque-Bera Test R Chi^2 112.9727 0 Shapiro-Wilk Test R W 0.7759069 0 Ljung-Box Test R Q(10) 10117.3 0 Ljung-Box Test R Q(15) 14843.86 0 Ljung-Box Test R Q(20) 19319.84 0 Ljung-Box Test R^2 Q(10) 10.30174 0. Ljung-Box Test R^2 Q(15) 35.24533 0. Ljung-Box Test R^2 Q(20) 39.40267 0. LM Arch Test R TR^2 16.89715 0. Information Criterion Statistics: AIC BIC SIC HQIC -1.507327 -1.482809 -1.507372 -1.498112	 

Executive Summary

Importance of the Project in Real World

This model will be especially useful for anyone who wants to trade, looking for short-term gains in stocks. Human decision making is marred because of bias and more often influenced by emotions. The use of analytics will help to objectively evaluate and decide on the best stock to trade. Another reason is that humans are not especially suited to analyze large amount of data. Computers can process much faster and produce better results. Currently the models are not reliable enough to be used independently, hence one must use other forms of information (intrinsic data about the company, political climate etc.,) in tandem with the results of the analytics model to make the best decision.

Statement of Objective

We implemented time series analysis and came up with the best model for forecasting the future stock price using ARIMA and GARCH model. Our hope was to test the validity of research in technical analysis of stock data with the help of our model.

Gap in Literature

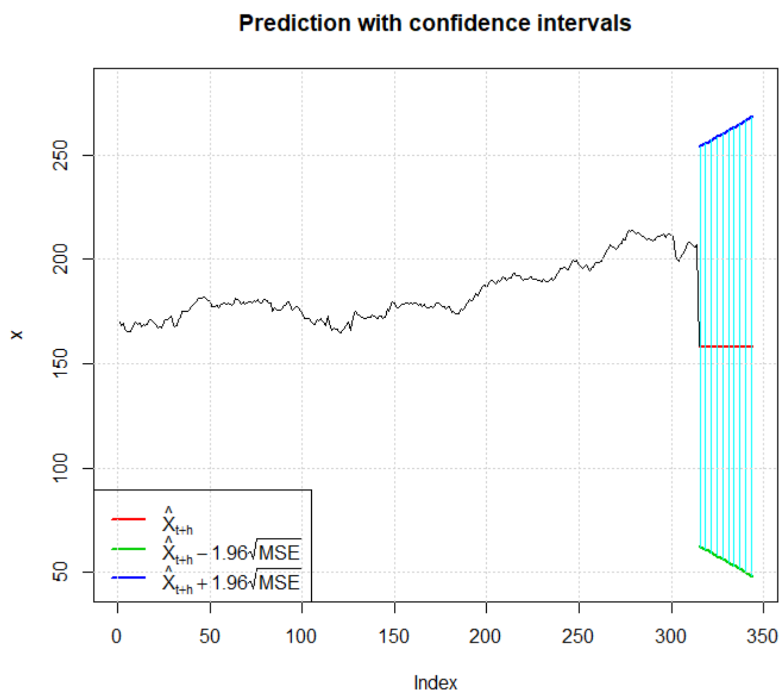
While numerous papers cite best approaches for time series forecasts, none cite the actual results of employing these results on actual stock market. With our project we plotted the forecasted stock price against the actual stock price and noticed a stark difference in the two.

Fulfilment of Objectives

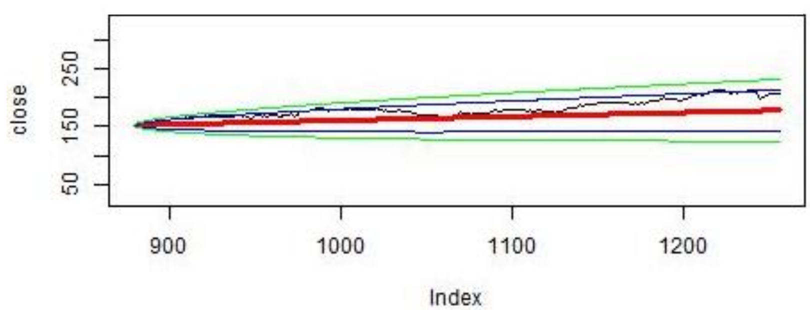
We successfully learned to employ time series models to analyze time varying data sets. We also learned to identify best models based on diagnostic plots, AIC, BIC, and SIC values. This project played a major part in creating awareness in our team regarding financial markets and pursue data analysis as a tool to solve big decision problems. Also, the forecast plots exposed gaps between the actual prices and the forecasted value.

Conclusion

We successfully implemented time series methods namely ARIMA and GARCH on company 3M listed in S&P 500 index. We systematically analyzed the dataset using the above methods and figured out the best variant. Finally, we used the best variant to run a comparison to make a recommendation about which model to use for short term forecasting of the stock prices. As observed from the standard deviation, the values of the GARCH model far greater than the ARIMA, which conclusively proves that though ARIMA might be a simpler model it performs comparatively better for short term forecasting. Hence, we recommend ARIMA(0,1,2) with drift to forecast the time series data of the stock price. We therefore support Dr. Tse's conclusion that ARIMA performs as well as more complex models with regards to short term forecasting.



Forecast Plot of best ARIMA model – ARIMA(0,1,2)



After plotting the forecasted value (on training data) against the test data set, we noticed a stark difference in the two prices. Hence, we conclude that even our best ARIMA model will not be suitable for making substantial gains on the volatile stock market. Many qualitative factors and quantitative factors (which is the focus of Fundamental Analysis) are not covered by technical analysis dataset, reinforcing the random walk model of stock market prices.

References

[1] Arowolo, W.B, Predicting Stock Prices Returns Using Garch Model

[2] Ayodele A. Adebiyi, Stock Price Prediction Using the ARIMA Model

[3] Robert Engle, The Use of ARCH/GARCH Models in Applied Econometrics

[4] Trend Selene Yue Xu, Stock Price Forecasting Using Information from Yahoo Finance and Google

[5] https://people.duke.edu/~rnau/arimrule.htm

[6] http://www.stern.nyu.edu/rengle/GARCH101.PDF

[7] https://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/

[8] https://en.wikipedia.org/wiki/Autoregressive_conditional_heteroskedasticity

Appendix

R Code for Project analysis

```
#Clears the workspace

remove(list=ls())


#List of Custom file built

# Differencing(data-set, order of differencing to be done(like 1,2,3...))

# Dataform(data-set, "transformation function") [Transformation Functions available - Log, sqrt, raw (means original data), percent]


# sourcing all the custom files built into R-environment

#Command : source("filepath with the file name")

source("C:/Laxmi/ISEN 613/Project/Project_codes/Dataform.R")

source("C:/Laxmi/ISEN 613/Project/Project_codes/Differencing.R")


'

#Packages to be installed

install.packages("tseries")

install.packages("zoo")

install.packages("leaps")

install.packages("locfit")

install.packages("mgcv")

install.packages("nlme")

install.packages("gridExtra")

install.packages("ggplot2")

install.packages("dplyr")

install.packages("forecast")

install.packages("rugarch")
```

```
install.packages("e1071")

install.packages("fGarch")

install.packages("FinTS", repos = "http://R-Forge.R-project.org")

install.packages("funitRoots", repos = "http://R-Forge.R-project.org")

'



#List of Libraries

library(openxlsx)

library(tseries)

library(zoo)

library(gridExtra)

library(leaps)

library(locfit)

library(nlme)

library(mgcv)

library(readxl)

library(dplyr)

library(forecast)

library(FinTS)

library(e1071)

library(funitRoots)

library(rugarch)

library(fGarch)


#reading data file. Enclosed in " " i.e "C:/Laxmi/ISEN 613/Project/Project_Data_Set.xlsx" is file path

input <- read.csv(file = "C:/Laxmi/ISEN 613/Project/Project_Data_Set_S_and_P_500.csv" , header=TRUE,
sep = ",")
```

```
#"Stocks" in the data-set are uniquely identified and stored as row vector in "stocks_list"

stocks_list = unique(input$Name)


# "while" loop has been used to evaluate and model data for each stock.

# subroutine files, like "Differences" has been called in loop, so as to be evaluated

#each time the loop is run.

i = 1


input_data = as.data.frame(input[input$Name == stocks_list[i],])

#Transforming data to be used for model development and testing

#************************************

model_data = Dataform(input_data,"raw")

index_column = matrix(c(seq(from = 0, to = ((nrow(model_data))-1), by = 1)),ncol=1)

model_data = data.frame(model_data, index_column)

colnames(model_data)[(ncol(model_data))] <- "index"



#Plots of data - Automatic Generation

j=1

while(j <= (ncol(model_data)-1))

 {

   lm.fit1 = lm(model_data[,j]~index, data = model_data)

   'lm.fit2 = lm(model_data[,j]~poly(index,2), data = model_data)

   lm.fit3 = lm(model_data[,j]~poly(index,3), data = model_data)


   best_fit =
which.max(c(summary(lm.fit1)$adj.r.squared,summary(lm.fit2)$adj.r.squared,summary(lm.fit3)$adj.r.sq
uared))

   '
```

```
   #if (best_fit == 1)
   {lm.fit = lm.fit1}
   'f (best_fit == 2)
   {lm.fit = lm.fit2}
   if (best_fit == 3)
   {lm.fit = lm.fit3}'

  file_name = paste("Plot of", names(model_data)[j], " of ", stocks_list[i])
  jpeg(paste(file_name,".jpeg"))
  par(mfrow=c(2,1))
  plot(model_data[,j], xlab = "Index", ylab = names(model_data)[j] , main = file_name )
  lines(predict(lm.fit), lwd=3, col="blue")
  plot((model_data[,j]-predict(lm.fit, data=model_data)), xlab = "Index", ylab = paste("Residuals of",
names(model_data)[j]), main = paste(file_name, "residuals"))
  dev.off()
  j = j+1
}

#Stationarizing the data by differencing and Plotting the same
#*******************************************************
order_of_differencing = 1
model_data_diff = Differencing(model_data,order_of_differencing)
j=1
while(j <= ncol(model_data_diff))
{
  file_name = paste("Plot of", names(model_data)[j], "of", stocks_list[i], " stock differenced ",
order_of_differencing , "time in order")
  jpeg(paste(file_name,".jpeg"))
  plot(model_data_diff[,j], xlab = "Index", ylab = paste("Difference of",names(model_data)[j]) , main =
file_name , type="l")
```

```
  dev.off()
  j = j+1
}

#**********************************************************************
analysis_data = model_data_diff #this is the data that is stationary and further worked upon in
                              # in ARIMA modelling and GARCH Modelling

#Dickey-Fuller Test on individual column of "analysis_data". We can feed any data
j=1
adf.test_p_value = matrix(c(rep(0,(ncol(analysis_data)-1)*4)),ncol=4)
while(j <= (ncol(analysis_data)-1))
{
  a = adf.test(analysis_data[,j], alternative ="stationary", k=0)
  #"k" is the number of "lag" involved. If "k" is not specified, we will "k" for which p-value is least
  #Null Hypothesis: Not Stationary || Alternative Hypothesis = Stationary
  # p > 0.05 : Null Hypothesis is "True", otherwise, alternate is true.
  adf.test_p_value[j,2] = a$p.value #Storing p-value
  adf.test_p_value[j,1] = paste(stocks_list[i],"_",names(analysis_data)[j]) #Storing the name of data_set
to which the p-value belongs

  a = adf.test(analysis_data[,j], alternative ="stationary")
  adf.test_p_value[j,3] = a$parameter
  adf.test_p_value[j,4] = a$p.value
  write.table(adf.test_p_value, paste("C:/Laxmi/ISEN 613/Project/a1_log_diff.txt"), sep="\t")
  j = j+1
}

#ACF and PACF plots are generated on stationary data. Parameters
```

```
#p - estimated from PACF plots

#q - estimated from ACF plots


j = 1

while(j <= (ncol(analysis_data)-1))

{

 jpeg(paste("ACF plot of", names(analysis_data)[j],"of", stocks_list[i],".jpeg"))

 plot(acf(analysis_data[,j]), main=paste("ACF plot of",names(analysis_data)[j],"of",stocks_list[i]))

 dev.off()


 jpeg(paste("PACF plot of", names(analysis_data)[j],"of", stocks_list[i],".jpeg"))

 plot(pacf(analysis_data[,j]), main=paste("PACF plot of",names(analysis_data)[j],"of",stocks_list[i]))

 dev.off()

 j=j+1

}

##########################################################

##########################################################

#                 ARIMA MODELIING                  #

##########################################################

##########################################################


#Splitting data into "training" and "testing"

#For ARIMA original data, i.e raw data. Any differencing done for stationarity

#is fed in parameters (p,q,d), as it is.

train = seq(1:(floor(0.7*nrow(model_data))))

test = seq(from=length(train)+1, to=length(train)+10, by=1)

number_of_future = 10
```

```
#This section of code is the individually evaluate various combinations of ARIMA(p,d,q)

# parameters (p,d,q)


j=4

#while(j <= (ncol(model_data)-1))

#{

  arima.model1 = arima(model_data[train,j], order=c(1,0,0))

  pred.model1 = forecast(arima.model1,h=number_of_future)

  file_name_ar_1 = paste("Plot of ARIMA(1,0,0) of",names(model_data)[j]," of ",stocks_list[i])

  max_upper_1 = (pred.model1$upper[which.max(pred.model1$upper)])+100

  min_lower_1 = (pred.model1$lower[which.min(pred.model1$lower)])-100

  max_residual_1 = pred.model1$residuals[which.max(pred.model1$residuals)]

  min_residual_1 = pred.model1$residuals[which.min(pred.model1$residuals)]

  jpeg(paste(file_name_ar_1,".jpeg"))

  par(mfrow = c(2,1))

  plot.ts(model_data[(length(train)+1):nrow(model_data),6],
model_data[(length(train)+1):nrow(model_data),j],xlab="Index",
ylab=names(model_data)[j],main=paste(file_name_ar_1,names(model_data)[j]),lwd=1, col="black",
type="l", ylim = c(min_lower_1, max_upper_1))

  lines(pred.model1$mean, lwd = 3, col="red")

  lines(pred.model1$upper[,1],lwd = 1, col="blue")

  lines(pred.model1$lower[,1],lwd = 1, col="blue")

  lines(pred.model1$upper[,2],lwd = 1, col="green")

  lines(pred.model1$lower[,2],lwd = 1, col="green")

  plot(pred.model1$residuals, xlab="Index", ylab="Residuals of Forecasted Values",
main=paste("Residuals of Forecasted Values",names(model_data)[j]),lwd=1, col="black", type="l", ylim =
c(min_residual_1, max_residual_1))

  dev.off()


  arima.model2 = arima(model_data[train,j], order=c(0,1,0))
```

```
pred.model2 = forecast(arima.model2,h=number_of_future)
file_name_ar_2 = paste("Plot of ARIMA(0,1,0) of",names(model_data)[j]," of ",stocks_list[i])
max_upper_2 = (pred.model2$upper[which.max(pred.model2$upper)])+100
min_lower_2 = (pred.model2$lower[which.min(pred.model2$lower)])-100
max_residual_2 = pred.model2$residuals[which.max(pred.model2$residuals)]
min_residual_2 = pred.model2$residuals[which.min(pred.model2$residuals)]
jpeg(paste(file_name_ar_2,".jpeg"))
par(mfrow = c(2,1))
plot.ts(model_data[(length(train)+1):nrow(model_data),6],
model_data[(length(train)+1):nrow(model_data),j],xlab="Index",
ylab=names(model_data)[j],main=paste(file_name_ar_2,names(model_data)[j]),lwd=1, col="black",
type="l", ylim = c(min_lower_2, max_upper_2))
lines(pred.model2$mean, lwd = 3, col="red")
lines(pred.model2$upper[,1],lwd = 1, col="blue")
lines(pred.model2$lower[,1],lwd = 1, col="blue")
lines(pred.model2$upper[,2],lwd = 1, col="green")
lines(pred.model2$lower[,2],lwd = 1, col="green")
plot(pred.model2$residuals, xlab="Index", ylab="Residuals of Forecasted Values",
main=paste("Residuals of Forecasted Values",names(model_data)[j]),lwd=1, col="black", type="l", ylim =
c(min_residual_2, max_residual_2))
dev.off()

arima.model3 = arima(model_data[train,j], order=c(0,0,1))
pred.model3 = forecast(arima.model3,h=number_of_future)
file_name_ar_3 = paste("Plot of ARIMA(0,0,1) of",names(model_data)[j]," of ",stocks_list[i])
max_upper_3 = (pred.model3$upper[which.max(pred.model3$upper)])+100
min_lower_3 = (pred.model3$lower[which.min(pred.model3$lower)])-100
max_residual_3 = pred.model3$residuals[which.max(pred.model3$residuals)]
min_residual_3 = pred.model3$residuals[which.min(pred.model3$residuals)]
jpeg(paste(file_name_ar_3,".jpeg"))
```

```
  par(mfrow = c(2,1))

  plot.ts(model_data[(length(train)+1):nrow(model_data),6],
model_data[(length(train)+1):nrow(model_data),j],xlab="Index",
ylab=names(model_data)[j],main=paste(file_name_ar_3,names(model_data)[j]),lwd=1, col="black",
type="l", ylim = c(min_lower_3, max_upper_3))

  lines(pred.model3$mean, lwd = 3, col="red")

  lines(pred.model3$upper[,1],lwd = 1, col="blue")

  lines(pred.model3$lower[,1],lwd = 1, col="blue")

  lines(pred.model3$upper[,2],lwd = 1, col="green")

  lines(pred.model3$lower[,2],lwd = 1, col="green")

  plot(pred.model3$residuals, xlab="Index", ylab="Residuals of Forecasted Values",
main=paste("Residuals of Forecasted Values",names(model_data)[j]),lwd=1, col="black", type="l", ylim =
c(min_residual_3, max_residual_3))

  dev.off()


  arima.model4 = arima(model_data[train,j], order=c(0,1,1))

  pred.model4 = forecast(arima.model4,h=number_of_future)

  file_name_ar_4 = paste("Plot of ARIMA(0,1,1) of",names(model_data)[j]," of ",stocks_list[i])

  max_upper_4 = (pred.model4$upper[which.max(pred.model4$upper)])+100

  min_lower_4 = (pred.model4$lower[which.min(pred.model4$lower)])-100

  max_residual_4 = pred.model4$residuals[which.max(pred.model4$residuals)]

  min_residual_4 = pred.model4$residuals[which.min(pred.model4$residuals)]

  jpeg(paste(file_name_ar_4,".jpeg"))

  par(mfrow = c(2,1))

  plot.ts(model_data[(length(train)+1):nrow(model_data),6],
model_data[(length(train)+1):nrow(model_data),j],xlab="Index",
ylab=names(model_data)[j],main=paste(file_name_ar_4,names(model_data)[j]),lwd=1, col="black",
type="l", ylim = c(min_lower_4, max_upper_4))

  lines(pred.model4$mean, lwd = 3, col="red")

  lines(pred.model4$upper[,1],lwd = 1, col="blue")

  lines(pred.model4$lower[,1],lwd = 1, col="blue")

  lines(pred.model4$upper[,2],lwd = 1, col="green")
```

```
  lines(pred.model4$lower[,2],lwd = 1, col="green")

  plot(pred.model4$residuals, xlab="Index", ylab="Residuals of Forecasted Values",
main=paste("Residuals of Forecasted Values",names(model_data)[j]),lwd=1, col="black", type="l", ylim =
c(min_residual_4, max_residual_4))

  dev.off()


  arima.model5 = arima(model_data[train,j], order=c(1,1,0))

  pred.model5 = forecast(arima.model5,h=number_of_future)

  file_name_ar_5 = paste("Plot of ARIMA(1,1,0) of",names(model_data)[j]," of ",stocks_list[i])

  jpeg(paste(file_name_ar_5,".jpeg"))

  max_upper_5 = (pred.model5$upper[which.max(pred.model5$upper)])+100

  min_lower_5 = (pred.model5$lower[which.min(pred.model5$lower)])-100

  max_residual_5 = pred.model5$residuals[which.max(pred.model5$residuals)]

  min_residual_5 = pred.model5$residuals[which.min(pred.model5$residuals)]

  par(mfrow = c(2,1))

  plot.ts(model_data[(length(train)+1):nrow(model_data),6],
model_data[(length(train)+1):nrow(model_data),j],xlab="Index",
ylab=names(model_data)[j],main=paste(file_name_ar_5,names(model_data)[j]),lwd=1, col="black",
type="l", ylim = c(min_lower_5, max_upper_5))

  lines(pred.model5$mean, lwd = 3, col="red")

  lines(pred.model5$upper[,1],lwd = 1, col="blue")

  lines(pred.model5$lower[,1],lwd = 1, col="blue")

  lines(pred.model5$upper[,2],lwd = 1, col="green")

  lines(pred.model5$lower[,2],lwd = 1, col="green")

  plot(pred.model5$residuals, xlab="Index", ylab="Residuals of Forecasted Values",
main=paste("Residuals of Forecasted Values",names(model_data)[j]),lwd=1, col="black", type="l", ylim =
c(min_residual_5, max_residual_5))

  dev.off()


  arima.model6 = arima(model_data[train,j], order=c(1,0,1))

  pred.model6 = forecast(arima.model6,h=number_of_future)
```

```
file_name_ar_6 = paste("Plot of ARIMA(1,0,1) of",names(model_data)[j]," of ",stocks_list[i])

max_upper_6 = (pred.model6$upper[which.max(pred.model6$upper)])+100

min_lower_6 = (pred.model6$lower[which.min(pred.model6$lower)])-100

max_residual_6 = pred.model6$residuals[which.max(pred.model6$residuals)]

min_residual_6 = pred.model6$residuals[which.min(pred.model6$residuals)]

jpeg(paste(file_name_ar_6,".jpeg"))

par(mfrow = c(2,1))

plot.ts(model_data[(length(train)+1):nrow(model_data),6],
model_data[(length(train)+1):nrow(model_data),j],xlab="Index",
ylab=names(model_data)[j],main=paste(file_name_ar_6,names(model_data)[j]),lwd=1, col="black",
type="l", ylim = c(min_lower_6, max_upper_6))

lines(pred.model6$mean, lwd = 3, col="red")

lines(pred.model6$upper[,1],lwd = 1, col="blue")

lines(pred.model6$lower[,1],lwd = 1, col="blue")

lines(pred.model6$upper[,2],lwd = 1, col="green")

lines(pred.model6$lower[,2],lwd = 1, col="green")

plot(pred.model6$residuals, xlab="Index", ylab="Residuals of Forecasted Values",
main=paste("Residuals of Forecasted Values",names(model_data)[j]),lwd=1, col="black", type="l", ylim =
c(min_residual_6, max_residual_6))

dev.off()


summary(arima.model1)

summary(arima.model2)

summary(arima.model3)

summary(arima.model4)

summary(arima.model5)

summary(arima.model6)


#Diagnostic Plots for above

jpeg(paste("Diagnostic Plots of ARIMA(1,0,0)",names(model_data)[j],"of",stocks_list[i],".jpeg"))
```

```
  par(mfrow=c(2,2))

  qqnorm(arima.mode$residuals, main = paste("qqplot of residuals of
ARIMA(1,0,0)",names(model_data)[j],"of",stocks_list[i]) )

  plot(acf(arima.mode1$residuals), main=paste("ACF plot of residuals of
ARIMA(1,0,0)",names(model_data)[j],"of",stocks_list[i]))

  hist((arima.model1$residuals))

  dev.off()


  jpeg(paste("Diagnostic Plots of ARIMA(0,1,0)",names(model_data)[j],"of",stocks_list[i],".jpeg"))

  par(mfrow=c(2,2))

  qqnorm(arima.model2$residuals, main = paste("qqplot of residuals of
ARIMA(0,1,0)",names(model_data)[j],"of",stocks_list[i]) )

  plot(acf(arima.model2$residuals), main=paste("ACF plot of residuals of
ARIMA(0,1,0)",names(model_data)[j],"of",stocks_list[i]))

  hist((arima.model2$residuals))

  dev.off()


  jpeg(paste("Diagnostic Plots of ARIMA(0,0,1)",names(model_data)[j],"of",stocks_list[i],".jpeg"))

  par(mfrow=c(2,2))

  qqnorm(arima.model3$residuals, main = paste("qqplot of residuals of
ARIMA(0,0,1)",names(model_data)[j],"of",stocks_list[i]) )

  plot(acf(arima.model3$residuals), main=paste("ACF plot of residuals of
ARIMA(0,0,1)",names(model_data)[j],"of",stocks_list[i]))

  hist((arima.model3$residuals))

  dev.off()


  jpeg(paste("Diagnostic Plots of ARIMA(0,1,1)",names(model_data)[j],"of",stocks_list[i],".jpeg"))

  par(mfrow=c(2,2))

  qqnorm(arima.model4$residuals, main = paste("qqplot of residuals of
ARIMA(0,1,1)",names(model_data)[j],"of",stocks_list[i]) )

  plot(acf(arima.model4$residuals), main=paste("ACF plot of residuals of
ARIMA(0,1,1)",names(model_data)[j],"of",stocks_list[i]))
```

```
  hist((arima.model4$residuals))

  dev.off()


  jpeg(paste("Diagnostic Plots of ARIMA(1,1,0)",names(model_data)[j],"of",stocks_list[i],".jpeg"))

  par(mfrow=c(2,2))

  qqnorm(arima.model5$residuals, main = paste("qqplot of residuals of
ARIMA(1,1,0)",names(model_data)[j],"of",stocks_list[i]) )

  plot(acf(arima.model5$residuals), main=paste("ACF plot of residuals of of
ARIMA(1,1,0)",names(model_data)[j],"of",stocks_list[i]))

  hist((arima.model5$residuals))

  dev.off()


  jpeg(paste("Diagnostic Plots of ARIMA(1,0,1)",names(model_data)[j],"of",stocks_list[i],".jpeg"))

  par(mfrow=c(2,2))

  qqnorm(arima.model6$residuals, main = paste("qqplot of residuals ARIMA(1,0,1)
of",names(model_data)[j],"of",stocks_list[i]) )

  plot(acf(arima.model6$residuals), main=paste("ACF plot of
residualsARIMA(1,0,1)",names(model_data)[j],"of",stocks_list[i]))

  hist((arima.model6$residuals))

  dev.off()

#}


#Development of Arima Model. The following command returns the best ARIMA parameters

j=4


#while(j <= (ncol(model_data)-1))

#{

  arima.model_1 = auto.arima(model_data[train,j], max.p = 10, max.q = 10, max.d = 10)

  #predicting / forecasting "n.ahead" and "h" values into future

  forecast.model1=forecast(arima.model_1,h=number_of_future)
```

```
  file_name_ar_best = paste("Plot of best ARIMA output using 'auto.arima' of",names(model_data)[j],"
of ",stocks_list[i])

  max_upper = (forecast.model1$upper[which.max(forecast.model1$upper)])+100

  min_lower = (forecast.model1$lower[which.min(forecast.model1$lower)])-100


  max_residual = forecast.model1$residuals[which.max(forecast.model1$residuals)]

  min_residual = forecast.model1$residuals[which.min(forecast.model1$residuals)]

  jpeg(paste(file_name_ar_best,".jpeg"))

  par(mfrow=c(2,1))

  plot.ts(model_data[(length(train)+1):nrow(model_data),6],
model_data[(length(train)+1):nrow(model_data),j],xlab="Index", ylab=names(model_data)[j],lwd=1,
col="black", type="l", ylim = c(min_lower, max_upper))

  lines(forecast.model1$mean, lwd = 3, col="red")

  lines(forecast.model1$upper[,1],lwd = 1, col="blue")

  lines(forecast.model1$lower[,1],lwd = 1, col="blue")

  lines(forecast.model1$upper[,2],lwd = 1, col="green")

  lines(forecast.model1$lower[,2],lwd = 1, col="green")

  plot(forecast.model1$residuals, xlab="Index", ylab="Residuals of Forecasted Values",
main=paste("Residuals of Forecasted Values",names(model_data)[j]),lwd=1, col="black", type="l", ylim =
c(min_residual, max_residual))

  dev.off()


  jpeg(paste("Diagnostic Plots of",names(model_data)[j],"of",stocks_list[i],".jpeg"))

  par(mfrow=c(2,2))

  qqnorm(forecast.model1$residuals, main = paste("qqplot of
residuals",names(model_data)[j],"of",stocks_list[i]) )

  plot(acf(forecast.model1$residuals), main=paste("ACF plot of
residuals",names(model_data)[j],"of",stocks_list[i]))

  hist((forecast.model1$residuals))

  dev.off()
```

```
  summary(arima.model_1)

  Box.test(forecast.model1$residuals^2,lag=20,type = "Ljung-Box")


  j=j+1

}


##########################################################

##########################################################

#                    GARCH                    #

##########################################################

##########################################################


#PRE-TESTING

#Statistics about the data assigned to "FinTS.stats_anly_data"

j = 1

#************************************************************

FinTS.stats_anly_data = model_data #"Data to be analysed


FinTS.stats_data = matrix(c(rep(0, ((ncol(FinTS.stats_anly_data)-1)*9))), ncol=9) #8 attributes in
"FinTS.stats" command

colnames(FinTS.stats_data) = c("Name of Quantity","Start", "Size", "Mean", "Standard.Deviation",
"Skewness","Excess.Kurtosis", "Minimum","Maximum")

while(j <= (ncol(FinTS.stats_anly_data)-1))

{

  stats_fin = FinTS.stats(FinTS.stats_anly_data[,j])

  FinTS.stats_data[j,1] = paste(stocks_list[i],"_",names(FinTS.stats_anly_data)[j])

  'k = 2

  while(k <= (ncol(FinTS.stats_data)))

  {
```

```
   FinTS.stats_data[,k] = stats_fin$names(FinTS.stats_data)[k]

   k=k+1

  }'

  FinTS.stats_data[j,2] = stats_fin$Start

  FinTS.stats_data[j,3] = stats_fin$Size

  FinTS.stats_data[j,4] = stats_fin$Mean

  FinTS.stats_data[j,5] = stats_fin$Standard.Deviation

  FinTS.stats_data[j,6] = stats_fin$Skewness

  FinTS.stats_data[j,7] = stats_fin$Excess.Kurtosis

  FinTS.stats_data[j,8] = stats_fin$Minimum

  FinTS.stats_data[j,9] = stats_fin$Maximum

  j=j+1

}


#"Jb.test" of "FinTS.stats_anly_data" stored in "FinTS.stats_JBtest_data"

j=1

FinTS.stats_JBtest_data = matrix(c(rep(0, ((ncol(FinTS.stats_anly_data)-1)*4))), ncol=4) #3 attributes in
"FinTS.stats" command

colnames(FinTS.stats_JBtest_data) = c("Name of Quantity","X-squared", "df", "p-value")

while(j <= (ncol(FinTS.stats_anly_data)-1))

{

  jb_test = jarque.bera.test(FinTS.stats_anly_data[,j])

  FinTS.stats_JBtest_data[j,1] = paste("JB test on",stocks_list[i],"_",names(FinTS.stats_anly_data)[j])

  FinTS.stats_JBtest_data[j,2] = jb_test$statistic

  FinTS.stats_JBtest_data[j,3] = jb_test$parameter

  FinTS.stats_JBtest_data[j,4] = jb_test$p.value

  j=j+1

}
```

```
#ARCH test on "FinTS.stats_anly_data"

j=1

#****************************************************

number_of_lag = 1

FinTS.stats_Archtest_data = matrix(c(rep(0, ((ncol(FinTS.stats_anly_data)-1)*4))), ncol=4) #3 attributes
in "FinTS.stats" command

colnames(FinTS.stats_Archtest_data) = c("Name of Quantity","X-squared", "df", "p-value")

while(j <= (ncol(FinTS.stats_anly_data)-1))

{

  arch_test = ArchTest(FinTS.stats_anly_data[,j], lag = number_of_lag)

  FinTS.stats_Archtest_data[j,1] = paste("ARCH Test
on",stocks_list[i],"_",names(FinTS.stats_anly_data)[j],"with lag",number_of_lag )

  FinTS.stats_Archtest_data[j,2] = arch_test$statistic

  FinTS.stats_Archtest_data[j,3] = arch_test$parameter

  FinTS.stats_Archtest_data[j,4] = arch_test$p.value

  j=j+1

}



##fitting garch model

a1=garchFit(formula = ~ garch(2,2),data = model_data[train,j])

summary(a1)


plot.ts(model_data[test,6], model_data[test,j],xlab="Index", ylab=names(model_data)[j],lwd=1,
col="black", type="l")

lines(forecast.model1$mean[1:10], lwd = 3, col="red")

#lines(forecast.model1$upper[test,1],lwd = 1, col="blue")

#lines(forecast.model1$lower[test,1],lwd = 1, col="blue")

lines(forecast.model1$upper[1:10,2],lwd = 1, col="green")

lines(forecast.model1$lower[1:10,2],lwd = 1, col="green")
```

```
lines((predict(a1))$meanForecast[]+2*a1@sigma.t, lwd = 2, col="blue")

lines((model_data[test,j])+2*a1@sigma.t, lwd = 2, col="blue")


a2=garchFit(formula = ~garch(1,1),data = New_data$Open)

summary(a2)


a3=garchFit(formula = ~garch(1,1),data = New_data$High)

summary(a3)


a4=garchFit(formula = ~garch(1,1),data = New_data$Low)

summary(a4)


a5=garchFit(formula = ~garch(1,1),data = New_data$Volume)

summary(a5)


predict(a1)


plot(a1)



############################################################

############################################################

#                    GARCH                    #

############################################################

############################################################


j = 1

fit1 = auto.arima(FinTS.stats_anly_data[,j],trace = TRUE, test = "kpss", ic="aic" )

j = j+1
```

fit2 = auto.arima(FinTS.stats_anly_data[,j],trace = TRUE, test = "kpss", ic="aic")

j = j+1

fit3 = auto.arima(FinTS.stats_anly_data[,j],trace = TRUE, test = "kpss", ic="aic")

j = j+1

fit4 = auto.arima(FinTS.stats_anly_data[,j],trace = TRUE, test = "kpss", ic="aic")

j = j+1

#ARCh effect test

Box.test(fit1$residuals^2,lag=12,type = "Ljung-Box")

Box.test(fit2$residuals^2,lag=12,type = "Ljung-Box")

Box.test(fit3$residuals^2,lag=12,type = "Ljung-Box")

Box.test(fit4$residuals^2,lag=12,type = "Ljung-Box")

less than 0.05 , therefore we reject the null hypothesis of no ARCH effect

j=1

res.garch1_spec = ugarchspec(variance.model = list(garchOrder=c(1,1)), mean.model = list(armaOrder= c(1,1)))

res.garch1_fit = ugarchfit(spec = res.garch1_spec, data = FinTS.stats_anly_data[,j])

print(res.garch1_fit)

j = j+1

res.garch2_spec = ugarchspec(variance.model = list(garchOrder=c(1,1)), mean.model = list(armaOrder= c(1,1)))

res.garch2_fit = ugarchfit(spec = res.garch2_spec, data = FinTS.stats_anly_data[,j])

print(res.garch1_fit)

j = j+1

res.garch3_spec = ugarchspec(variance.model = list(garchOrder=c(1,1)), mean.model = list(armaOrder= c(1,1)))

res.garch1_fit = ugarchfit(spec = res.garch3_spec, data = FinTS.stats_anly_data[,j])

```
print(res.garch3_fit)

j = j+1

res.garch4_spec = ugarchspec(variance.model = list(garchOrder=c(1,1)), mean.model = list(armaOrder=
c(1,1)))

res.garch1_fit = ugarchfit(spec = res.garch4_spec, data = FinTS.stats_anly_data[,j])

print(res.garch4_fit)

ctrl= list(tol= 1e-7, delta = 1e-9)

j = 1

res_garch1_roll = ugarchroll(res.garch1_spec,FinTS.stats_anly_data[,j],n.start = 1000,refit.every =
1,refit.window = "moving",solver = "hybrid",calculate.VaR = TRUE,VaR.alpha = 0.01,keep.coef =
TRUE,solver.control = ctrl,fit.control = list(scale=1))

report(res_garch1_roll, type = "Var", VaR.alpha = 0.01, conf.level=0.99)

plot(res.garch1_fit)

j+1

res_garch2_roll = ugarchroll(res.garch2_spec,FinTS.stats_anly_data[,j],n.start = 1000,refit.every =
1,refit.window = "moving",solver = "hybrid",calculate.VaR = TRUE,VaR.alpha = 0.01,keep.coef =
TRUE,solver.control = ctrl,fit.control = list(scale=1))

report(res_garch2_roll, type = "Var", VaR.alpha = 0.01, conf.level=0.99)

plot(res.garch2_fit)

j+1

res_garch3_roll = ugarchroll(res.garch3_spec,FinTS.stats_anly_data[,j],n.start = 1000,refit.every =
1,refit.window = "moving",solver = "hybrid",calculate.VaR = TRUE,VaR.alpha = 0.01,keep.coef =
TRUE,solver.control = ctrl,fit.control = list(scale=1))

report(res_garch3_roll, type = "Var", VaR.alpha = 0.01, conf.level=0.99)

plot(res.garch3_fit)
```

```
j+1

res_garch4_roll = ugarchroll(res.garch4_spec,FinTS.stats_anly_data[,j],n.start = 1000,refit.every = 1,refit.window = "moving",solver = "hybrid",calculate.VaR = TRUE,VaR.alpha = 0.01,keep.coef = TRUE,solver.control = ctrl,fit.control = list(scale=1))

report(res_garch4_roll, type = "Var", VaR.alpha = 0.01, conf.level=0.99)

plot(res.garch4_fit)


res_garch1_fcst = ugarchforecast(res.garch1_fit,n.ahead = 12)

res_garch1_fcst


res_garch2_fcst = ugarchforecast(res.garch2_fit,n.ahead = 12)

res_garch2_fcst


res_garch3_fcst = ugarchforecast(res.garch3_fit,n.ahead = 12)

res_garch3_fcst


res_garch4_fcst = ugarchforecast(res.garch4_fit,n.ahead = 12)

res_garch4_fcst



##############################################################

##############################################################
```