FOSSEE Summer Fellowship Report

on

FLOSS - R

submitted by

Ashwin Guptha (Vellore Institute of Technology, Chennai)

under the guidance of

Prof. Kannan M. Moudgalya
Chemical Engineering Department
IIT Bombay

Prof. Radhendushka Srivastava
Department of Mathematics
IIT Bombay

June 15, 2020

Acknowledgement

I want to express my sincere gratitude to Prof. Kannan M. Moudgalya, Department of Chemical Engineering, IIT Bombay, for creating the FOSSEE Fellowship programme and providing students from all over India an opportunity to participate in it. I would equally like to thank my FLOSS mentor, Prof. Radhendushka Srivastava, Department of Mathematics, IIT Bombay, for his immense support, patience, motivation, knowledge & influence throughout this fellowship and for helping me on various statistical models. I would also like to express my gratitude to the other members of the R FLOSS team, namely Mrs. Smita Wangikar and Mr. Digvijay Singh for their guidance and valuable inputs throughout the fellowship and also for providing me with an overview on data analysis and LaTeX. I would also like to thank the other fellows who got selected along with me, namely M. Sai Anand, Sakshee Phade and Amish Sharma for their support, intellectual discussions and enthusiasm. I am very grateful to be given such a fantastic opportunity to work on this exciting project.

Contents

Chapter 1

Introduction

In this report, I mention my contributions to open-source software, made in the duration of the FOSSEE Fellowship, starting from 20th April 2020 to 15th June 2020. Contributions were made using a Free-Libre/Open Source Software (FLOSS) known as "R" as a part of the FOSSEE project by IIT Bombay and MHRD, Government of India. The FOSSEE project is a part of the National Mission on Education through ICT. The thrust area is the adaptation and deployment of open-source simulation packages equivalent to proprietary software, funded by MHRD, based at the Indian Institute of Technology Bombay (IITB). My contributions involved making Spoken Tutorial scripts and analysis of the effect of COVID-19 on the Indian stock market.

Chapter 2

Spoken Tutorial

The Spoken Tutorial project aims to make video tutorials on Free and Open Source Software (FOSS) available in several Indian languages. The goal is to enable the use of spoken tutorials to teach in any Indian language to learners of all levels of expertise - Beginner, Intermediate or Advanced. Every tutorial has to go through a series of stages to ensure that it is perfect for its audience, which is crucial for achieving the goal of this project. I contributed to the creation of the "Decision Tree" and "Random Forest" classification tutorial scripts.

2.1 Decision Tree

The decision tree is a supervised learning algorithm useful for classification by splitting data into two or more subsets based on the significance of the input variables. This tutorial shall explain how to use the decision tree algorithm on a data set using R packages. The data set used is "Aids2" from the "MASS" package [1]. Other packages used were "party" [2] and "caret" [3] for applying and visualizing the accuracy of the algorithm.

2.2 Random Forest

The random forest algorithm uses a set of decision trees. Each tree is assigned a random collection of features. This tutorial shall explain the significance of this algorithm and how it is used on a data set using R packages. It also gives a comparison with other classification algorithms and techniques described in previous tutorials. The data set used is "Aids2" from the "MASS" package [1]. Other packages used were "randomForest" [4] and "caret" [3] for applying and visualizing the accuracy of the algorithm.

Chapter 3

Analysis of the effects of COVID-19 on Indian stock market

3.1 Abstract

The stock market always excited many researchers and business analysts. It is a general belief that predicting or capturing a trend in the stock market is close to impossible due to its randomness. Now due to COVID-19 pandemic, the fluctuations have been more random than they were before, which further piques the interest for researchers to capture a trend of a challenging market. Applications of models like "spline regression," "auto-regressive" and "GARCH" show great promise in time series analysis of the market. In this report, I focused on capturing the trend of both Nifty and Sensex, before and after the pandemic. I shall conclude in this report whether COVID-19 affected the market or not, and what model best captures the trend using R.

3.2 Introduction

COVID-19 is enormously impacting various economic sectors of the world. One of these is the stock market. The stock market is defined as an aggregation of buyers and sellers of stocks or shares which represent possession of part of a business. In India, there are two leading stock exchanges, namely, Bombay Stock Exchange (BSE), whose index is Sensex and the National Stock Exchange (NSE), whose index is Nifty. Stock market data was very irregular, and during COVID-19, its trend changed dramatically. Various statistical models were already available for prediction, but the goal was to find the one fitting best over the data under observation. Spline regression tested to be the best option for a prediction model that represents the major trends from the original data. I also calculated the residuals from the model. Later the differentiated curves of the fitted data were observed to realize the rate of change of the stock market. Also, the initial residuals indicate a prominent periodic curve in which the "AR" and "GARCH" models were able to capture the trend well and fit satisfactorily. The overall objective was to examine how, where and when COVID-19 affects the stock market.

3.3 Methodology

3.3.1 Data Collection

The analysis was performed over the stock market open data for both exchanges from 1st January 2019 till 19th May 2020. There were also other indices for the data apart from open, namely, close, high and low. All four show identical trends, so analysis on any one would prove to be sufficient. The Sensex open data was collected from the official BSE India website [5] and the Nifty open data was collected from the official NSE India website [6]. The data obtained was in good shape and no cleaning was required.

3.3.2 Data Exploration

The data consisted of 339 data points.

	Date	Sensex		Date	Nifty
1	2019-01-01	36161.80	1	2019-01-01	10881.70
2	2019-01-02	36198.13	2	2019-01-02	10868.85
3	2019-01-03	35934.50	3	2019-01-03	10796.80
4	2019-01-04	35590.79	4	2019-01-04	10699.70
5	2019-01-07	35971.18	5	2019-01-07	10804.85
6	2019-01-08	35964.62	6	2019-01-08	10786.25

	Date	Sensex		Date	Nifty
334	2020-05-12	31342.93	334	2020-05-12	9168.85
335	2020-05-13	32841.87	335	2020-05-13	9584.20
336	2020-05-14	31466.33	336	2020-05-14	9213.95
337	2020-05-15	31296.28	337	2020-05-15	9182.40
338	2020-05-18	31248.26	338	2020-05-18	9158.30
339	2020-05-19	30450.74	339	2020-05-19	8961.70

Figure 3.1: Time series data of Sensex and Nifty prices

A better method to improve the data is to convert it into a time series. In R, this could be done using the function "xts" of the package "quantmod" [7]. The default function is as follows -

xts(x,order.by=NULL)

x - object containing the time series data
order.by - vector of different dates/times

By executing the following code, I changed the original data into a time series -

```
library(quantmod)
timeS<-xts(dataS[,-1],order.by = dataS[,1])
timeN<-xts(dataN[,-1],order.by = dataN[,1])
```

The next step would be to visualize the data. Significant fluctuations could be observed from the generated plot.

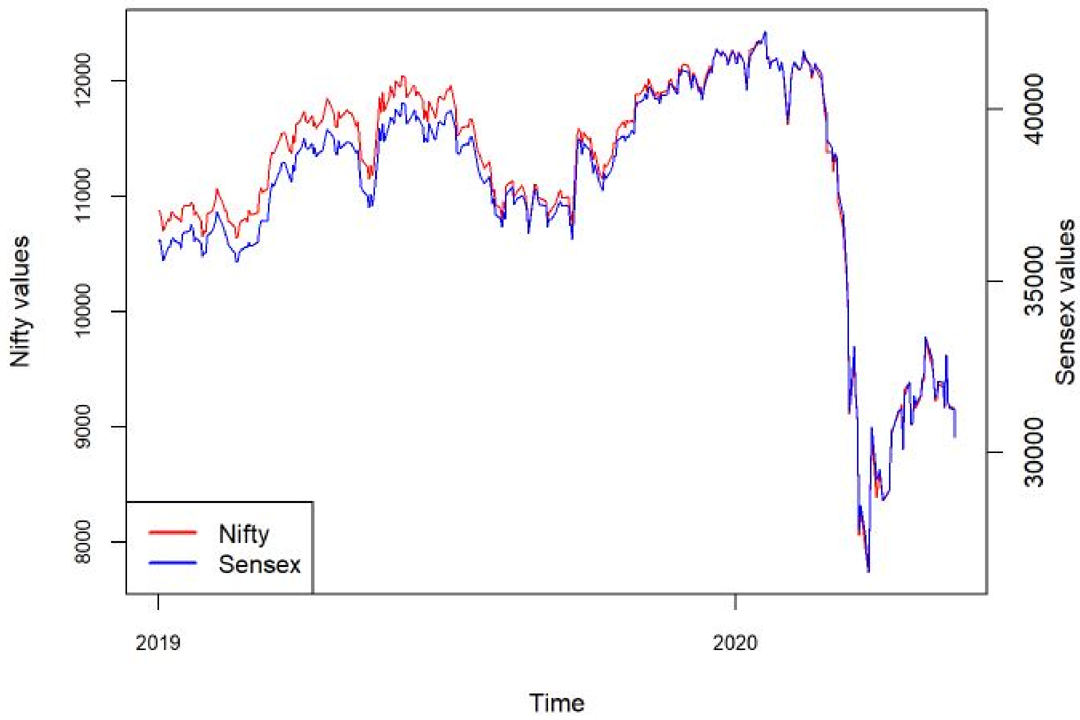

Figure 3.2: Sensex and Nifty data

From figure 3.2, many dips and rises in the values could be observed. Towards February end, there was a sharp decline in the prices. Differentiation would help quantify the significance of the drop. Understandably, such a sharp decline was unseen in the recent past. There was also a sharp increase in March-end and April. The primary statistical measures for both Sensex and Nifty prices were calculated by making use of "summary" function on the respective data.

```
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
26500   36481   38647   37768   40027   42263
```

Figure 3.3: Summary of Sensex data

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
7735	10880	11491	11201	11907	12430

Figure 3.4: Summary of Nifty data

3.3.3 Data Analysis

3.3.3.1 Spline Regression

Spline regression is a preferred version of the standard linear or polynomial regression as it requires a low degree polynomial for curve fitting [8]. A simple linear or polynomial regression fit would prove to be insufficient in this case as the graph follows two different trajectories before and after the COVID-19 pandemic. To capture both trends and obtain an accurate fitting model, I used spline regression. The most common spline regression fit is cubic spline fitting in which I defined a set of points called knot points. Since spline is a special function defined piecewise for the whole fit, knot points help separate each piece and compute individual cubic equations. In other words, these knot points help determine the start and endpoint for each equation. Out of the initial 339 data points, eleven points, as marked in figure 3.5, were chosen to be the knot points. The same knot points were applied to Nifty as well as Sensex as both have similar trends.

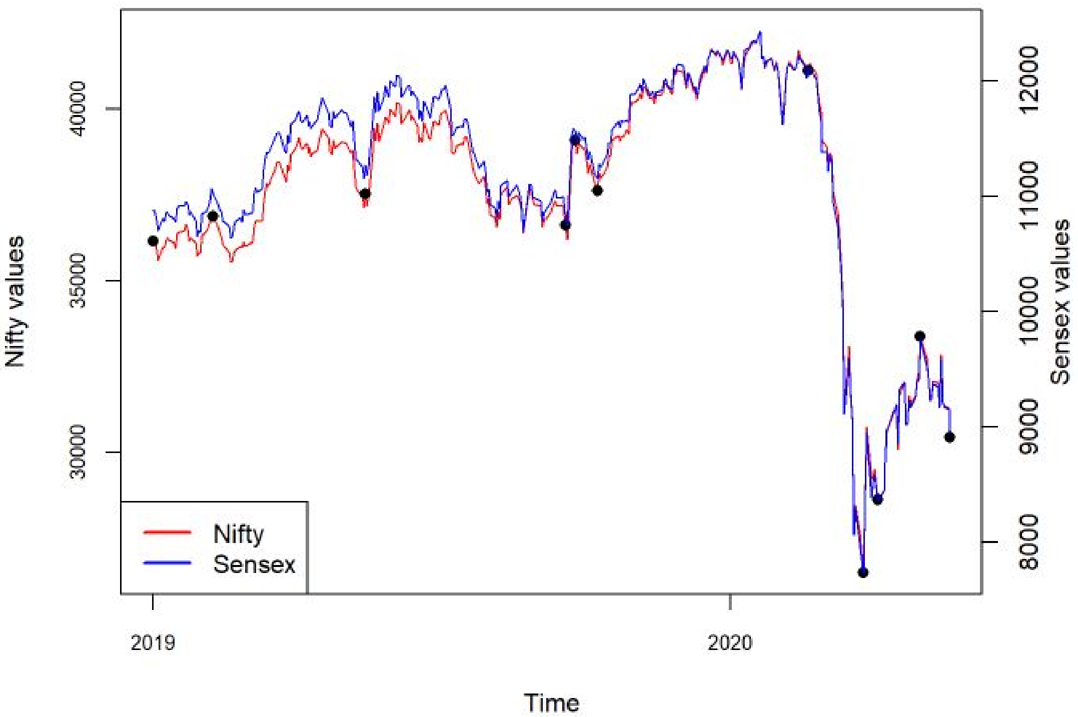

Figure 3.5: Knot points

To apply the spline regression in R, I used the "lm" function with stock values as the dependent variable and date as the independent variable. I utilized cubic regression with the knot points mentioned in figure 3.5. The following code elucidates how to execute spline regression with the given data in R -

```
# Applying spline regression using knots points. The points are the index of chosen values. They will be same for Nifty and Sensex.

# The following is for Nifty and similarly I had applied for Sensex.

point<-c(1,29,91,177,181,189,282,305,311,327,339)
# Knot points by index number.

dataN<-data.frame(index,dataN)

for(i in c(1:(length(point)-1)))
  {
    fit2<-lm(dataN[point[i]:point[i+1],3]~((I(dataN[point[i]:point[i+1],1])+I(dataN[point[i]:point[i+1],1]^2)+I(dataN[point[i]:point[i+1],1]^3))))
    dataN[point[i]:point[i+1],4]<-fit2$fitted
    dataN[point[i]:point[i+1],5]<-fit2$residuals
  }
names(dataN)[4]<-"Fit"
names(dataN)[5]<-"Res"
dataN$points<-NA
dataN[point,6]<-dataN[point,3]
```

Following coefficients were obtained for generating ten equations of the fitted curve as per the given eleven knot points -

Equation	Intercept	Degree1	Degree2	Degree3
1	3.569763e+04	1.368888e+02	-10.954319	2.645245e-01
2	4.814712e+04	-8.208614e+02	17.246052	-1.051567e-01
3	-5.469980e+04	2.179449e+03	-16.318020	3.922962e-02
4	1.605438e+09	-2.693054e+07	150576.500000	-2.806208e+02
5	-3.650485e+05	2.727266e+03	6.828232	-5.283670e-02
6	-1.158827e+05	1.783014e+03	-6.709182	8.373018e-03
7	-3.288133e+07	3.313964e+05	-1109.087000	1.233731e+00
8	-2.576535e+09	2.501869e+07	-80975.870000	8.736028e+01
9	-1.534222e+08	1.440811e+06	-4509.379000	4.704413e+00
10	3.359342e+08	-3.025995e+06	9086.384000	-9.094560e+00

Figure 3.6: Sensex coefficients

Equation	Intercept	Degree1	Degree2	Degree3
1	94199579.63	3.714412e+01	-3.260230	0.079943560
2	13780.81	-2.108956e+02	4.592556	-0.028514710
3	-18820.56	7.156223e+02	-5.399194	0.013050440
4	461264200.00	-7.739206e+06	43281.660000	-80.679170000
5	553656.90	-1.008447e+04	61.754470	-0.124789600
6	-29043.32	4.645525e+02	-1.732930	0.002137055
7	-8821558.00	8.885148e+04	-297.062900	0.330024500
8	-743176700.00	7.218807e+06	-23372.340000	25.223610000
9	-48870700.00	4.588170e+05	-1435.558000	1.497179000
10	94199580.00	-8.482418e+05	2546.241000	-2.547684000

Figure 3.7: Nifty coefficients

From the generated coefficients, equations were created to obtain the best fit. The fit is shown in figures 3.8 and 3.9.

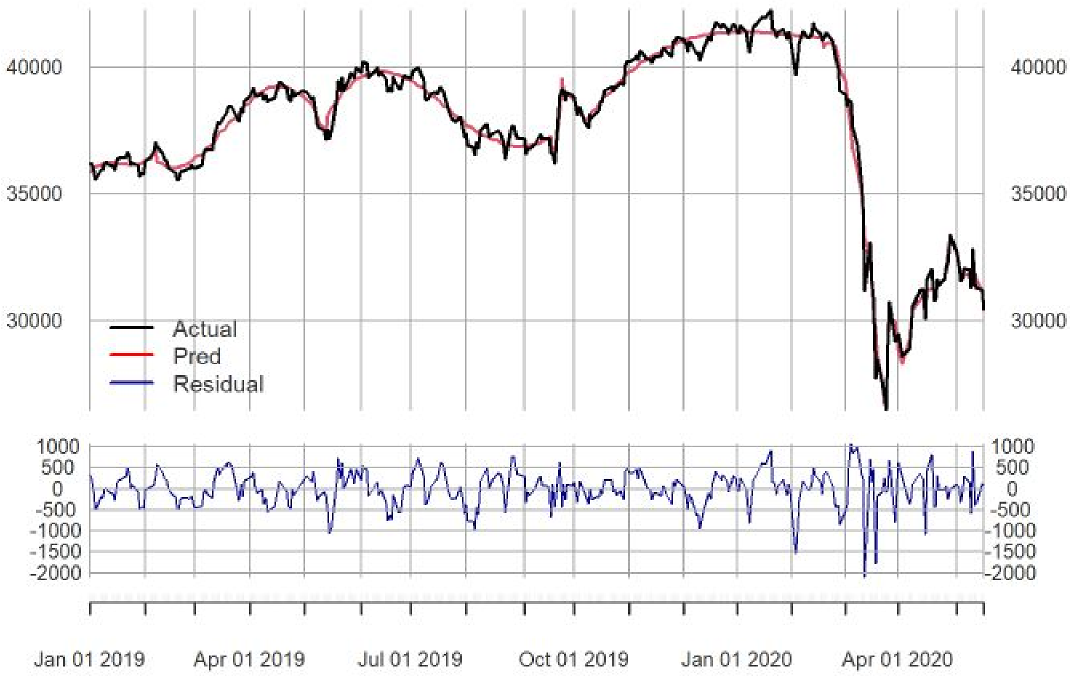

Figure 3.8: Sensex spline regression fit

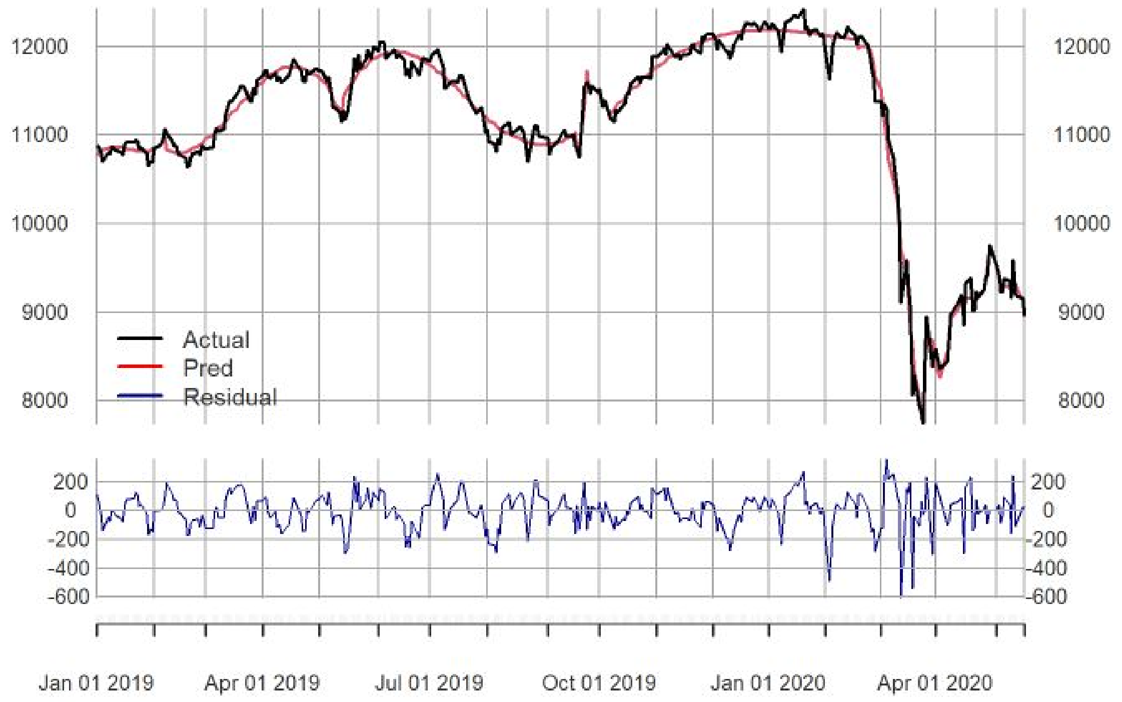

Figure 3.9: Nifty spline regression fit

3.3.3.2 First Order Differentiation

First-order differentiation/derivative of a function measures the sensitivity of the function to change to the independent variable. Here, I differentiated the two fitted graphs obtained after the spline regression. "D()" function in R was used to perform differentiation over the acquired expressions as shown below -

D(exp,"x")

exp - expression to be differentiated
x - variable by which the equation is differentiated

3.3.3.3 Second Order Differentiation

Second-order differentiation/derivative of a function measures the sensitivity of the rate of change of the function to the independent variable. This test indicates the pace at which the stock prices fluctuate within this given timeline. I again used the function "D()" to compute derivatives.

3.3.3.4 Residuals

Applying a model on the initial graph (figure 3.2) obtained by the data points may prove to be insufficient. There is more scope to fit a curve on the residuals of the fitted model than on the actual data. These residuals indicate when the predicted model has a higher value than the actual data and vice versa. It provides excellent accuracy on how and when the model differs from the actual data which will prove to be very useful.

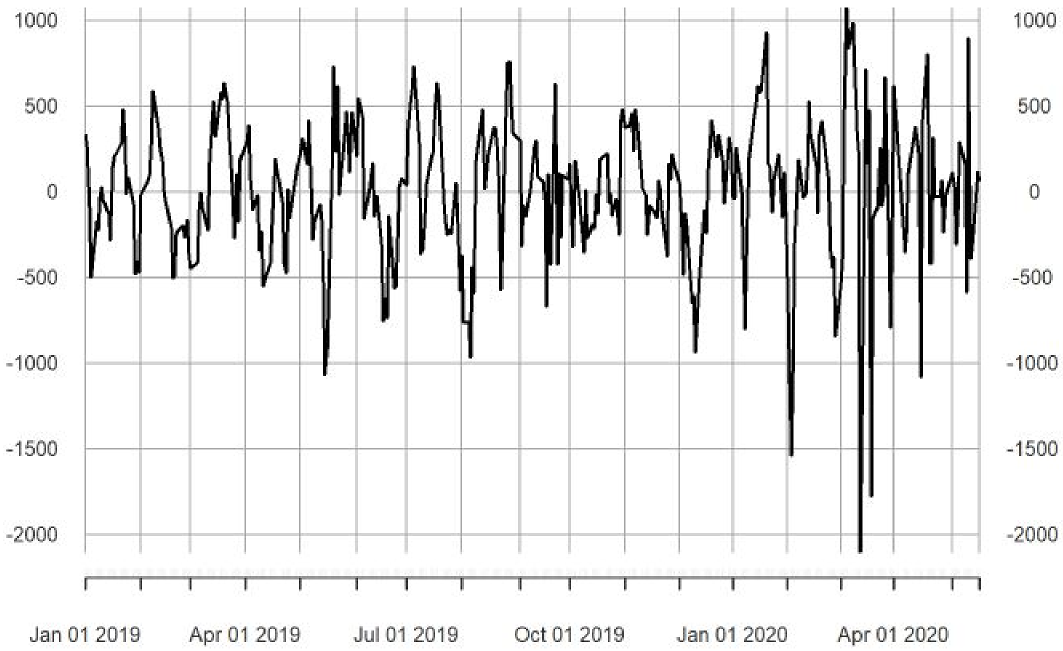

Figure 3.10: Residuals of Sensex

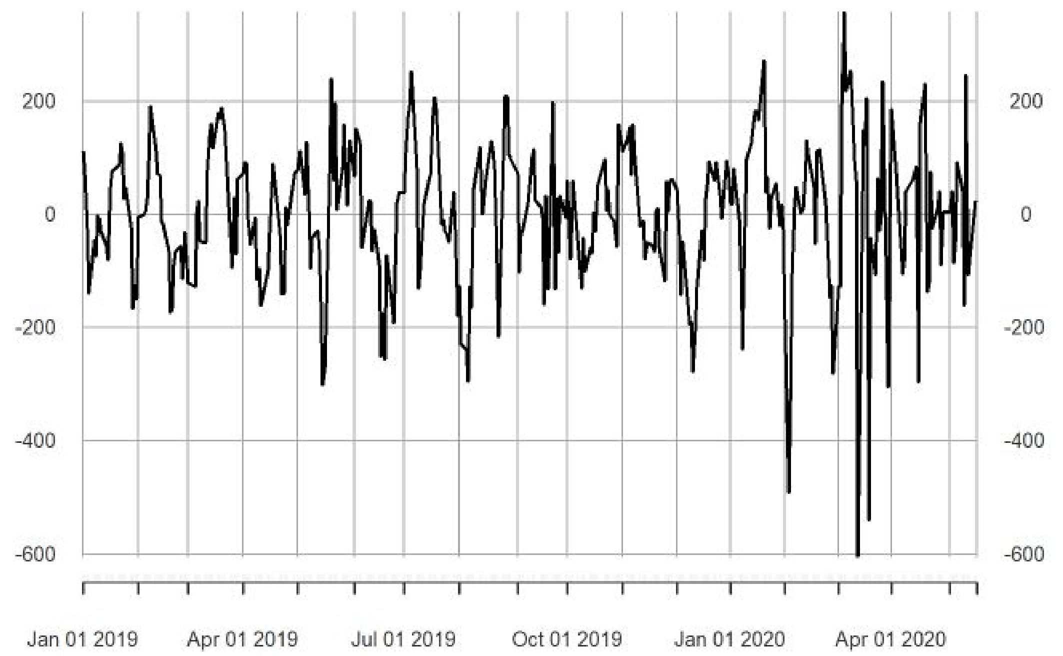

Figure 3.11: Residuals of Nifty

3.3.3.5 ACF and PACF

ACF is an autocorrelation function which returns the correlation of the series with its lagged values considering all components of a time series like seasonality, trend, cyclic, etc. before computation. PACF is a partial autocorrelation function that finds the correlation of the residuals with the lag of the next value. The ACF and PACF plots of this residual, may not be used for finding the AR value or the MA value but can be used to see yet another periodic trend. It can be done in R using the following code -

```
Acf(dataS$Res,100)
# 100 is the lag value.

Pacf(dataS$Res,100)
```

The ACF and PACF plots have been applied on the residuals, as shown in figures 3.10 and 3.11.

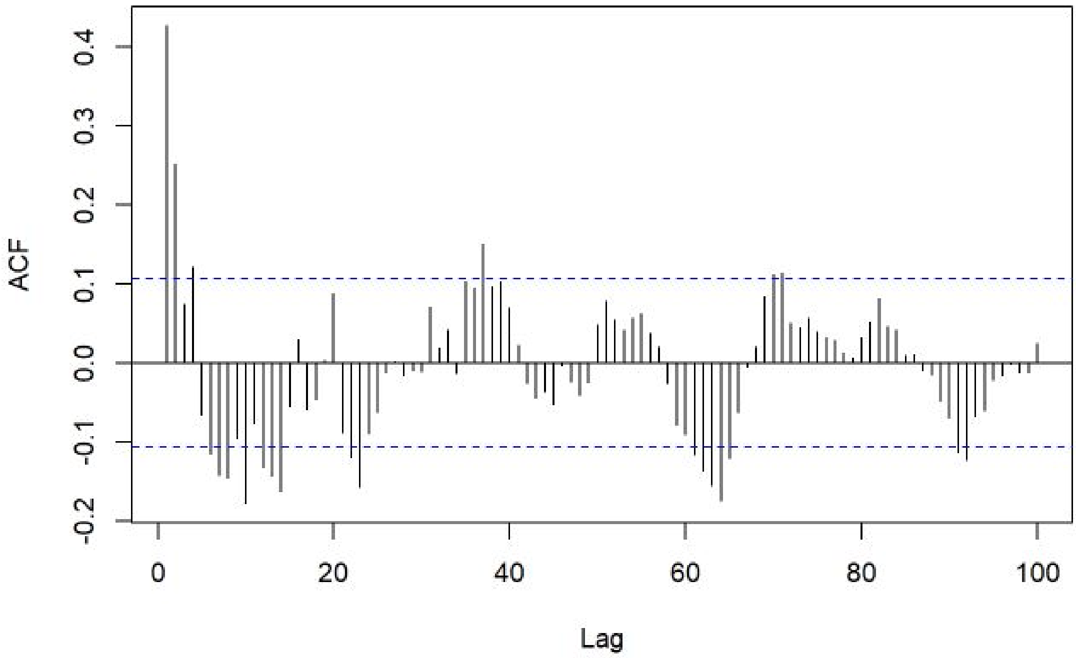

Figure 3.12: ACF plot of Sensex residuals

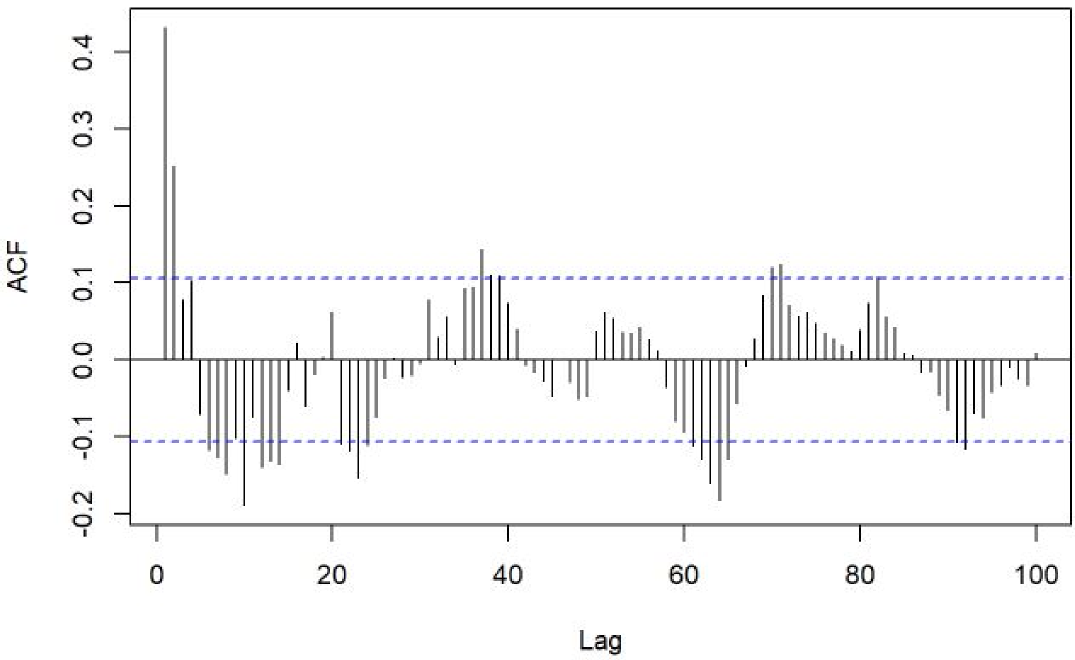

Figure 3.13: ACF plot of Nifty residuals

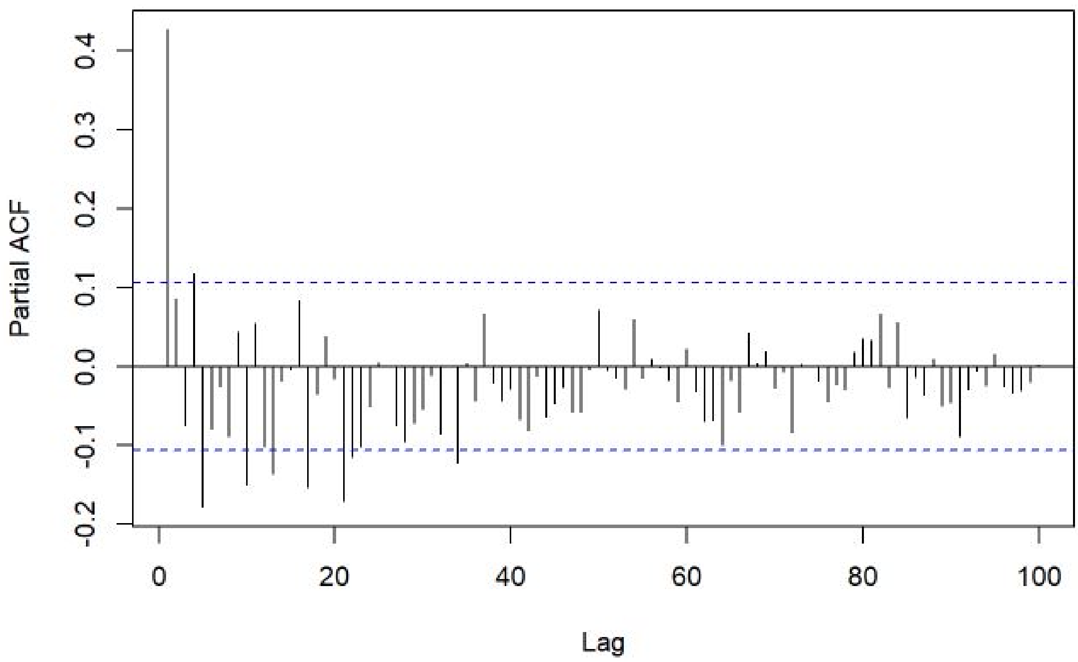

Figure 3.14: PACF plot of Sensex residuals

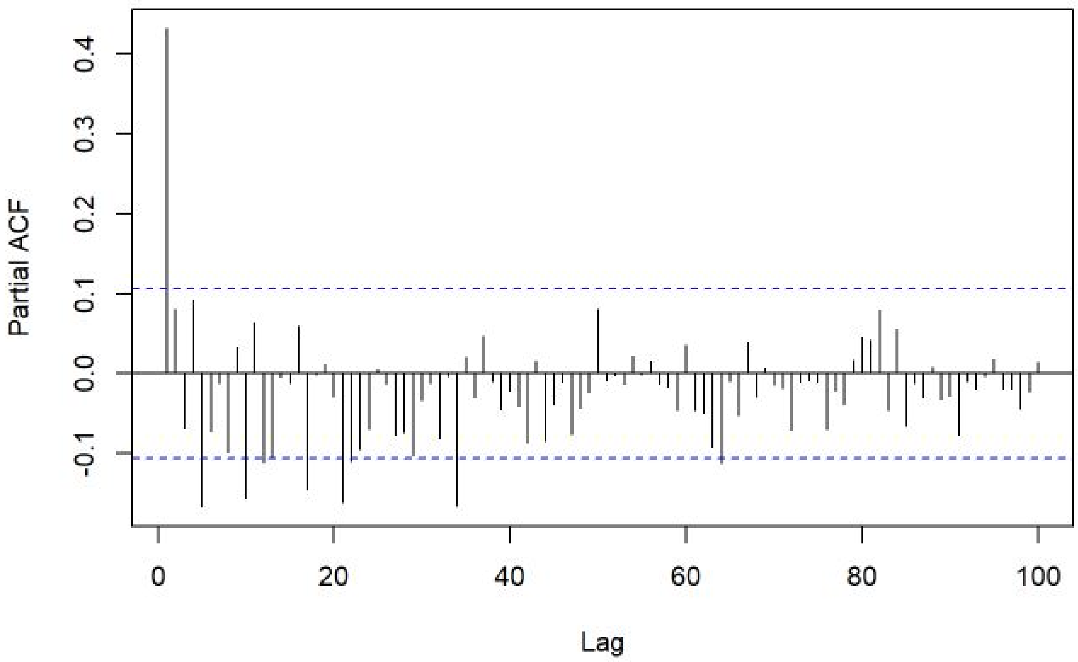

Figure 3.15: PACF plot of Sensex residuals

3.3.3.6 AR(1) Model

An AR model or autoregressive model predicts the behaviour of a time series based on previous values. "AR(p)" model will make a prediction based on the "p" number of past values (number of lagged targets). It is generally used for stationary values. I used "AR(1)" model on the residuals obtained, to create a model where the predicted value will depend only on the previous value and the intercept. Every "AR(1)" model will follow the following general equation -

$$\mathbf{y(t)} = \mu + \phi\mathbf{*y(t\text{-}1)} + \mathbf{e(t)}$$

y(t) - predicted value at time t
y(t-1) - value at time t-1
μ - intercept value
ϕ - coefficient for the lagged value
e(t) - error at time t (error function)

3.3.3.6.1 Significance of ϕ

When ϕ=0, y(t) is a white noise.
When ϕ=1 and w=0, y(t) is equivalent to a random walk.
When ϕ=1 and w is not equal to 0, y(t) is equivalent to a random walk with drift.
When ϕ<0, y(t) tends to oscillate around the mean. In almost every case, 1<ϕ<1.

In R, I used the function "arima()" which is defined as follows -

arima(data,order=c(0,0,0))

data - data for model fitting
order - specifications corresponding to the order of the model

The first integer given to the "order" parameter is the AR order, the second is the degree of difference and the last integer is the MA order. As AR(1) was applied, the second and third integers were set to zero and the first integer was assigned the value one as shown below -

```
ar1_model<-arima(dataS$Res,c(1,0,0))
```

3.3.3.7 GARCH(1,1) Model

Heteroskedasticity describes the irregularity of the error term in a statistical model, and hence the results drawn from such models shall not be accurate all the time. Therefore I used the generalized autoregressive conditional heteroskedasticity (GARCH) process to describe an approach for estimating the volatility as mentioned earlier [9].

GARCH involves the following three steps -
1. Fit the best autoregressive model
2. Compute autocorrelations of the error terms
3. Test for significance

The following equations represent the GARCH(1,1) model -

$$\mathbf{a(t)} = \mathbf{e(t)v(t)}$$

a(t) - predicted function
e(t) - error function
v(t) - standard deviation or volatility of the time series

$$\mathbf{v(t)} = \mathbf{sqrt(}\ \omega + \alpha 1 \mathbf{*a(t\text{-}1)}^2 + \beta 1 \mathbf{*v(t\text{-}1)}^2\ )$$

"ω," "$\alpha 1$" and "$\beta 1$" are constants or coefficients that are unique to the chosen data. In R, I used "ugarchspec" and "ugarchfit" functions (from package "rugarch" [10]) to fit the residual curve. "gsSelect" function (from package "GEVStableGarch" [11]) helps to select the best GARCH model parameters for curve fitting. I applied "gsSelect" function on the generated residual data to obtain the most suitable model. The function is as follows -

gsSelect(data,order.max=c(1,1,1,1),selection.criteria = "AIC")

data - data for processing
selection.criteria - criteria for selecting the most suitable model
order.max - maximum order of the GARCH model to be fitted when searching for

the best model.

The first two integers passed into the "order.max" parameter are the maximum AR and MA orders and the last two integers represent the maximum GARCH parameters‘.

3.3.3.7.1 Selection Criteria

3.3.3.7.1.1 AIC

AIC or Akaike Information Criterion tests the accuracy of a fitted model. The AIC value is the estimate of the amount of information lost by a given model. The AIC equation is as follows [12] -

$$\mathbf{AIC = -2*(loglikelihood) + 2K}$$

K - number of model parameters
loglikelihood - measure of model fit

"Loglikelihood" is directly proportional to the quality of fit.

3.3.3.7.1.2 BIC

BIC or Bayesian Information Criterion also tests goodness of fit of a model. It is similar to AIC as it is also an estimate of the amount of information lost by a given model. The BIC equation is as follows [12] -

$$\mathbf{BIC = k*log(n) - 2*log(L(\theta))}$$

n - sample size
k - number of model parameters
θ - set of all parameters
L(θ) - likelihood of the tested model

I used "gsSelect" to find the best model for Sensex residuals as shown in the following code snippet -

```
library(GEVStableGarch)
gsSelect(dataS$Res,order.max=c(1,1,1,1),cond.dist = "norm",selection.criteria = "
    AIC")
```

```
------------------------------------------------------------------
Best Model:  arma(1, 0) + garch(1, 1)
------------------------------------------------------------------
```

Figure 3.16: Result of "gsSelect"

The function tested all possible models within the range of "order.max" parameter and found "AR(1) + GARCH(1,1)" to be the best model for the data under consideration. The same result is obtained when substituted with Nifty residuals and BIC selection criterion.

3.4 Results

3.4.1 Differentiation

The first order differentiation equations are as follows -
Sensex -

1.	$136.8888 - 10.954319 * (2 * x) + 0.2645245 * (3 * x^2)$
2.	$17.246052 * (2 * x) - 820.8614 - 0.1051567 * (3 * x^2)$
3.	$2179.449 - 16.31802 * (2 * x) + 0.03922962 * (3 * x^2)$
4.	$150576.5 * (2 * x) - 26930540 - 280.6208 * (3 * x^2)$
5.	$2727.266 + 6.828232 * (2 * x) - 0.0528367 * (3 * x^2)$
6.	$1783.014 - 6.709182 * (2 * x) + 0.008373018 * (3 * x^2)$
7.	$331396.4 - 1109.087 * (2 * x) + 1.233731 * (3 * x^2)$
8.	$25018690 - 80975.87 * (2 * x) + 87.36028 * (3 * x^2)$
9.	$1440811 - 4509.379 * (2 * x) + 4.704413 * (3 * x^2)$
10.	$9086.384 * (2 * x) - 3025995 - 9.09456 * (3 * x^2)$

Nifty -

1.	$37.14412 - 3.26023 * (2 * x) + 0.07994356 * (3 * x^2)$
2.	$4.592556 * (2 * x) - 210.8956 - 0.02851471 * (3 * x^2)$
3.	$715.6223 - 5.399194 * (2 * x) + 0.01305044 * (3 * x^2)$
4.	$43281.66 * (2 * x) - 7739206 - 80.67917 * (3 * x^2)$
5.	$61.75447 * (2 * x) - 10084.47 - 0.1247896 * (3 * x^2)$
6.	$464.5525 - 1.73293 * (2 * x) + 0.002137055 * (3 * x^2)$
7.	$88851.48 - 297.0629 * (2 * x) + 0.3300245 * (3 * x^2)$
8.	$7218807 - 23372.34 * (2 * x) + 25.22361 * (3 * x^2)$
9.	$458817 - 1435.558 * (2 * x) + 1.497179 * (3 * x^2)$
10.	$2546.241 * (2 * x) - 848241.8 - 2.547684 * (3 * x^2)$

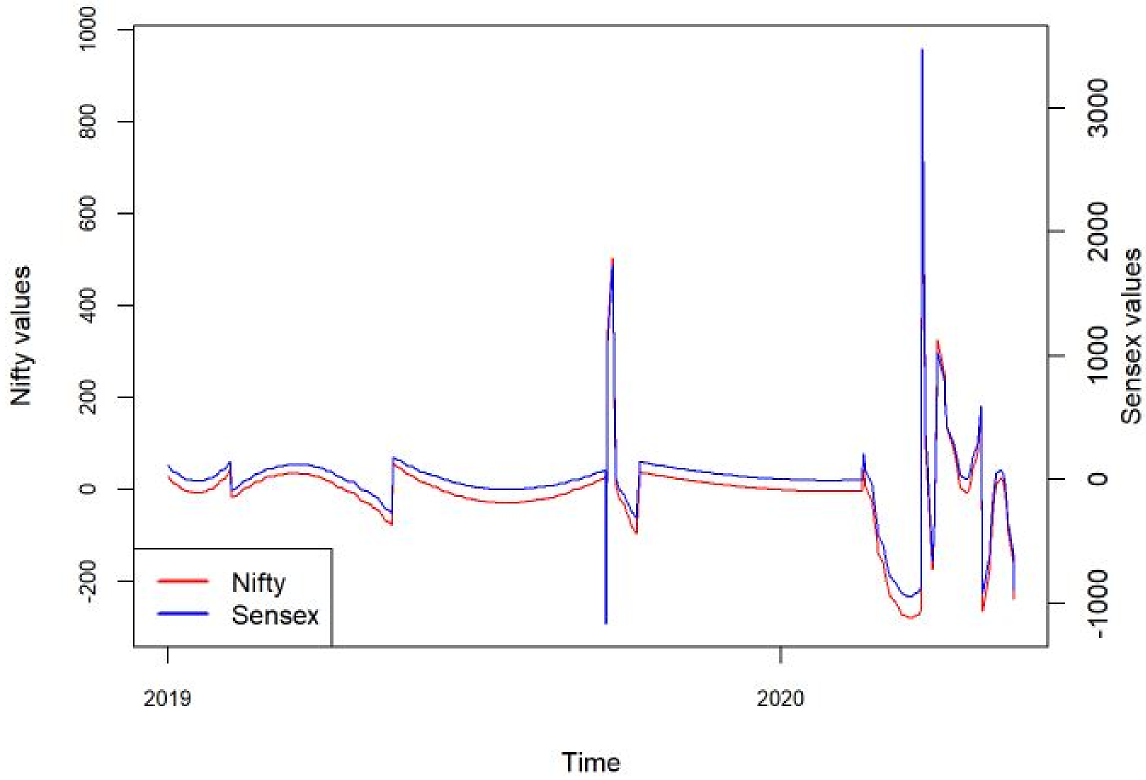

Figure 3.17: First order differentiation

The second order differentiation equations are as follows -
Sensex -

1.	$0.2645245 * (3 * (2 * x)) - 10.954319 * 2$
2.	$17.246052 * 2 - 0.1051567 * (3 * (2 * x))$
3.	$0.03922962 * (3 * (2 * x)) - 16.31802 * 2$
4.	$150576.5 * 2 - 280.6208 * (3 * (2 * x))$
5.	$6.828232 * 2 - 0.0528367 * (3 * (2 * x))$
6.	$0.008373018 * (3 * (2 * x)) - 6.709182 * 2$
7.	$1.233731 * (3 * (2 * x)) - 1109.087 * 2$
8.	$87.36028 * (3 * (2 * x)) - 80975.87 * 2$
9.	$4.704413 * (3 * (2 * x)) - 4509.379 * 2$
10.	$9086.384 * 2 - 9.09456 * (3 * (2 * x))$

Nifty -

1.	$0.07994356 * (3 * (2 * x)) - 3.26023 * 2$
2.	$4.592556 * 2 - 0.02851471 * (3 * (2 * x))$
3.	$0.01305044 * (3 * (2 * x)) - 5.399194 * 2$
4.	$43281.66 * 2 - 80.67917 * (3 * (2 * x))$
5.	$61.75447 * 2 - 0.1247896 * (3 * (2 * x))$
6.	$0.002137055 * (3 * (2 * x)) - 1.73293 * 2$
7.	$0.3300245 * (3 * (2 * x)) - 297.0629 * 2$
8.	$25.22361 * (3 * (2 * x)) - 23372.34 * 2$
9.	$1.497179 * (3 * (2 * x)) - 1435.558 * 2$
10.	$2546.241 * 2 - 2.547684 * (3 * (2 * x))$

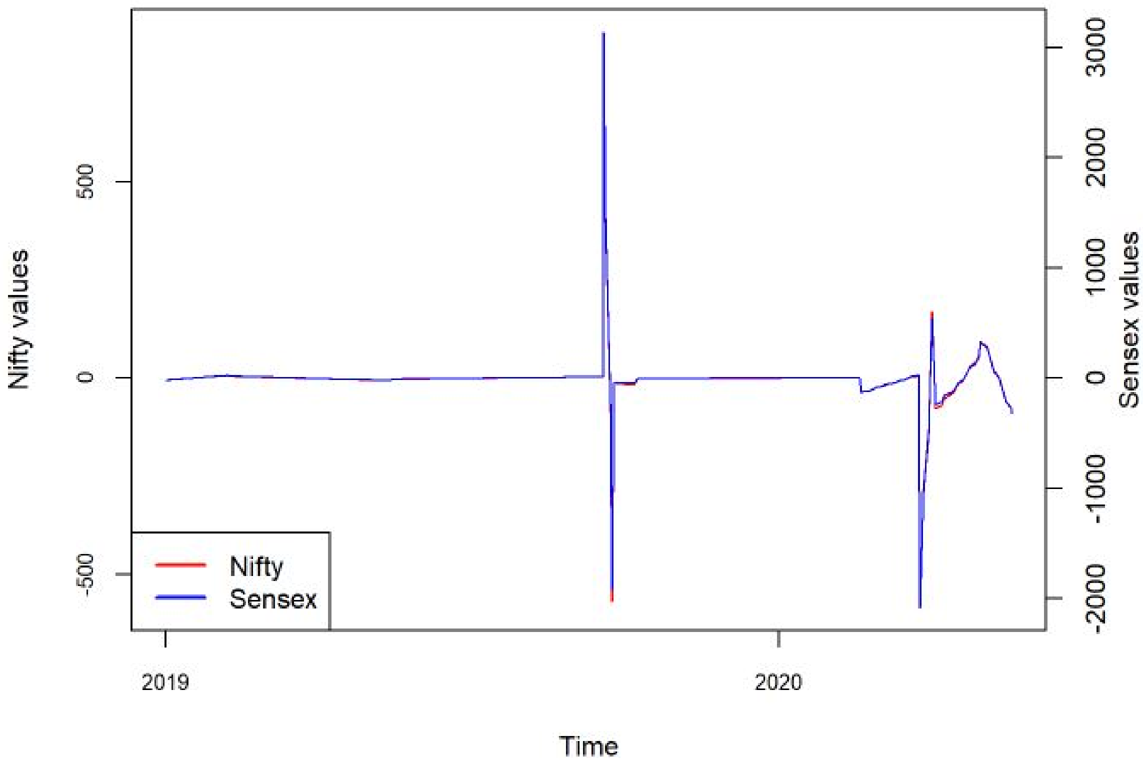

Figure 3.18: Second order differentiation

3.4.2 GARCH(1,1) with AR(1)

I created a GARCH object using "ugarchspec." The mean model selected was "armaOrder(1,0,0)," indicating AR(1) and the variance model was "garch(1,1)," indicating GARCH(1,1). The code snippet for the above is as follows -

```
spec<-ugarchspec(variance.model = list(garchOrder=c(1,1)),mean.model = list(
    armaOrder=c(1,0,0)))
spec_fitS<-ugarchfit(spec,dataS$Res)
spec_fitS
```

```
Conditional Variance Dynamics
-----------------------------------
GARCH Model  : sGARCH(1,1)
Mean Model   : ARFIMA(1,0,0)
Distribution : norm

Optimal Parameters
------------------------------------
         Estimate  Std. Error  t value Pr(>|t|)
mu      -13.21717  4.0818e+01 -0.32381 0.746085
ar1       0.61493  4.8929e-02 12.56761 0.000000
omega  3424.07822  3.3620e+03  1.01848 0.308450
alpha1    0.16351  6.2257e-02  2.62634 0.008631
beta1     0.83549  6.6092e-02 12.64139 0.000000

Robust Standard Errors:
         Estimate  Std. Error  t value Pr(>|t|)
mu      -13.21717  4.3344e+01 -0.30494 0.760414
ar1       0.61493  6.4083e-02  9.59578 0.000000
omega  3424.07822  5.7838e+03  0.59201 0.553842
alpha1    0.16351  9.1733e-02  1.78243 0.074679
beta1     0.83549  1.1018e-01  7.58328 0.000000

LogLikelihood : -2451.561

Information Criteria
------------------------------------

Akaike       14.493
Bayes        14.549
Shibata      14.493
Hannan-Quinn 14.515
```

Figure 3.19: GARCH and AR coefficients (Sensex)

```
Conditional Variance Dynamics
-----------------------------------
GARCH Model  : sGARCH(1,1)
Mean Model   : ARFIMA(1,0,0)
Distribution : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error  t value Pr(>|t|)
mu      -1.84751   12.503640 -0.14776 0.882534
ar1      0.60398    0.049643 12.16636 0.000000
omega  643.78612  447.791659  1.43769 0.150522
alpha1   0.16296    0.063241  2.57677 0.009973
beta1    0.79934    0.081336  9.82767 0.000000

Robust Standard Errors:
        Estimate  Std. Error  t value Pr(>|t|)
mu      -1.84751   12.948073 -0.14269 0.886538
ar1      0.60398    0.059222 10.19856 0.000000
omega  643.78612  827.357423  0.77812 0.436496
alpha1   0.16296    0.081814  1.99181 0.046392
beta1    0.79934    0.144032  5.54973 0.000000

LogLikelihood : -2051.399

Information Criteria
------------------------------------

Akaike       12.132
Bayes        12.189
Shibata      12.132
Hannan-Quinn 12.155
```

Figure 3.20: GARCH and AR coefficients (Nifty)

Five coefficients were obtained from the "ugarchfit" model namely, "ω," "α1," "β1," "ar1" and "mu." Out of the five, "mu" was the only coefficient to be rejected because of a very high p-value. A very high p-value indicates that the null hypothesis was accepted, implying insignificance of the associated coefficient. The "ar1" coefficient gave the "ϕ" value from the "ugarchfit" model. Since the "μ" value was rejected, I ignored it in the equations. I also ignored the error function as it was insignificant. Finally the following results were obtained -

Sensex AR equation of the generated model -

$$\mathbf{y(t) = 0.61493*y(t\text{-}1)}$$

Sensex GARCH equation of the generated model -

$$\mathbf{v(t) = sqrt(3424.07822 + 0.16351*y(t\text{-}1)^2 + 0.83549*v(t\text{-}1)^2)}$$

Nifty AR equation of the generated model -

$$\mathbf{y(t) = 0.60398*y(t\text{-}1)}$$

Nifty GARCH equation of the generated model -

$$\mathbf{v(t) = sqrt(643.78612 + 0.16296*y(t\text{-}1)^2 + 0.79934*v(t\text{-}1)^2)}$$

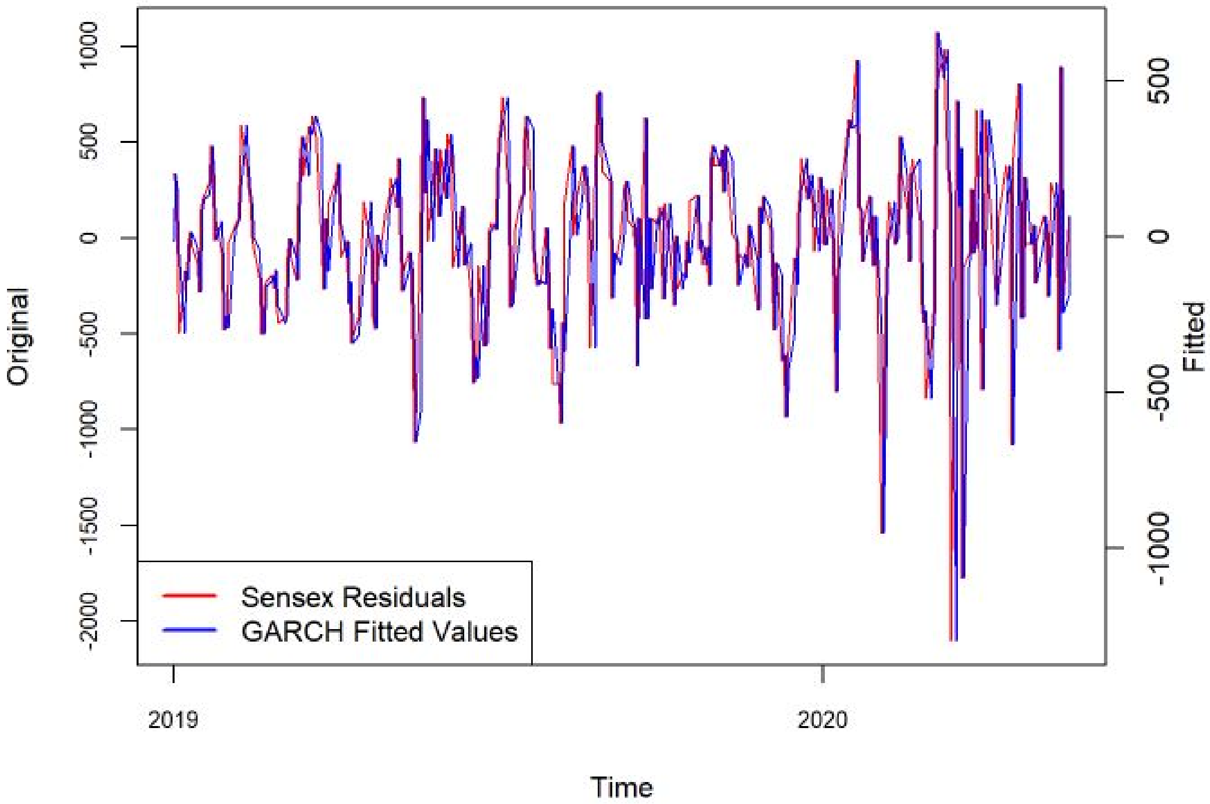

Figure 3.21: GARCH Sensex residual fit

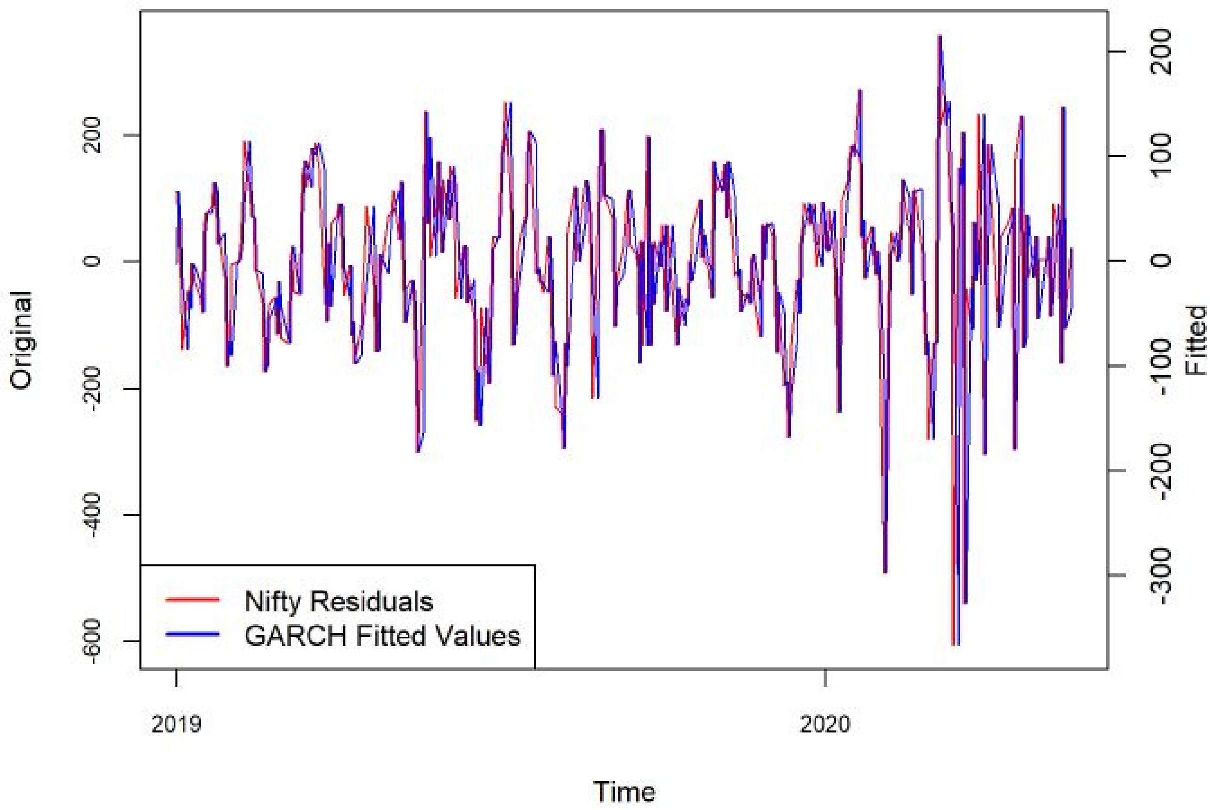

Figure 3.22: GARCH Nifty residual fit

Figures 3.21 and 3.22 depict the GARCH fitted values in conjunction with the generated residuals. It was observed that the model fits accurately for both Nifty & Sensex values and all necessary equations were obtained. Since the AIC and BIC values were low (around 12), it further supports the model. Hence "AR(1) + GARCH(1,1)" concluded to be the residual fit.

Since the residuals of the spline regression model have fitted well, it will be easier to predict future differences of the actual data. It is not possible to estimate the stock value on any given day accurately. Still, in the forthcoming days, it will be easier to tell how the actual value shall differ from the predicted value using the above equations. I conclude from the results that COVID-19 had a visible impact on the stock market, which can be captured using the generated models.

Chapter 4

Conclusion

Both Spoken Tutorial scriptwriting and case study project on the effects of COVID-19 on Indian Stock Market have contributed to promoting the usage of R FLOSS. The newly created Spoken Tutorials scripts shall be a part of R tutorial series on Machine Learning. It will help AI enthusiasts in learning practical machine learning skills using R. The stock market project can help various researchers to visualize variations of the market and observe changes in future trends. The case study results proved to be accurate by the information criterion. Scope of further research can be in the creation of an automatic function to find the most appropriate knot points instead of manual selection.

The entire FOSSEE fellowship experience was very informative and enjoyable. Every fellow learned new skills and methods which he/she can make use of in the future. Even though the fellowship was conducted remotely, it didn't hinder the experience and interactions between the fellows and instructors. Overall each fellow learned the different facets of working in an organization while contributing to the society.

References

[1] W. N. Venables and B. D. Ripley, "Modern Applied Statistics with S," *Statistics and Computing, Springer*, 2002.

[2] K. H. Torsten Hothorn and A. Zeileis, "Unbiased Recursive Partitioning: A Conditional Inference Framework," *Journal of Computational and Graphical Statistics*, 2006.

[3] M. Kuhn, "Caret: Classification and Regression Training," *ASCL*, pp. ascl–1505, 2015.

[4] A. Liaw and M. Wiener, "Classification and Regression by randomForest," *R News*, vol. 2, no. 3, pp. 18–22, 2002.

[5] G. O. India, "Sensex open data," `https://www.bseindia.com/Indices/IndexArchiveData.html`,06 2020.

[6] G. O. India, "Nifty open data," `https://www1.nseindia.com/products/content/equities/indices/historical_index_data.htm`,06 2020.

[7] J. A. Ryan and J. M. Ulrich, "quantmod: Quantitative Financial Modelling Framework," *R package version 0.4.17*, 2020.

[8] C. M. Bishop, "Pattern Recognition and Machine Learning," 2006.

[9] A. V. Metcalfe and P. S. Cowpertwait, "Introductory Time Series with R," 2009.

[10] A. Ghalanos, "rugarch: Univariate GARCH models," *R package version 1.4-2*, 2020.

[11] A. T. do Rego Sousa, C. E. G. Otiniano, and S. R. C. Lopes, "GEVStable-Garch," 2020.

[12] J. Friedman, T. Hastie, and R. Tibshirani, "The Elements of Statistical Learning," vol. 1, no. 10, 2006.