Ethereum Price Prediction

MBA652A – Statistical Modelling for Business Analytics

Project Report

Submitted by:

Ashwani Prabhakar (18114006)

Danish Nawaz(18114008)

Dhruv Patel(18114010)

Rohit Gupta(18114020)

Guided By:

Prof. Devlina Chatterjee

Contents

Acknowledgement:

We are highly indebted to Prof. Devlina Chatterjee, for her guidance and continuous support in completing this project. It is because of the knowledge and skills acquired during the course work, along with her comprehensive style of teaching, that we are able to understand the subject in a better way and are able to complete this modelling project successfully.

Declaration:

This is to certify that the project report entitled ‘Ethereum price prediction’ is based on our original research work. Our indebtedness to other works, studies and publicationshave been duly acknowledge at the relevant places.

Dhruv Patel	Danish Nawaz
(IME MTech)	(IME MTech)
Ashwani Prabhakar	Rohit Gupta
(IME MTech)	(IME MTech)

Introduction:

The Ethereum closing price value is influenced by various factors. It is mostly influenced by Supply Demand, Investors Sentiments and other internal factors. Here we will try to predict the closing prices of Ethereum using various time-series models and also using various Independent variables available in data such as high prices in previous days. Our approach will be to create various time-series models with best parameters and come up with an overall comparison of model using test set to decide the best model.

Objective:

The objective of the project is to learn how to apply the time-series models on data and come up with the best model using various accuracy measures using theory learned in class.

Data Source:

https://coinmarketcap.com/currencies/ethereum/historical-data/?start=20130428&end=20190413

The dataset we have chosen is historical dataset of Ethereum price for different days.

Variables:

1)Dependent variable(Y):

- **Closing price of Currency:** It is a continuous variable whose past 4-year data is available.We will try to predict next 45 day closing value and compare it with our available test data.

2)Independent variables(X):

Time: Since we are doing time-series analysis, time is the only independent variable in our model building.

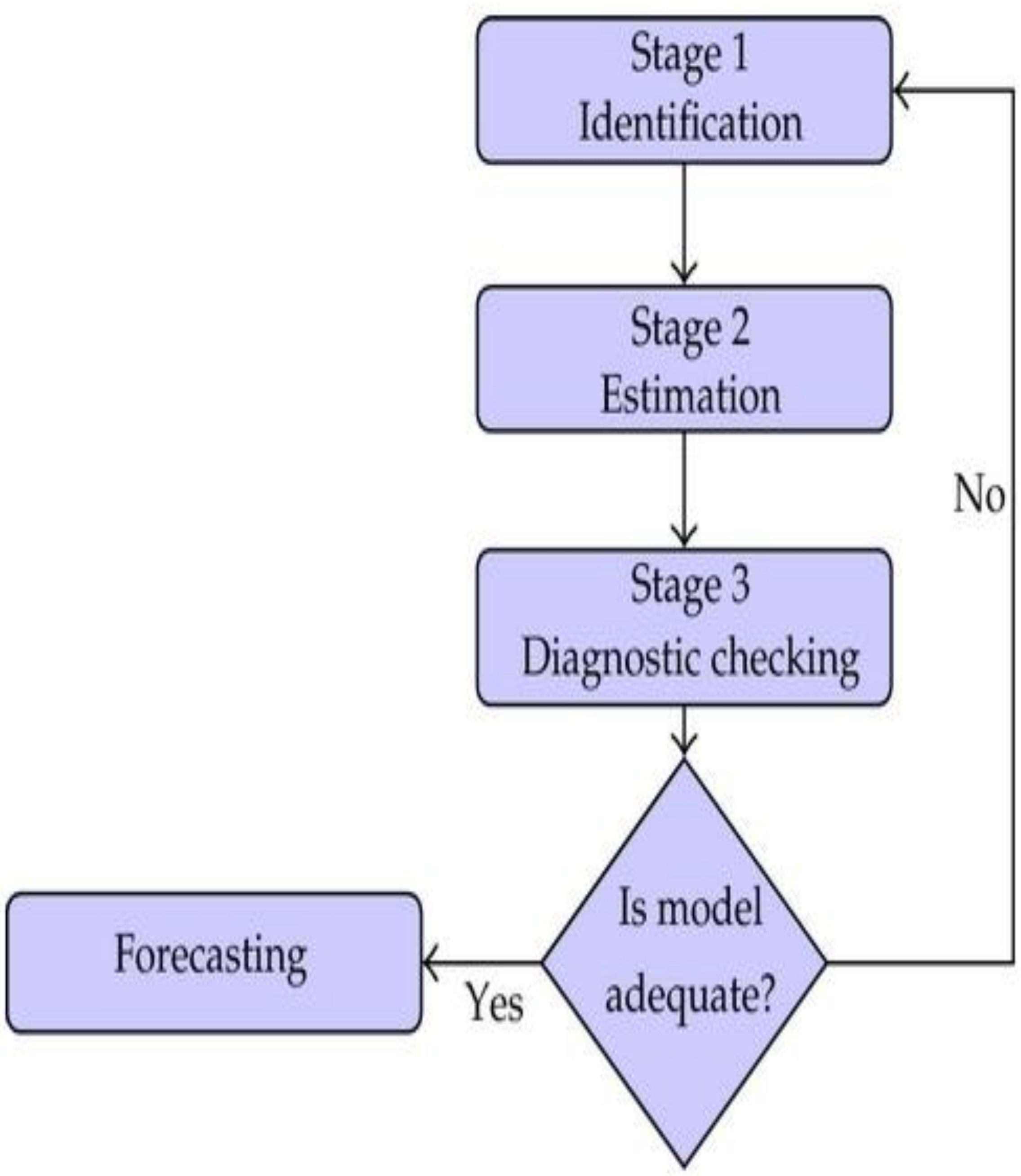
-We will also try to use below available time-series in some of our models.

- **Open-**Continuous, it is Opening price of the Ethereum on that day.
- **Volume-** Continuous, it is the number of Ethereum traded on that day.
- **High-** Continuous, it is the highest price of Ethereum on that particular day.
- **Low-** Continuous, it is the lowest price of Ethereum on that particular day
- **Market Cap**- Continuous, it is the total value of all Ethereum Volume available in market at the end of the day.

Data preprocessing:

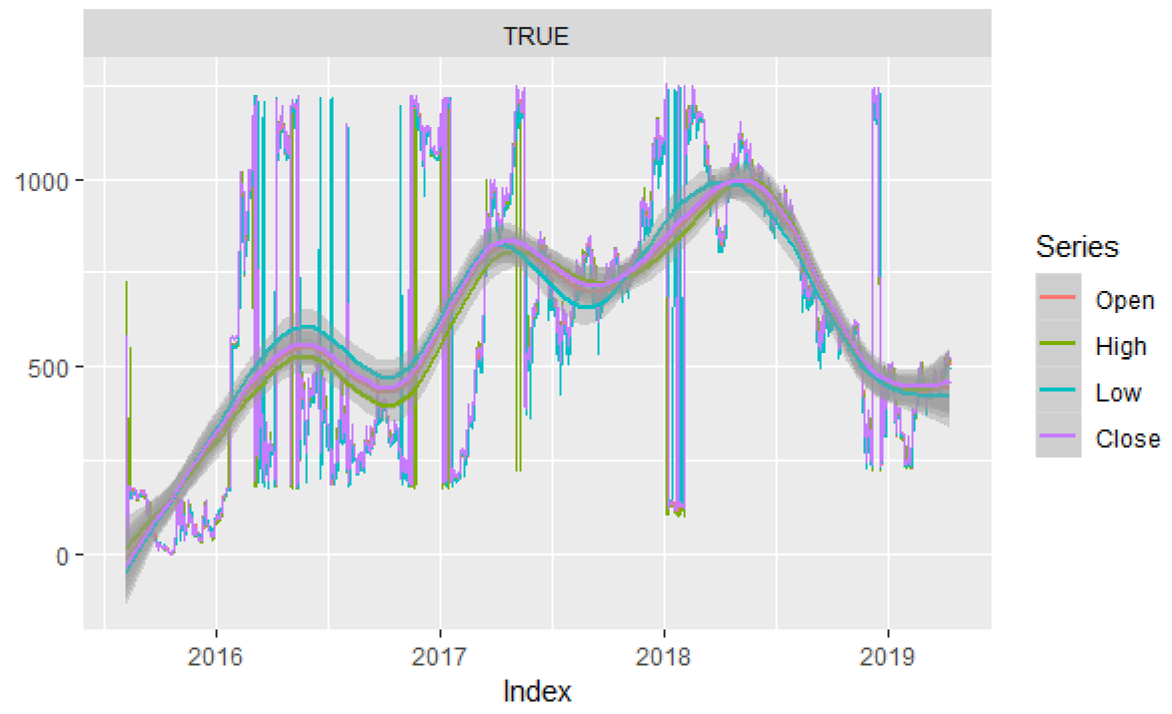
- First, we analyzed that the format of date was different and was unordered so We started with changing the format from dd-month-year format into numeric formatdd-mm-yyyy format such that we could order the date in ascending order for applying time- series models.
- Then while plotting summary of data we saw that the data was factored and also the variables volume and market capital have had commas inside the data, so we started with removing the commas from data using library gsubfun where we substituted the commas with null or no value .
- As the data was in factor form, we converted all the variables except Date into numeric or continuous variable.

- Then we transformed the Close variable in dataset into a time-series object which created a list like object to store various values of particular date.
- **BOX-JENKINS Methodology:**



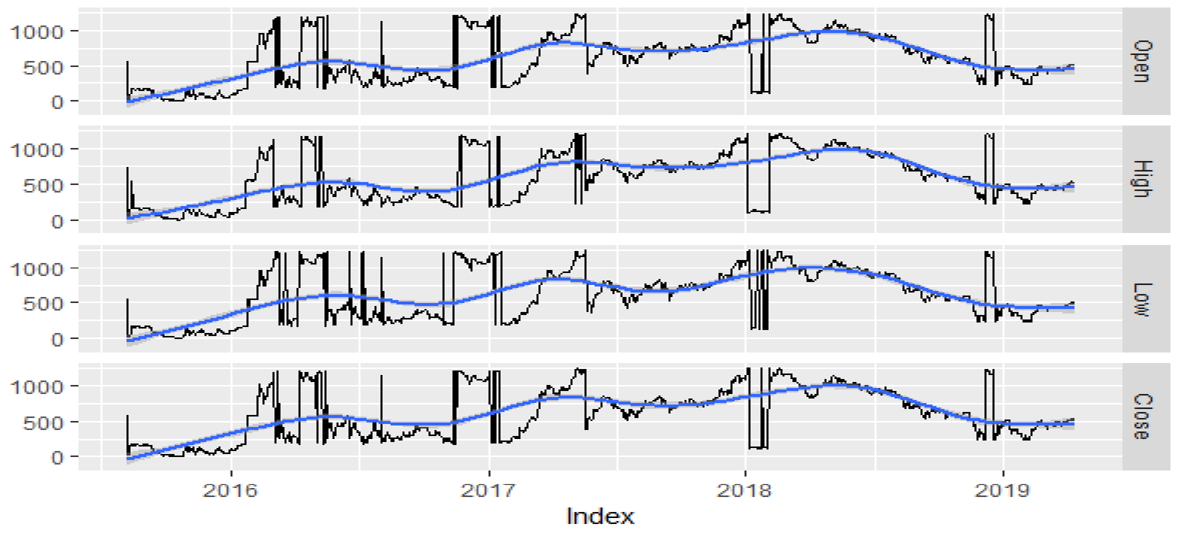
Exploratory Data Analysis:

- **Plotting all the plots together with smoothing to see the difference between various variables and further using in model.**



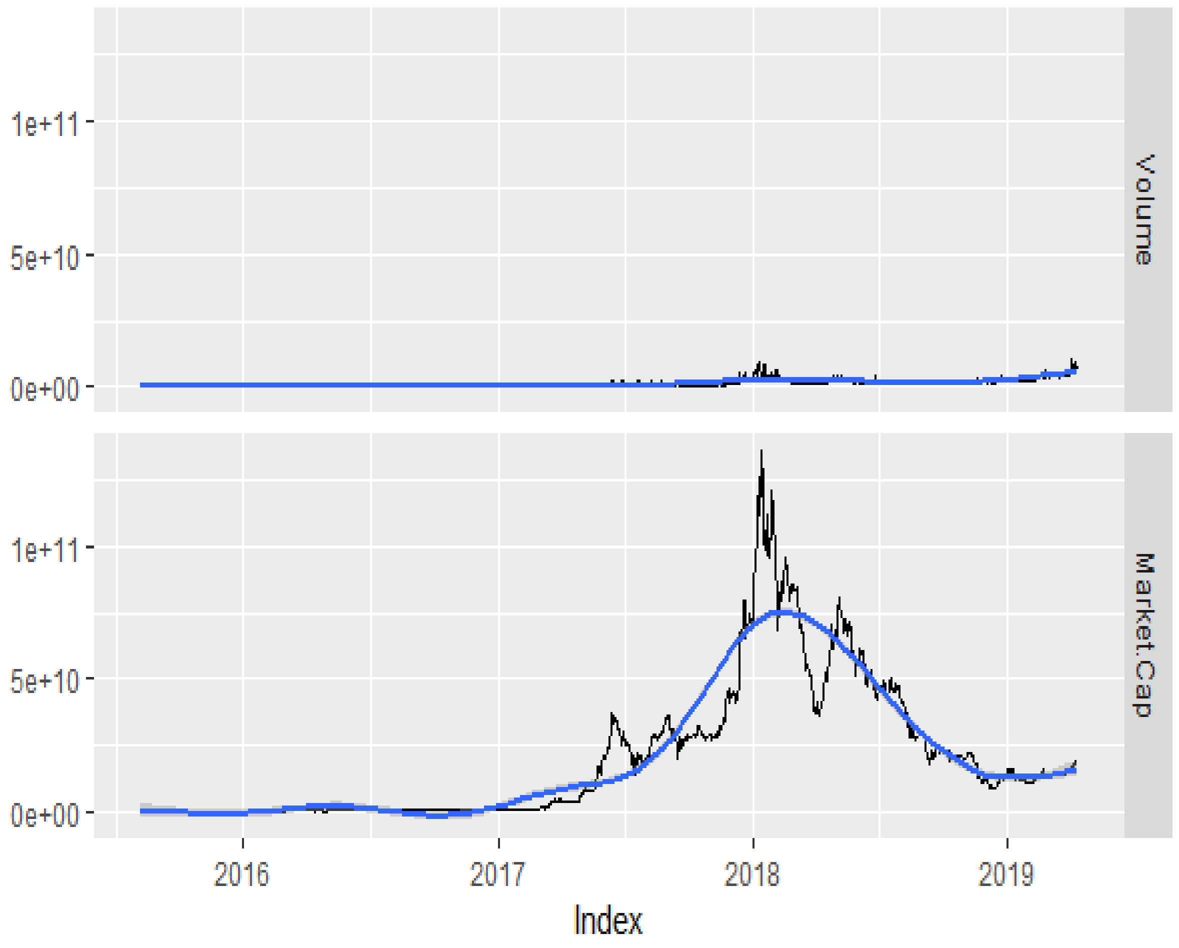
  - From plot it can be seen that all the 4 variables plot are similar which means each variables explain each other so we choose only close variable for fitting in our model .

- **Plot for viewing variability among different attributes.**



- This plot is similar to above plot it’s just for better visualization of relations among the variables.

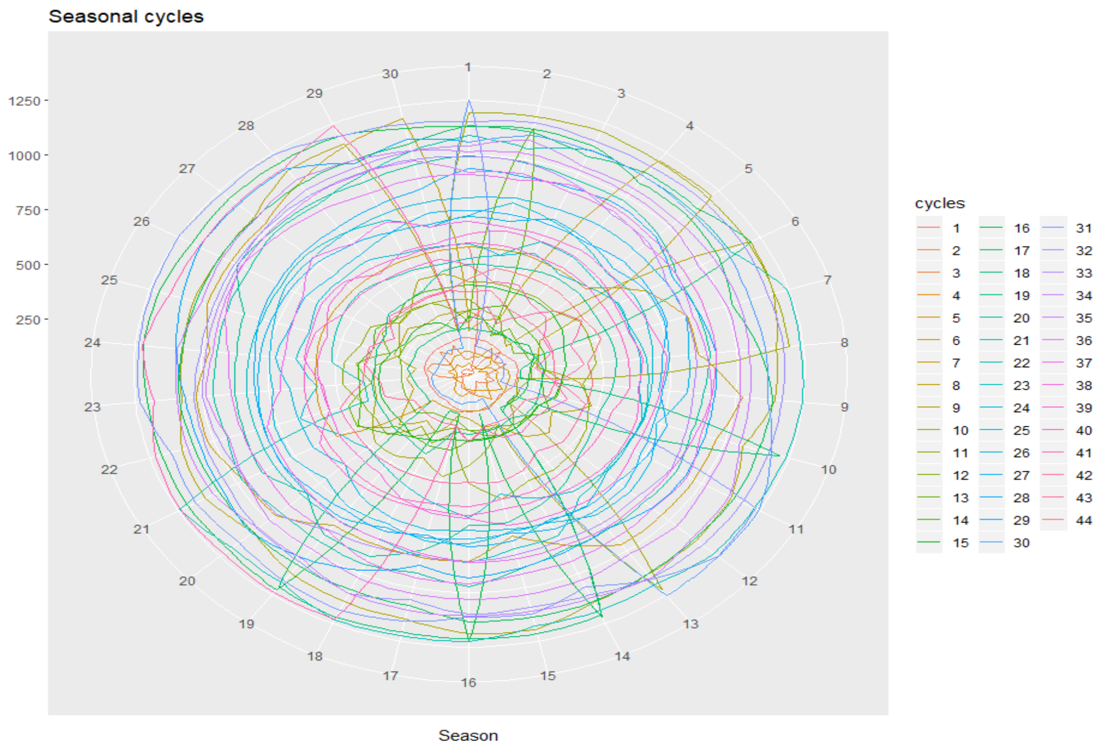
- **Plot for volume and market capital with time**



.

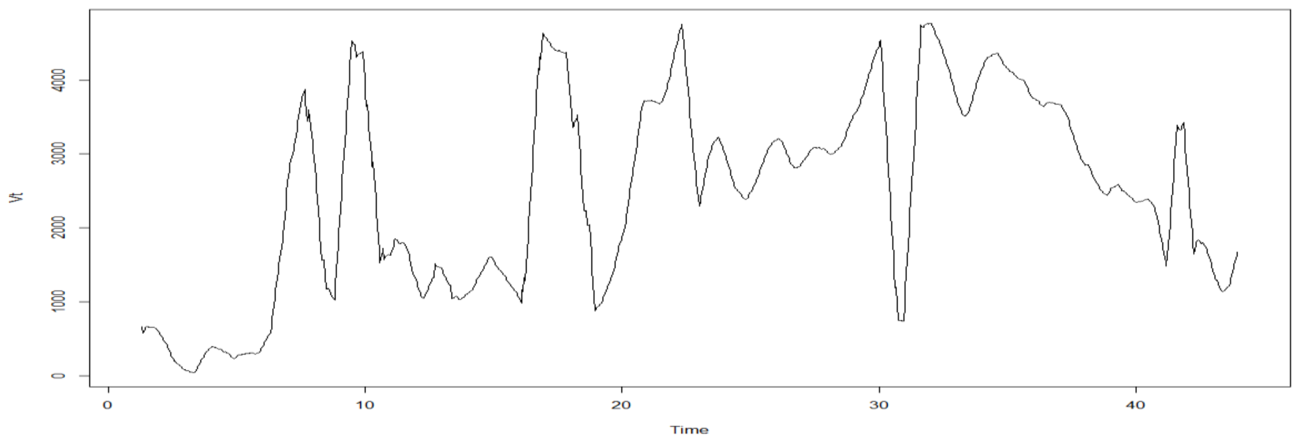
Seasonality Check:

In timeseries analysis, it is very essential to know about trend and seasonality of data before applying any model on data. As we saw in earlier plot of closing value vs time, there is very little trend in data. Here we will make some more plots to make an accurate check on seasonality.



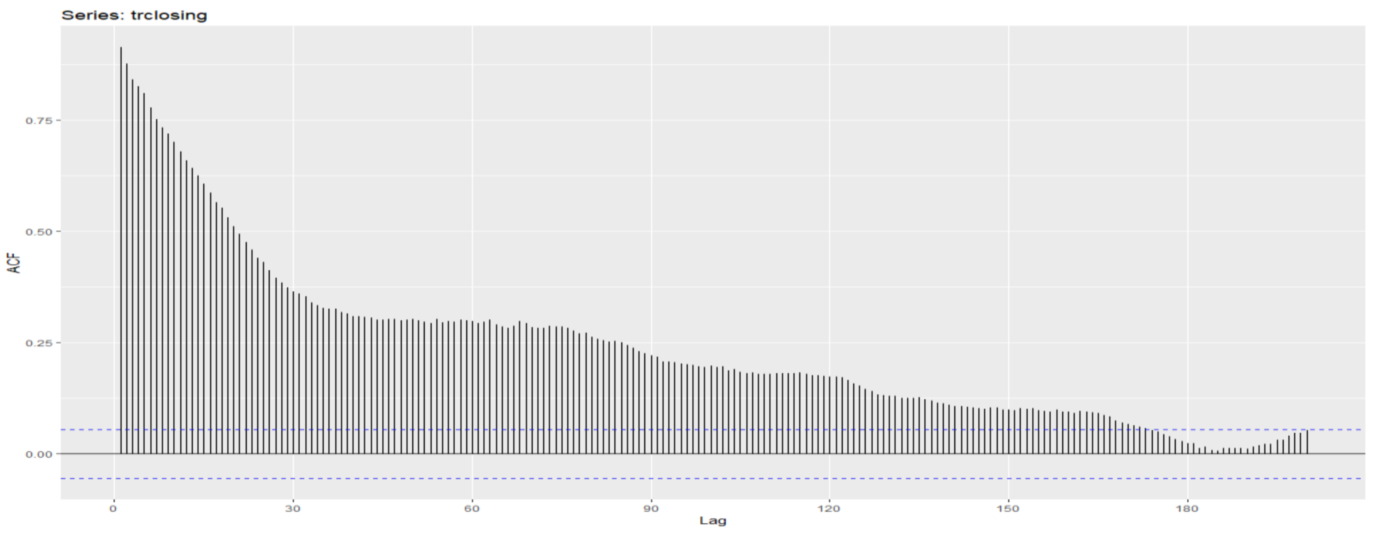
Here we have considered every 30 days a cycle and divided the time span in 44 parts. As we can see there is no clear ups/ down on any of the periods. We plotted similar plots for different cycle period and they also showed no clear sign of seasonality.

Then we tried to smooth the data by moving order smoothing to remove irregular fluctuations and to get the clear pattern of the data, as plotted below. This also confirms no sign of seasonality.

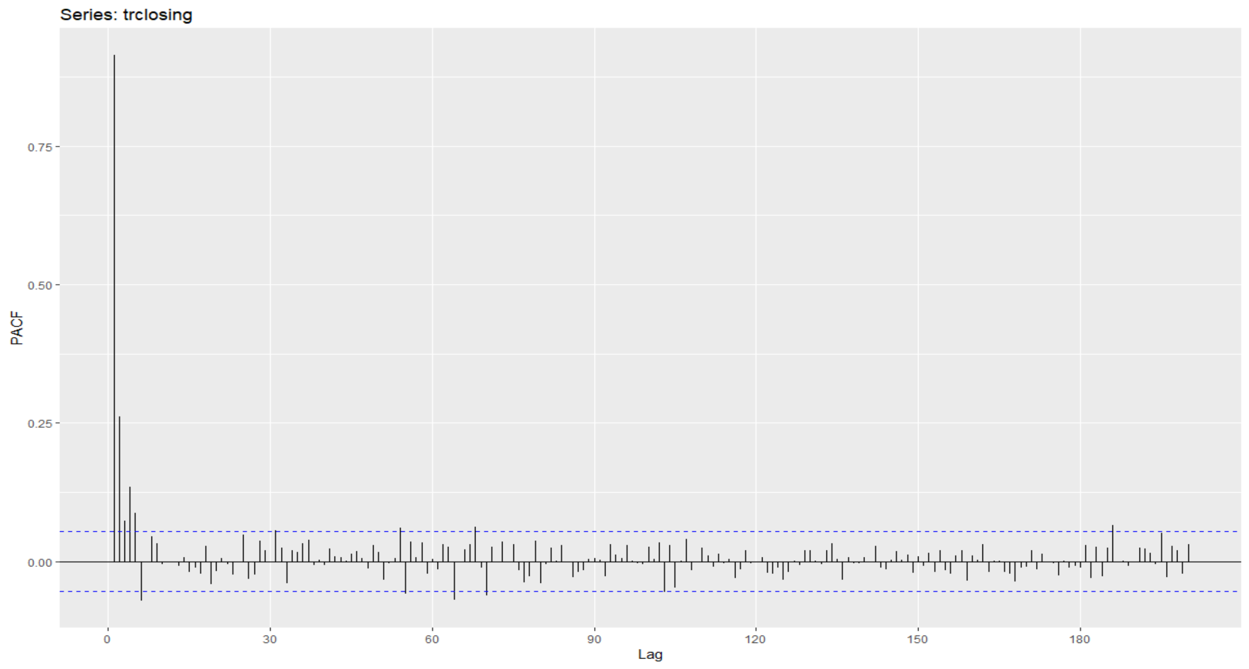


ACF and PACF Plots:

-ACF plot:



-PACF plot:



The autocorrelation function (ACF) measures how a series is correlated with itself at different lags. if your data is strongly seasonal, your peaks will coincide with the seasonality period, which is not the case here which again confirms about non-seasonality of data.

Unlike ACF, PACF takes into consideration the correlation between a time series and each of its lagged values after removing the effect of intermediate values between them. So, it is more useful to detect the ORDER of a autoregressive model.

Box-LjungTest:

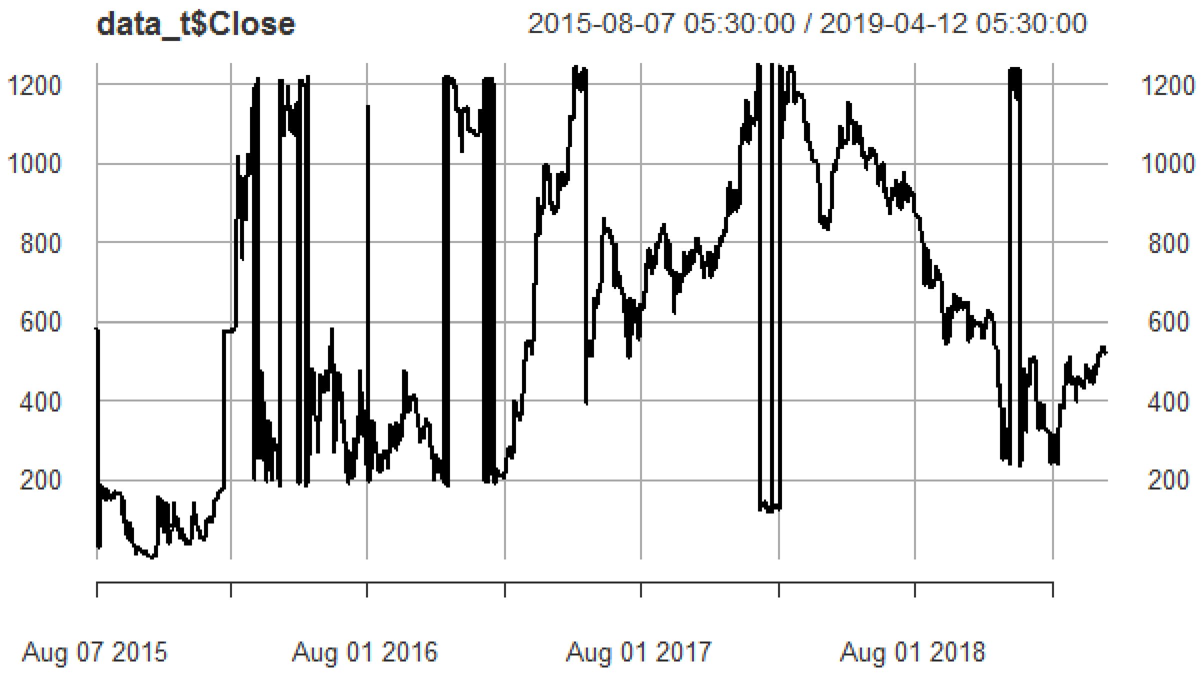
We took 10 lags and examined if there is any autocorrelation or not, null Hypothesis being there is not auto correlation . Results are:

Chi-squared = 8632.3, df = 10, p-value < 2.2e-16

P-value is very low so we reject null hypothesis of no auto correlation.

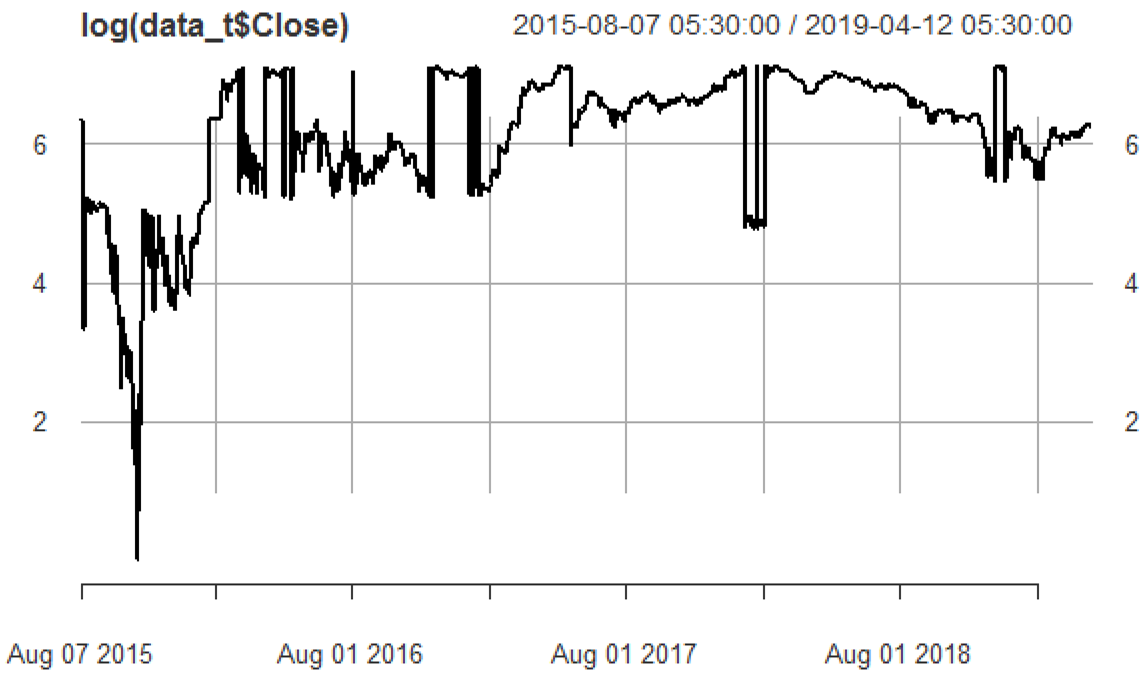
Stationarity check:

- First we started with plotting the closing price with date to look at the stationarity.



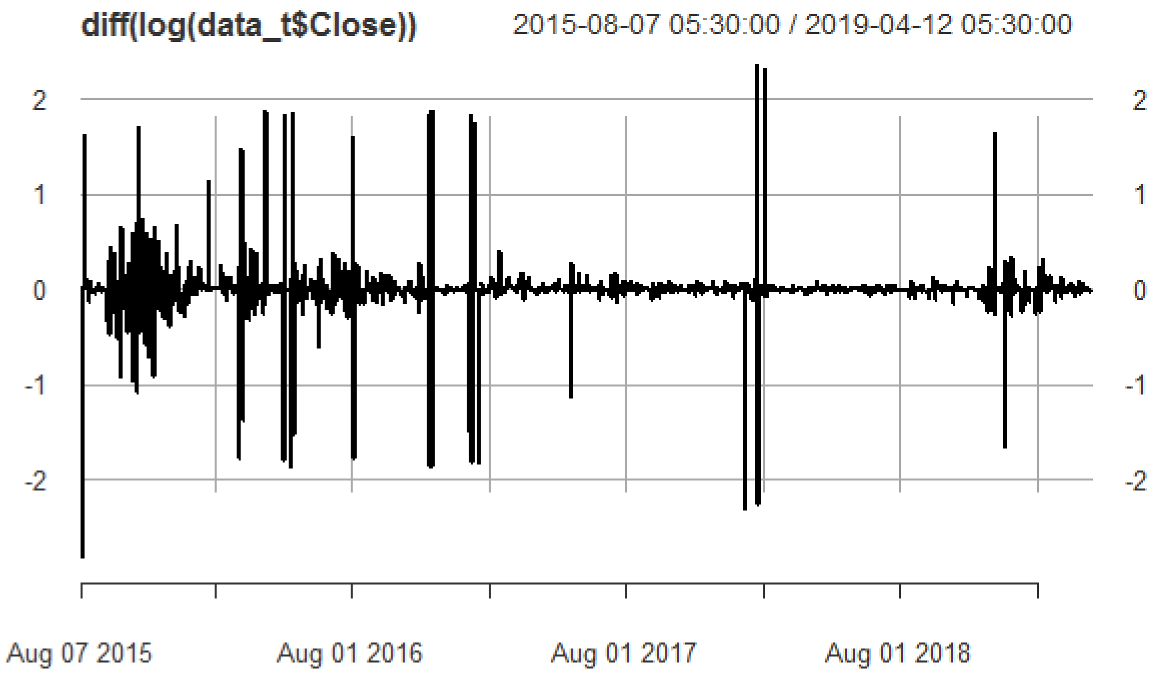
- From this plot we can see that there is no stationarity in data that is mean, variance is not constant so we applied some changes over the variable for making data stationary.

- **Log(Close) stationarity plot :**



  - From plot it can be seen that there is still non-stationarity in the data as there s large deviation of mean and variance of data so we reject this too for modelling.

- **Difference of log of closed:**



- From plot it can be seen that plot has stationarity as mean and variance is constant, so we will choose difference of log while further modelling.

- We performed dicky-fuller test for stationarity and analyzed different p-values with alternative hypothesisbeing: stationary.

Index no.	Variable	P-Value	Dickey-fuller value
1	Close	.99	1.0896
2	Difference(close)	.2295	-2.8241
3	log of close	.99	.56
4	difference ofLog of close	.08411	-3.2221

- From the p-values It can be seen that difference of log of close can be accepted with 90% confidence such that it can be concluded that the difference of log of close is stationary.

Simple Forecasting Technique:

Based on exploratory data analysis that has been done above, we have found that the closing price has almost no trend component over the years 2015 to 2019. We started with applying some simple forecasting techniques.

1.**<u>Average</u>**: As our data has equal variance across the year's consideration, a very simple method would be to take average of all the closing price to forecast future values.

$$Y_{(T+h|T)} = \frac{y_1+y_2+\cdots\ldots y_n}{T}$$

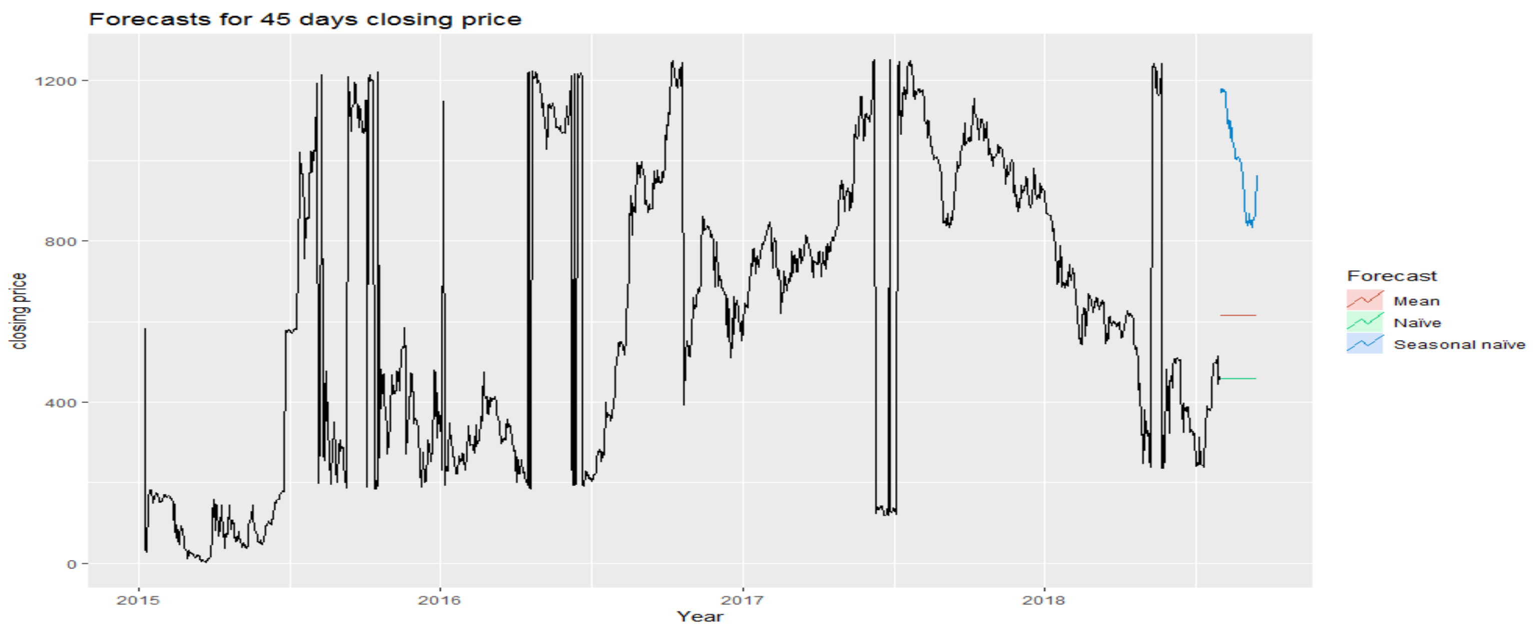
<u>2. Naïve:</u> This method takes the previous value and uses it for forecasting. This is simplest of all the methods but surprisingly the method works remarkably well for financial time series data because naïve method is optimal method when the data follow a random walk.

$$y_{(T+h|T)} = y_T$$

3.**<u>Snaive:</u>** This method is similar to Naïve method but it takes a value from the history of season that belong to the same period in time. This method is useful when there are seasonality in the data.

$$y_{(T+h|T)} = y_{T+h-m(K+1)}$$

Here m= the seasonal period, and k is the integer part of $(h-1)/m$ (the step of years in the forecast prior to time T+h

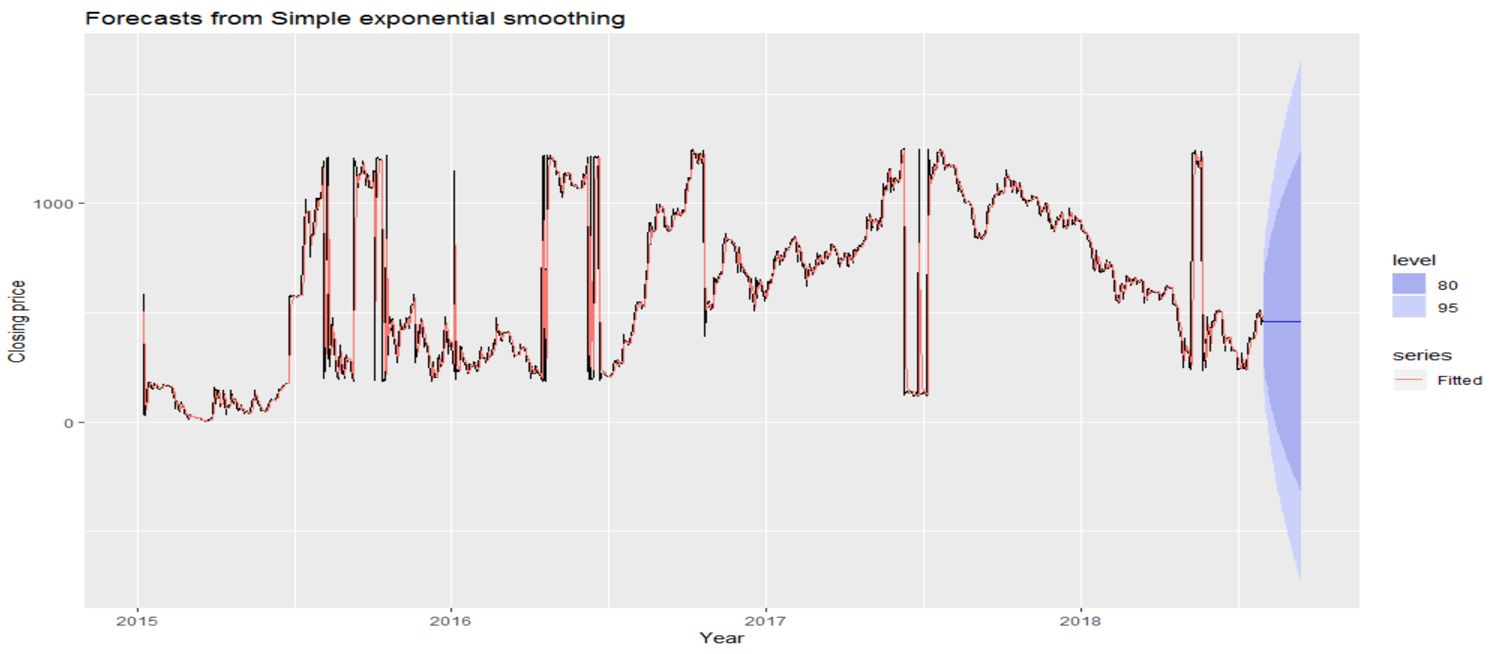


Plot for forecasting using Mean,naïve and Snaive methods

4. Exponential Smoothing method: This is a weighted moving average method of forecasting. This type of method is particularly useful when there is no clear trend or seasonal pattern in the data. As seen from the above exploratory analysis the closing price has no clear trend and seasonal component.

$$y_{T+1|T}=\alpha y_T+\alpha(1-\alpha)y_{T-1}+\alpha(1-\alpha)^2 y_{T-2}+\cdots$$

where $0 \leq \alpha \leq 1$ is the smoothing parameter. If the value of α is close to 1 then recent data have more weightage and vice versa. We have estimated the value of α from the data which comes out to be 0.61



The above plot shows the forecast for the 45 days. The dark blue line shows the estimated value and the shades shows the confidence interval at 80% and 90 % confidence interval.

-Models:

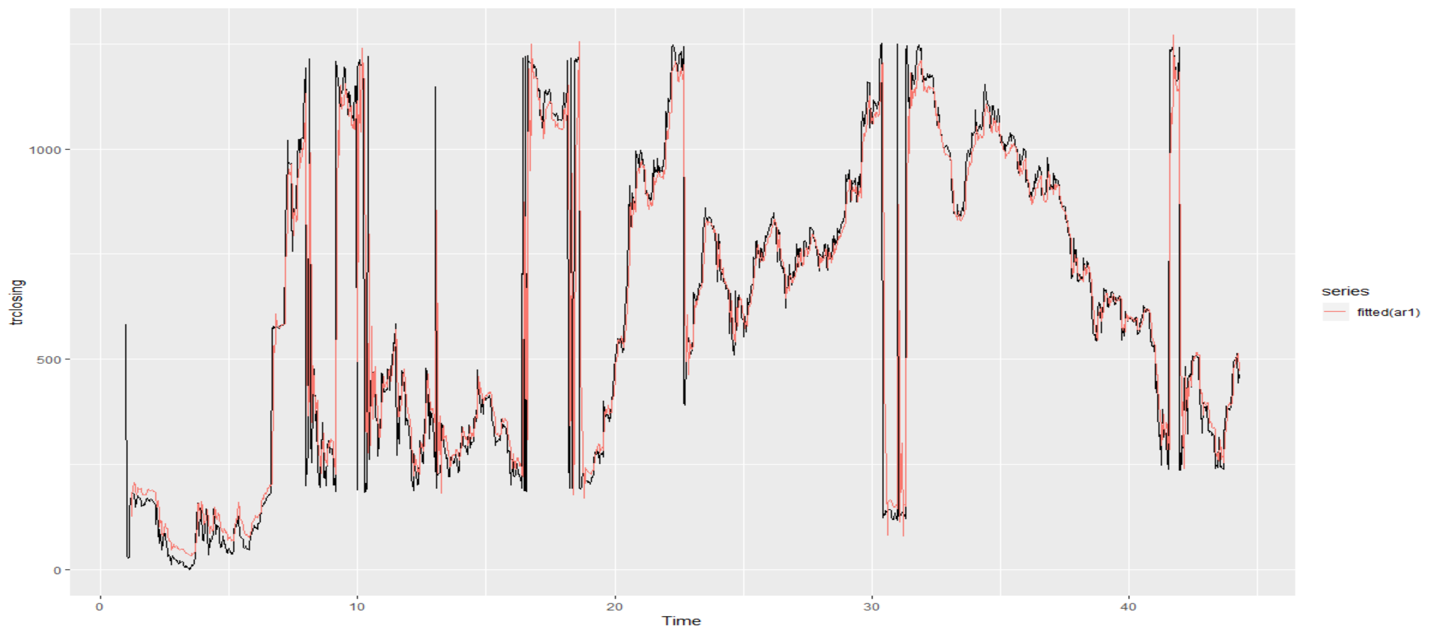
1. Autoregressive Model:

In this regression model, the response variable in the previous time period has become the predictors to predict current time observation and the errors have our usual assumptions about errors in a simple linear regression model. The order of an autoregression is the number of immediately preceding values in the series that are regressed to predict the present time value.

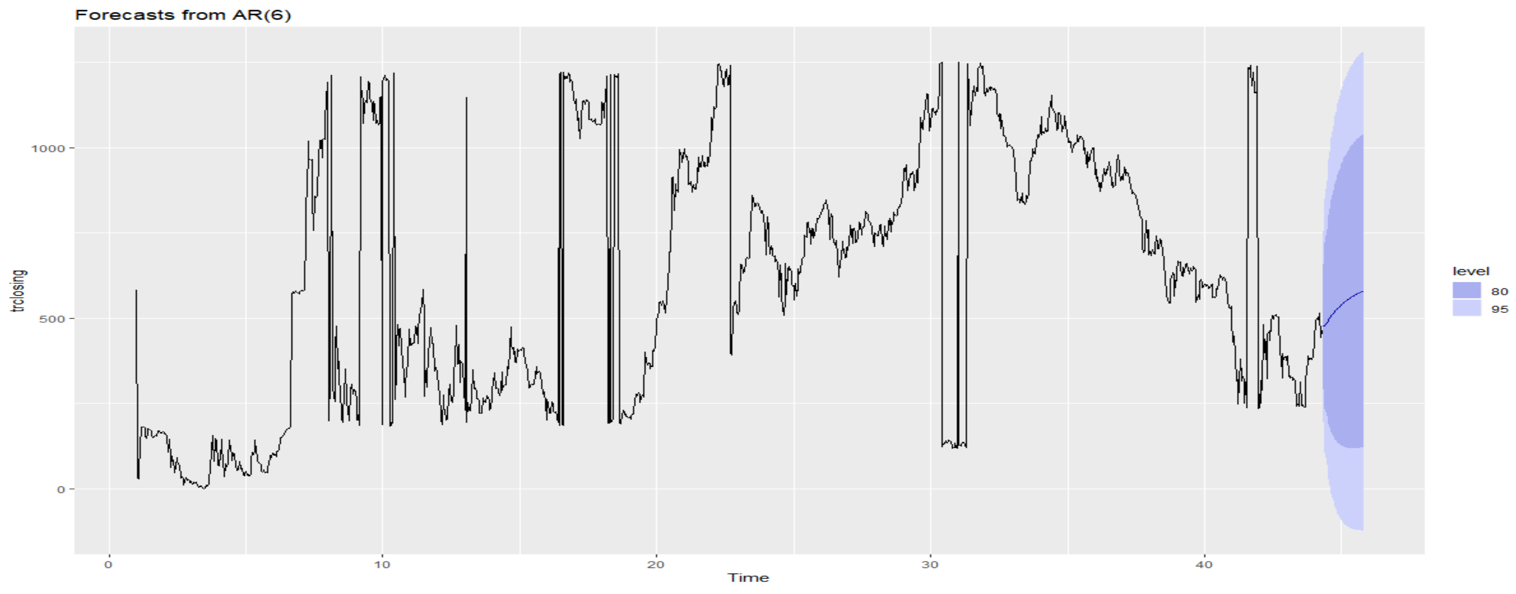
$$y_t = \beta_o + \beta_1 y_{t-1} + \epsilon_t$$

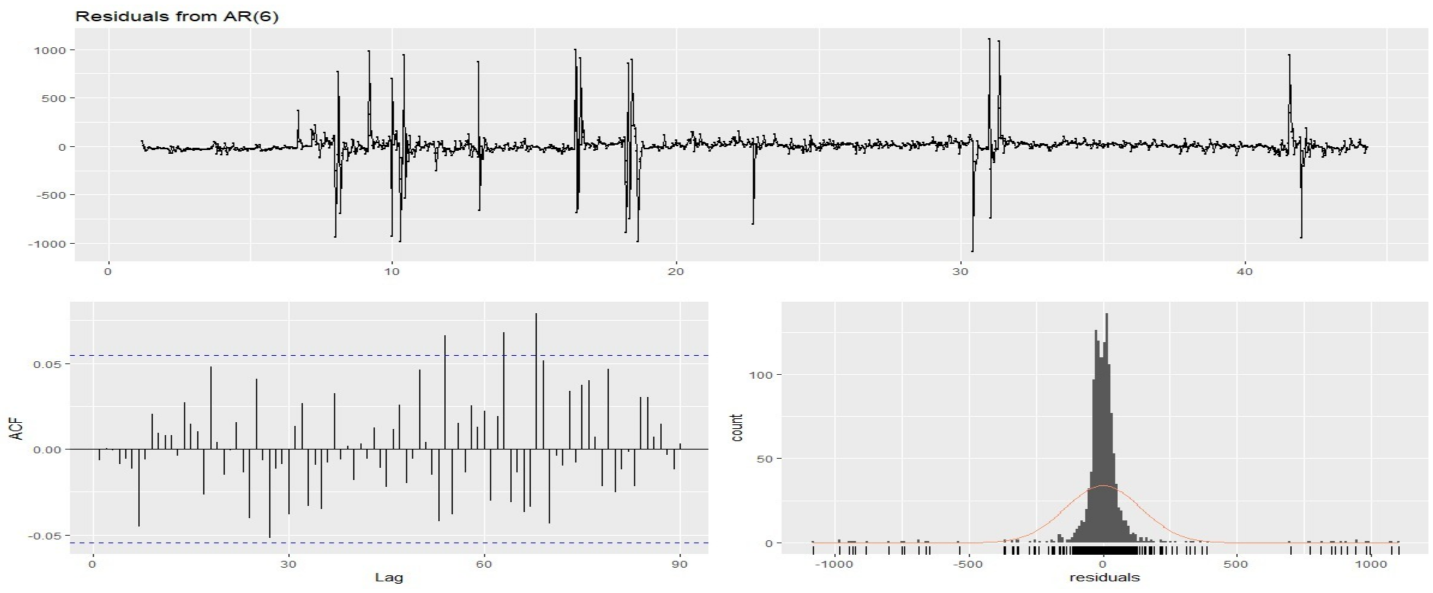
ar1=ar(trclosing,aic=TRUE)

Above command tries to look out for best AR model for given data, with minimum AIC value by varying the lags for different model. Here, minimum AIC lag value for our data is 6. So our model can be represented as AR(6).



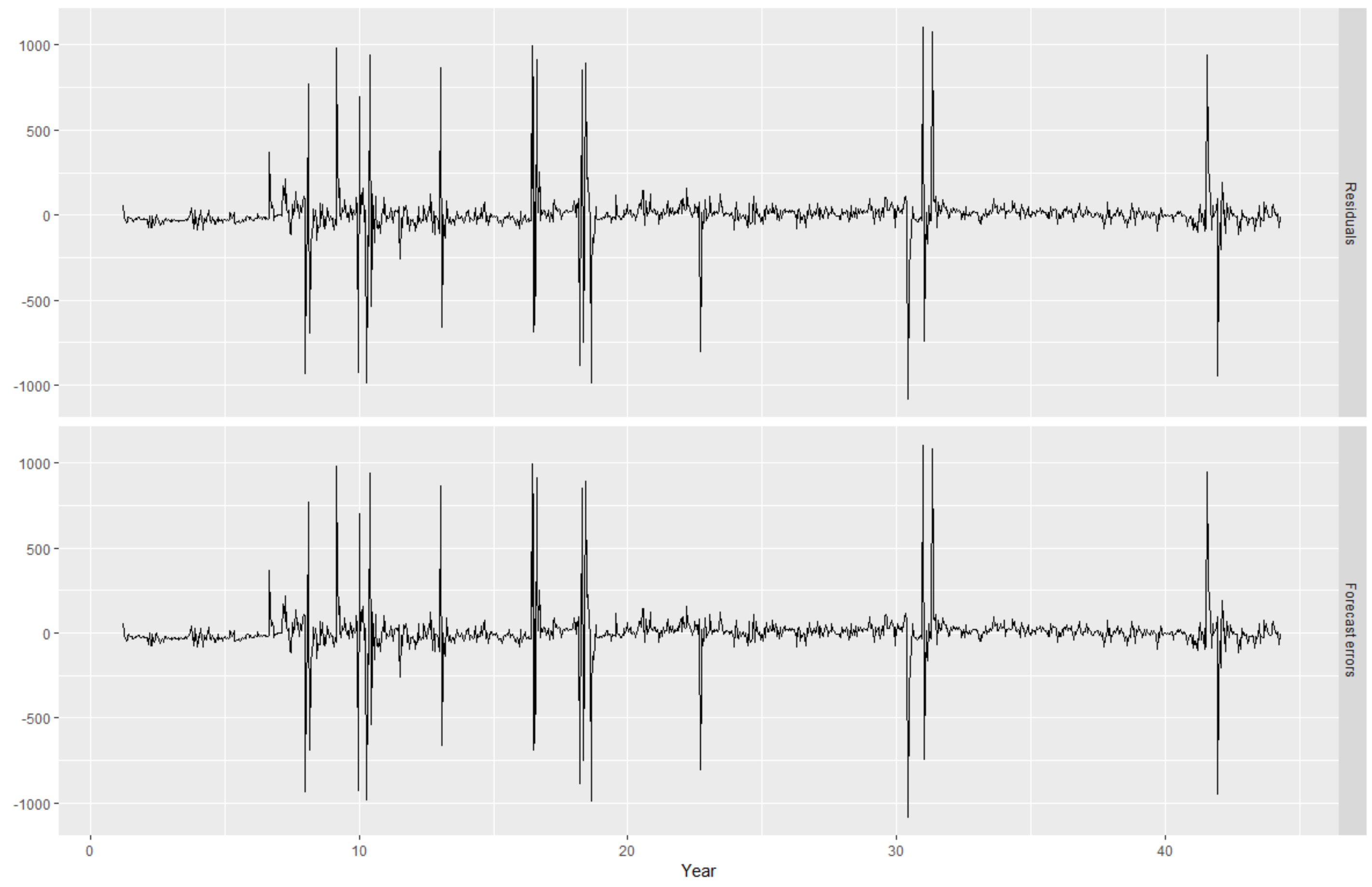
As we can see from the forecasted values for test data (45 predictions), the confidence interval is large, which shows AR model is not really best fit for this data as we can also understand intuitively as data has very little trend and seasonality.





LAG	AIC	BIC
1	16718.11	16733.62
2	16626.68	16669.36
3	16621.18	16647.03
4	16599.01	16630.03
5	16590.79	16626.98
6	16586.41	16630.77

Pseudo Out Of Forecast Error:

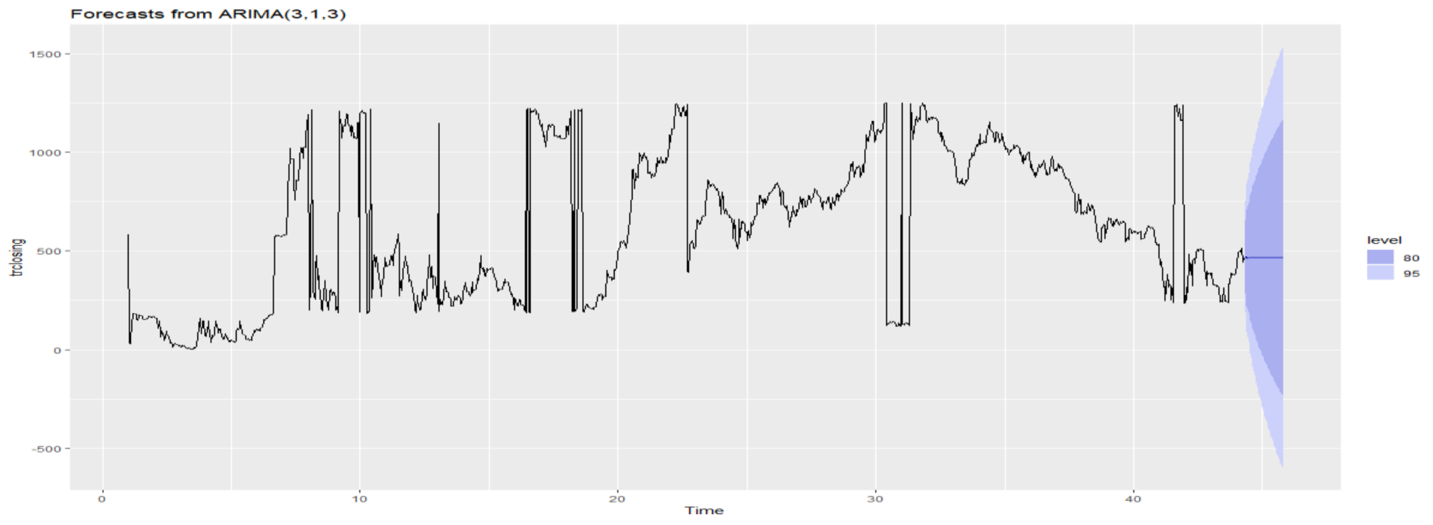


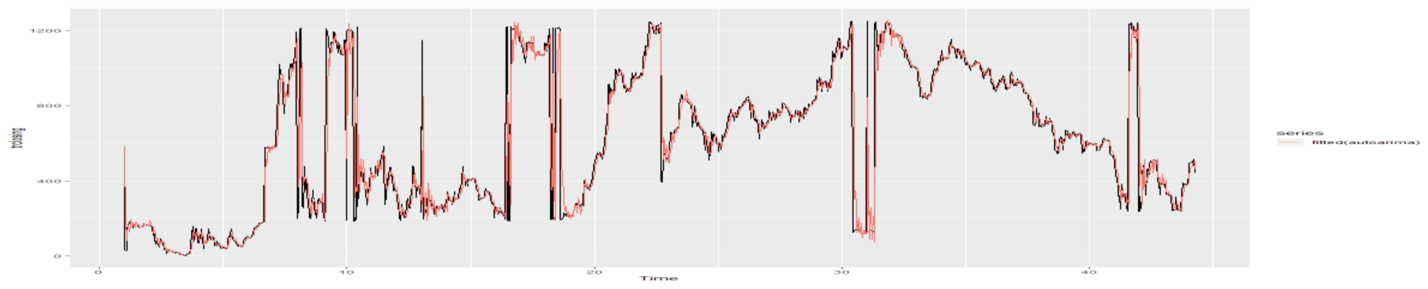
We applied AR model over Train dataset and predicted t+1 data point value and compared it with real value and plotted its residual plot,Seeing plot we can say that there is little or no residual error.

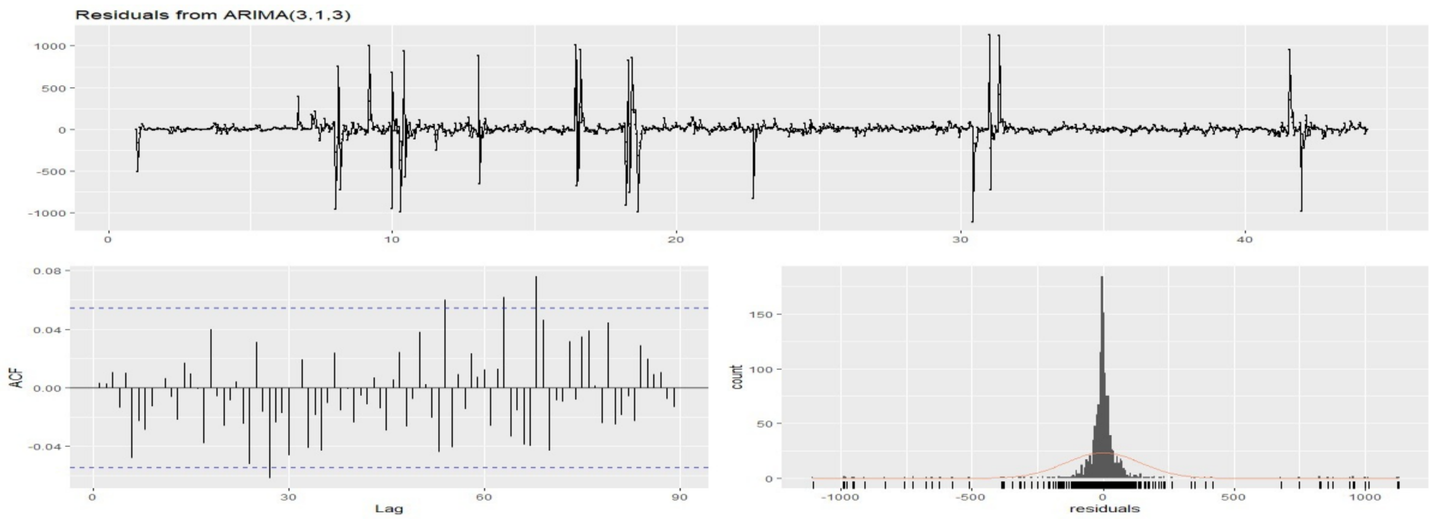
As we can see from the forecasted values for test data (45 predictions), the confidence interval is large, which shows AR model is not really best fit for this data as we can also understand intuitively as data has very little trend and seasonality.

ARIMA (3,1,3):

ARIMA is one of the most classic time series forecasting models. During the modelling process, we mainly interested in finding 3 parameters. Auto-regression (AR) term (p), namely the lags of previous value; Integral(I) term for non-stationary differencing and Moving Average (MA) for error term(q).







Here we applied auto arima function which tries to find best values for parameters p, d, q by minimizing AIC value of the model. Here the values for p, d, q are 3, 1, 3 respectively. Arima model works best with seasonal data, which is not the case here hence the results are more or less same as exponential smoothing.

Observations:

•Error Table of Different Models .

MODEL	ME	RMSFE	MAE
MEAN	3.3922	3806.81	3298
NAIVE	-24.44	1715	462
SNAIVE	1239	5087.55	3820
EXPONENTIAL SMOOTHING	12.1972	37.22948	29.5008
AUTO-REGRESSION	-64.5679488	68.28193	64.56795
ARIMA	10.13804997	39.0764	3.24061

CONCLUSION:

From principle of parsimony we found out that exponential smoothing predicts the Ethereum prices which are most close to the actual values. But market sentiments and government regulations dominate the prices to a very large extent and accurately predicting future Ethereum prices is very difficult.

R Codes:

```
# loading the data

data <- read.csv("C:/Users/DANISH/Desktop/smba_project2/Etherum.csv")

str(data)  # all the data are factors

#eliminating comma

library(gsubfn)

data$Volume = gsub(',','',data$Volume)

data$Market.Cap = gsub(',','',data$Market.Cap)


#converting into numeric

data$Open=as.numeric(data$Open)

data$High=as.numeric(data$High)

data$Low=as.numeric(data$Low)

data$Close=as.numeric(data$Close)

data$Volume=as.numeric(data$Volume)

data$Market.Cap=as.numeric(data$Market.Cap)

str(data)


# sorting and converting


data$Date=as.Date(data$Date, "%d-%b-%y")

data=data[order(data$Date),]

summary(data)


#splitting the data

test=data[1301:1345,]

data=data[1:1300,]


#converting dataset into time series objects

library(xts)

data_t = xts(data[, -1], order.by=as.POSIXct(data$Date))
```

```
class(data_t)

#Stationarity Check

#removing non-stationarity
plot(data_t$Close)
plot(log(data_t$Close))
plot(diff(log(data_t$Close)))

#augmented Dickey Fuller Test
library(tseries)
adf.test((data_t$Close), alternative="stationary",k= )
adf.test(na.omit(diff(data_t$Close)), alternative="stationary", k = 365)
adf.test(log(data_t$Close), alternative="stationary", k=365)
adf.test(na.omit(diff(log(data_t$Close))), alternative="stationary", k=365)
plot(diff(log(data_t$Close)))

#data visualization
library(forecast)
library(fpp2)
library(ggplot2)
autoplot(data_t$Close)
ts_data<- ts(data,start = c(2015, 8,7), frequency = 365)
ts_test<-ts(test)
autoplot(data_t[,1:4])
autoplot(data_t[,1:4], facets = TRUE)
#smoothing
autoplot(data_t[,1:4], facets = TRUE) +
geom_smooth()
autoplot(data_t[,1:4]) +
geom_smooth()
```

```
#lag plots and acf plots and white noise test

ggAcf(data_t)


#Ljung Test for serial autocorrelation

#null Hyp: there is no serial corelationupto lag 10

Box.test(data_t$Close,lag=10,type="Ljung-Box")  #we reject the null


#Model builing

#simple forecasting method

library(tseries)

autoplot(ts_data[,5]) +

autolayer(meanf(ts_data[,5], h=45),

          series="Mean", PI=FALSE) +

autolayer(naive(ts_data[,5], h=45),

          series="Naïve", PI=FALSE) +

autolayer(snaive(ts_data[,5], h=45),

          series="Seasonal naïve", PI=FALSE) +

ggtitle("Forecasts for 45 days closing price") +

xlab("Year") + ylab("closing price") +

  guides(colour=guide_legend(title="Forecast"))


#residual check

checkresiduals(naive(diff(log(ts_data[,5]))))

#reject the null hypo that autocorelation comes from white noise no aucor

#exponential moving average

fc <- ses(ts_data[,5], h=45)

round(accuracy(fc),2)

autoplot(fc)+

autolayer(fitted(fc), series="Fitted") +

ylab("Closing price") + xlab("Year")

fc$model # alpha value=0.6175 and lo=375.39

accuracy(ts(fc$mean),ts_test[,5])
```

```
# using pipe

#residual vs one step forecast error

cbind('Residuals' = residuals(fc),

      'Forecast errors' = residuals(fc,type='response')) %>%

autoplot(facet=TRUE) + xlab("Year") + ylab("")

checkresiduals(fc)


#AR model

library(vars)

require(strucchange)

lgcls<- lag(data_t$Close, start = c(2015,8,7))

lgcls[1]=500

data_qlr<-cbind(data_t,lgcls)

model <- lm(data_qlr$Close~data_qlr$Close.1)

ar1=ar(na.omit(diff(log(data_t$Close)),aic=TRUE))

myqlr<- qlr.test(model,from=c(2015,8), to=c(2018,3))

qlr

plot(myqlr,alpha=0.05)

ar1=ar(diff(log(trclosing,aic=TRUE)))

ar1

plot(forecast(ar1,h=45))

pred1=forecast(ar1,h=45)

accuracy(forecast(ar1))


#test accuracy

accuracy(pred1,test$Close)

forecast(ar1,h=45)%>%autoplot()

cbind('Residuals' = residuals(ar1),

      'Forecast errors' = residuals(ar1,type='response')) %>%

autoplot(facet=TRUE) + xlab("Year") + ylab("")

a1=arima(trclosing,order=c(1,0,0))
```

```
a2=arima(trclosing,order=c(2,0,0))

a3=arima(trclosing,order=c(3,0,0))

a4=arima(trclosing,order=c(4,0,0))

a5=arima(trclosing,order=c(5,0,0))

a6=arima(trclosing,order=c(6,0,0))

autoarima= auto.arima(trclosing, seasonal = FALSE)

cbind('Residuals' = residuals(autoarima),

     'Forecast errors' = residuals(autoarima,type='response')) %>%

autoplot(facet=TRUE) + xlab("Year") + ylab("")

accuracy(forecast(autoarima, h=45),test$Close)

forecast(autoarima,h=45)%>%autoplot()

library(lmtest)

library("dynlm", lib.loc="~/R/win-library/3.5")

grangertest(Close ~ High, order = 2, data = train)

adl1=dynlm(trclosing ~ L(train$High,1) + L(trclosing,1) + L(trclosing,2) + L(trclosing,3)+ L(trclosing,4)+ L(trclosing,5) +
L(trclosing,6))

coeftest(adl1)

accuracy(adl1)

accuracy(forecast(adl1, h=45),test$Close)

cbind('Residuals' = residuals(adl1),

     'Forecast errors' = residuals(adl1,type='response')) %>%

autoplot(facet=TRUE) + xlab("Year") + ylab("")

accuracy(forecast(adl1, h=45),test$Close)

accuracy(adl1)
```