## Introduction

Bitcoin (₿) is a cryptocurrency, a form of electronic cash. Bitcoin was invented by an unknown person or group of people using the name Satoshi Nakamoto and released as open-source software in 2009. Bitcoins are created as a reward for a process known as mining. They can be exchanged for other currencies, products, and services (Wikipedia). Bitcoin offers the promise of lower transaction fees than traditional online payment mechanisms and is operated by a decentralized authority, unlike government-issued currencies.

A time series is a series of data points indexed (or listed or graphed) in time order. Time series are very frequently plotted via line charts. Time series analysis comprises methods for analyzing time series data in order to extract meaningful statistics and other characteristics of the data. Time series forecasting is the use of a model to predict future values based on previously observed values. (Wikipedia).

## Bitcoin Time-Series Forecasting

Bitcoin is one of the revolutionary cryptocurrencies in the market today. Even though it was silent all along in the marketing world, it caught the eye of the entire world in the past few years by defying every forecasting algorithm ever made for it. Today, as of Feb 14, 2019, it stands at a market value of $3619.98, which is quite less as compared to prices in the year end of 2017 but still most of the online merchants have recognized it as a form of currency. Since then it has become kind of a stock market where people invest to generate profits.

Forecasting the value of bitcoin as forecasting the flow of stock market. Over the years many algorithms have been developed for forecasting stock markets, but very few have focused on bitcoin price prediction. In this project, we as a group have focused on predicting the price of Bitcoin prices for 10 days staring from Jan 17-Jan 26, 2019 based on historical price data by studying trends in the form of both seasonal and general trends. The forecasted prices would help you understand the risk factor involved in investing in the bitcoins and the possibility of having a profit or loss in it. With that said, it is to be recalled that forecasts models are most probably wrong and hence should always include a prediction error with it.

## Analysis

Time series forecasting is quite different from other models because it is time dependent, it has increasing or decreasing trend. Most of the time series have some form of seasonality trends, i.e. variations specific to a particular time frame. Based on the time series project, we had below mentioned research question to answer:

1) How to collect the data?

2) Sources of making the dataset available?

3) How to clean the data?

4) Which model is to be used for prediction?

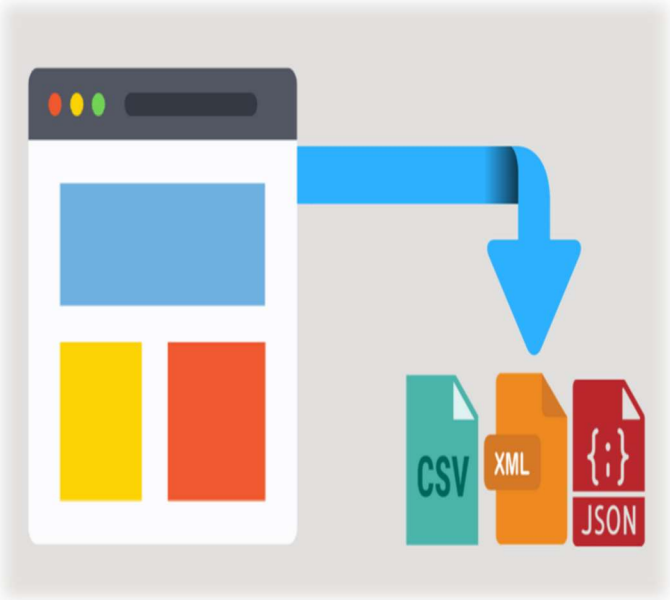
5) What would be the future scope of the prediction model?

### Data Collection

We have collected the historical data from the website coinmarketcap for our analysis. We have used two methods for collecting the data:

1) Web Scarping: We have scrapped the data for bitcoin and Ethereum and have converted the data from html format to csv format.

2) Library of coinmarketcap: Coinmarketcap library in R is used to get prices of cryptocurrencies from 'Coin Market Cap'.




There are many websites which provides API to import the cryptocurrency data. Coinmarketcap provides an API to get the both live and historical data. Coinmarketcap library in R is used to get prices of cryptocurrencies from 'Coin Market Cap'.

There are total seven variables in the data i.e. Data, Open, High, Low, Close, Volume and market cap. The Open, High, Low and Close columns in the data set signify the Opening, highest, lowest and closing price of Bitcoin and Ethereum against the USD on that particular day, whereas the volume refers to the number of transactions undertaken in the market on a particular day for both bitcoin and Ethereum and the market cap refers to the total amount of bitcoin and Ethereum in circulation.

## Data Cleaning

Data cleansing or data cleaning is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data. Below mentioned is cleaning of data conducted on the dataset:

1) We have formatted date in standard format from character

2) We have formatted the market cap as numeric from character by replacing ',' from the data.

3) We have formatted volume as numeric from character by replacing ',' from the data.

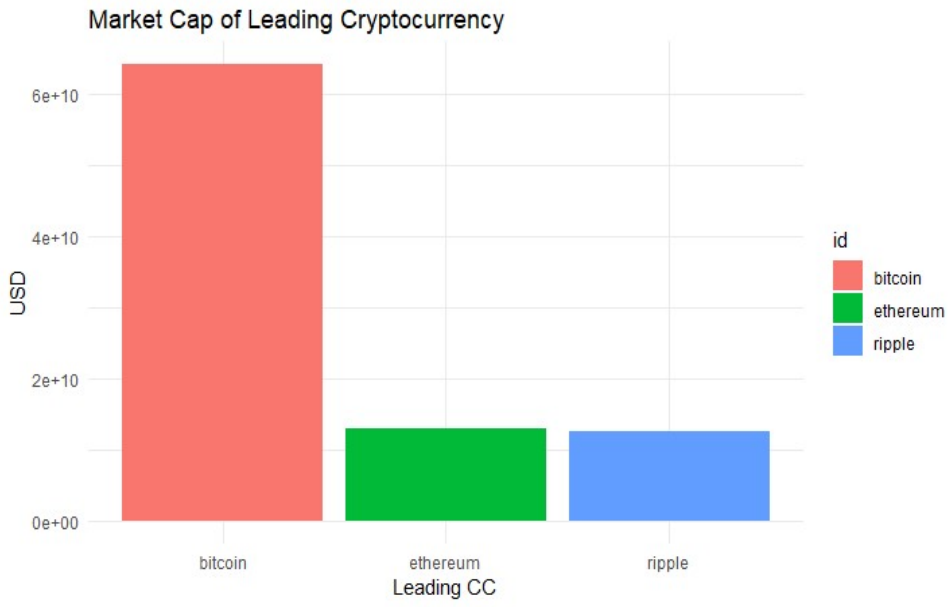
4) We have calculated total number of "NA" values in each column.

5) We have used the difference between the highest and the lowest price of the day to fill the missing values of the volume using the rest of the data set.

We have written the data frame into csv file. Further, no data cleaning was required for Ethereum data as it doesn't have any "NA" or missing values.

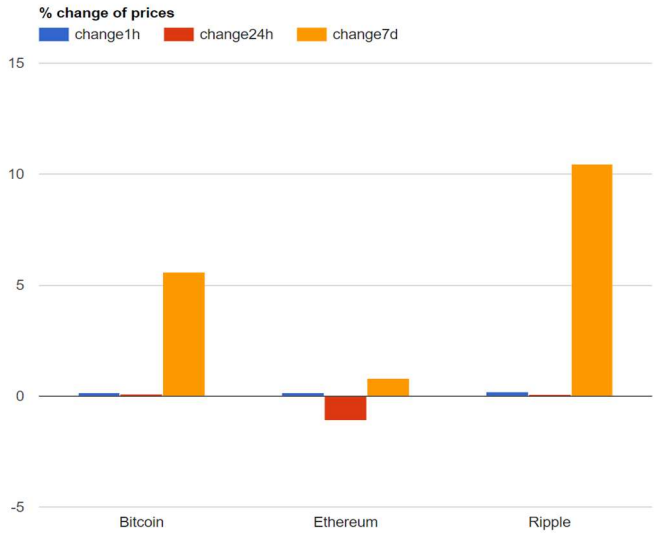
## Comparing Top 3 Cryptocurrencies

1) Market Cap of 3 leading cryptocurrencies



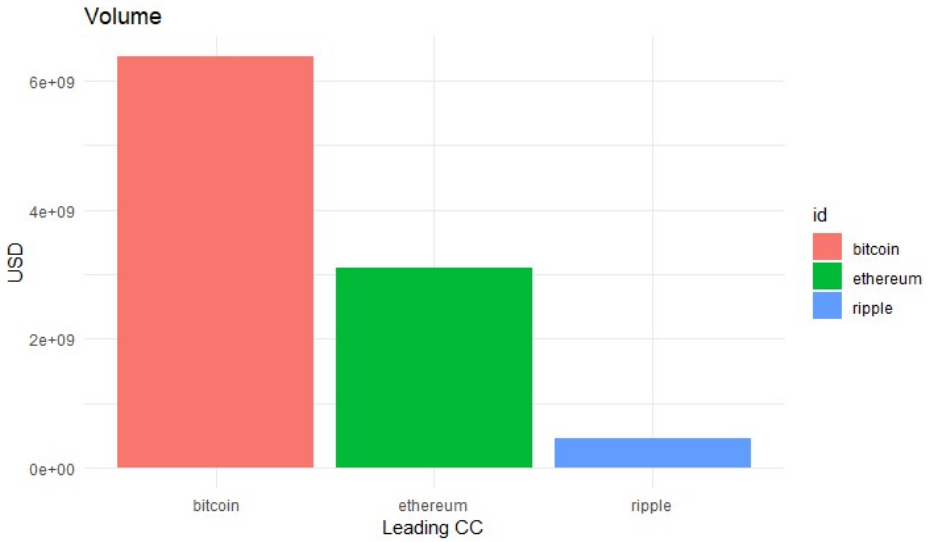

From the plot, it can be noticed that Bitcoin has the a very high market in comparison to Ethereum and ripple.

2) Variation in prices: Here, we have used googleVis library for visualization purpose as it can produce interactive charts.




Above plot is based on the changes of Prices in 1 hour, 24 hours and 7 days. Negative value of percentage variation shows a drop in the prices and positive means increase in the prices.

3) Volume

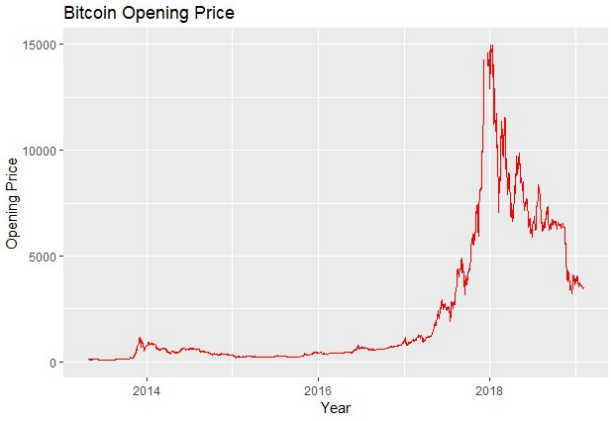



Volume is the number of transactions of a cryptocurrency. It can be noticed from the plot that number of transactions of Bitcoin is very high compared to other currencies.

**Plotting Bitcoin Data**

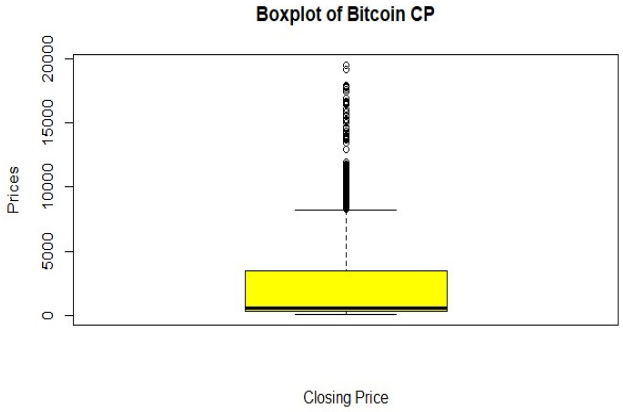
We have tried to plot bitcoin data based on variables in order to compare the relation between the variables and found out the all the variables are correlated to each other. Below is the plot drawn based on bitcoin data:

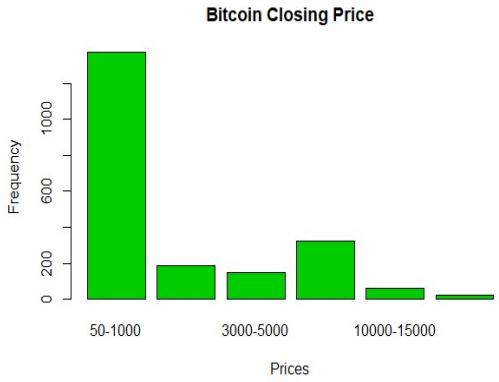
1) Bitcoin Opening Price & Closing Price



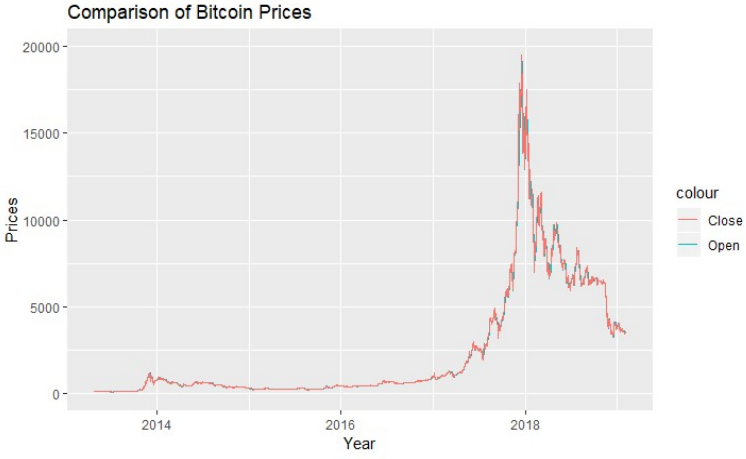


2) Boxplot and Histogram of Closing Price

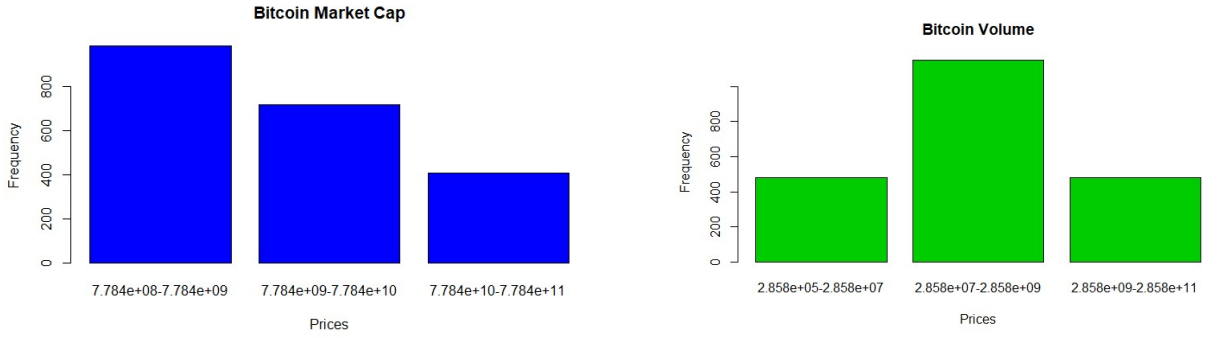


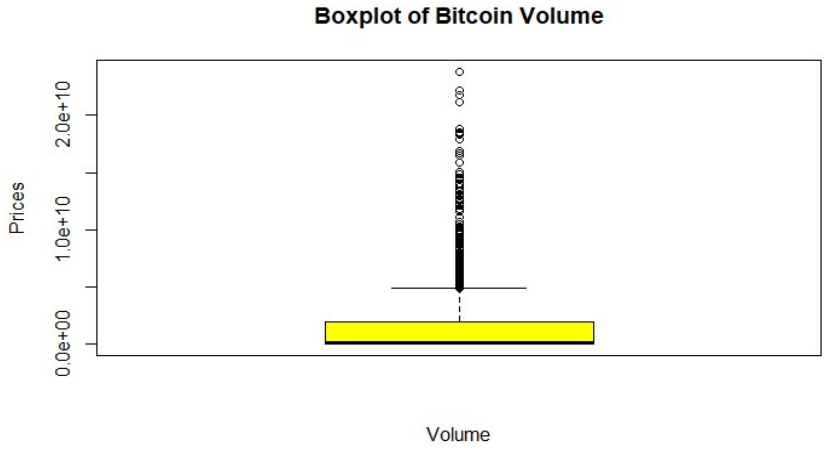




3) Comparing Bitcoin Opening Price & Closing Price: We can see from the first plot that bitcoin has seen a high price increase in the year of 2017 and major fall in the first quarter of 2018 due to various factors. We have also plotted comparison between opening and closing prices as there is not much of difference in the opening and closing price as it starts and closes near to same range.

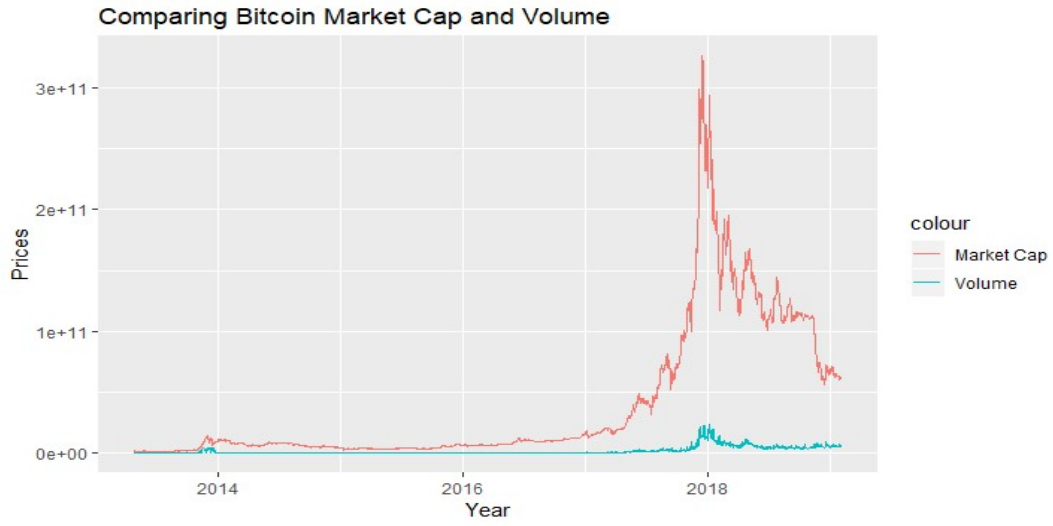


4) Boxplot and Histogram of Market Cap and Volume



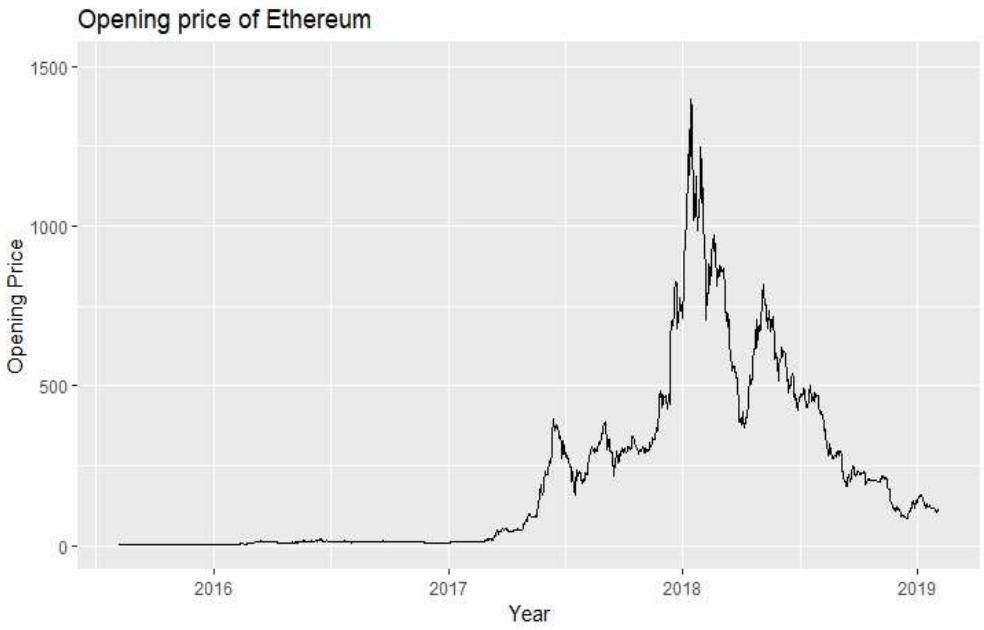

5) Comparing Bitcoin Market Cap and Volume



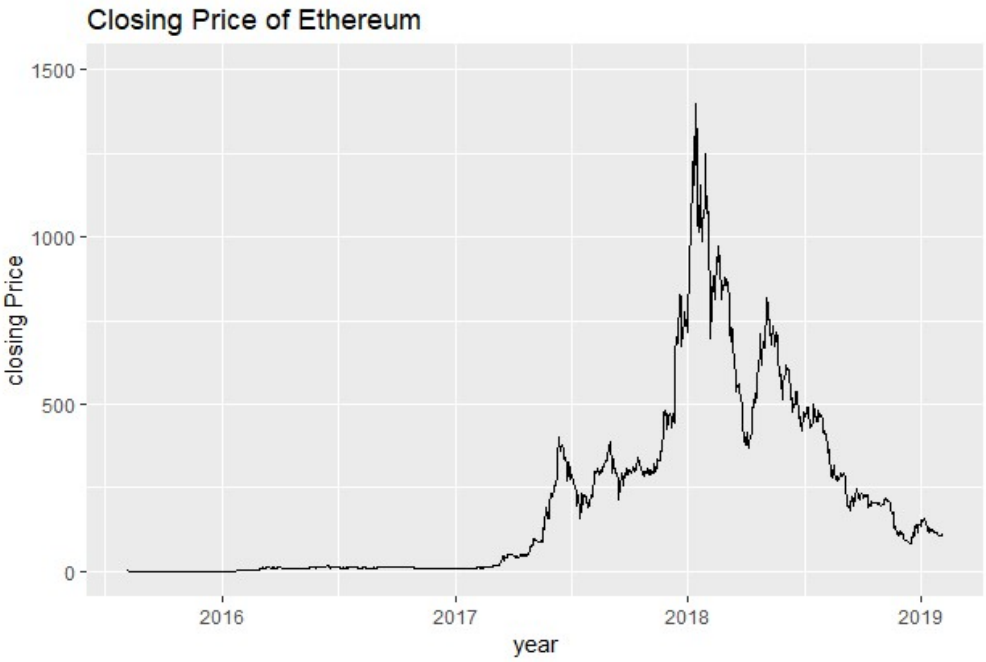

Bitcoin has 45% market cap in comparison to other currencies. Here, we tried to compare the market cap and volume which showed a significant difference.

**Plotting Ethereum Data**

1) Ethereum Opening Price



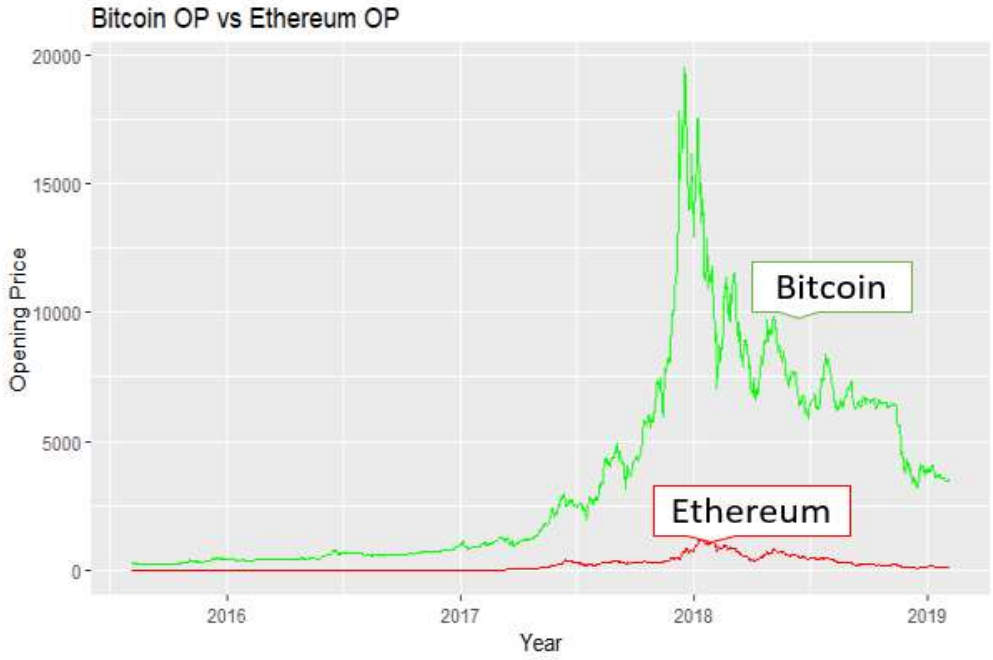

2) Ethereum Closing Price

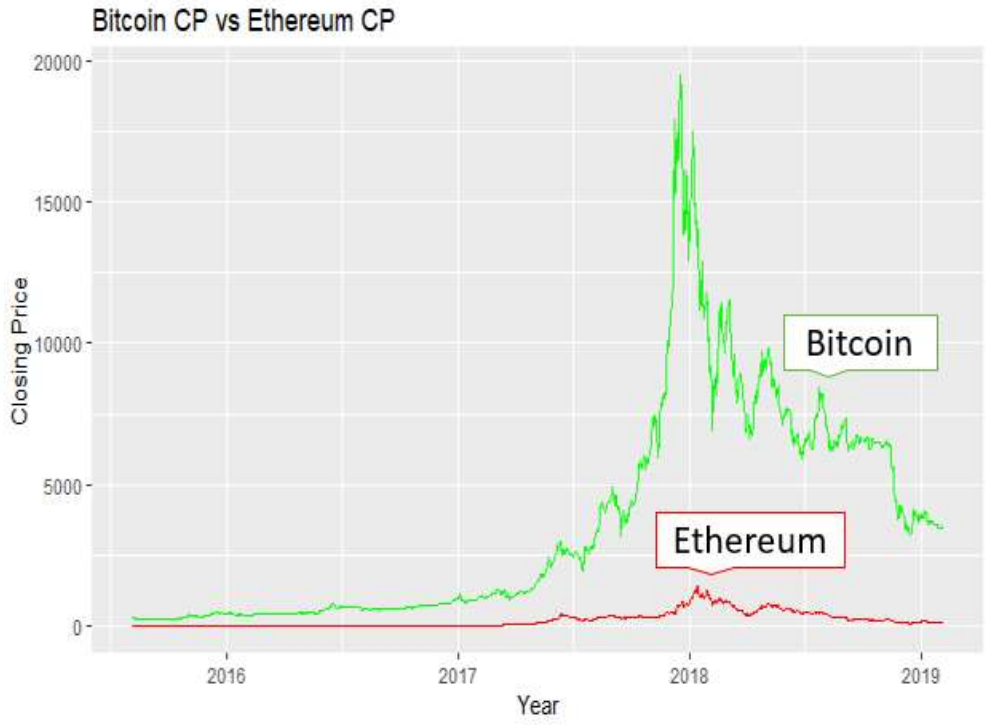



Below two plots are the opening and closing price of Ethereum which doesn't show us much of a difference.

**Bitcoin vs Ethereum**

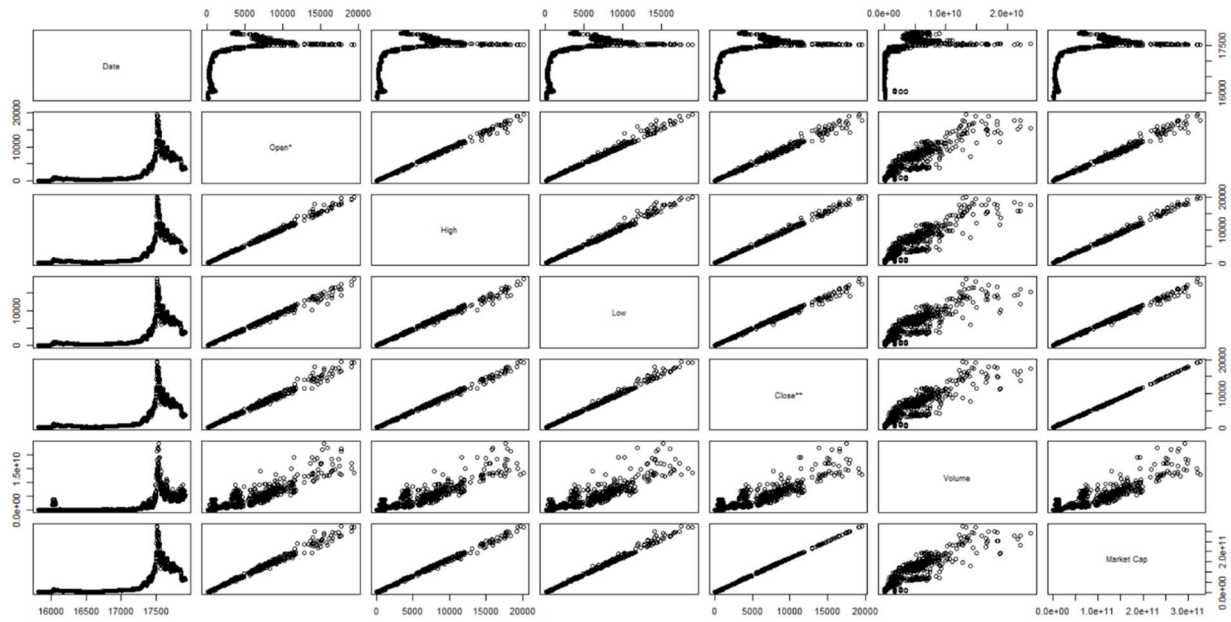
We have tried to compare the opening price of Bitcoin and Ethereum, and we have found that bitcoin had seen a boom and dip in their prices in the year 2017-2018 whereas Ethereum prices have been moving steadily once it got founded and same applies for closing price as there is not much of a difference for both the currencies.



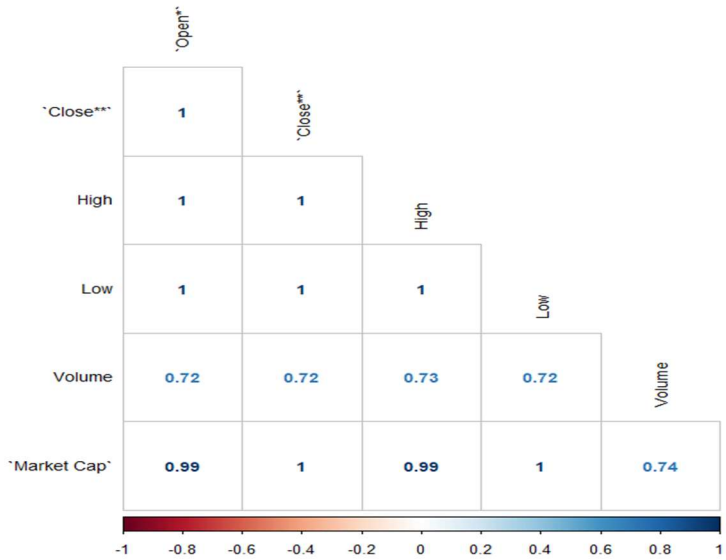
## Correlation between variables

Below is the correlation graph, where we can see that all the variables linearly correlated to each other.



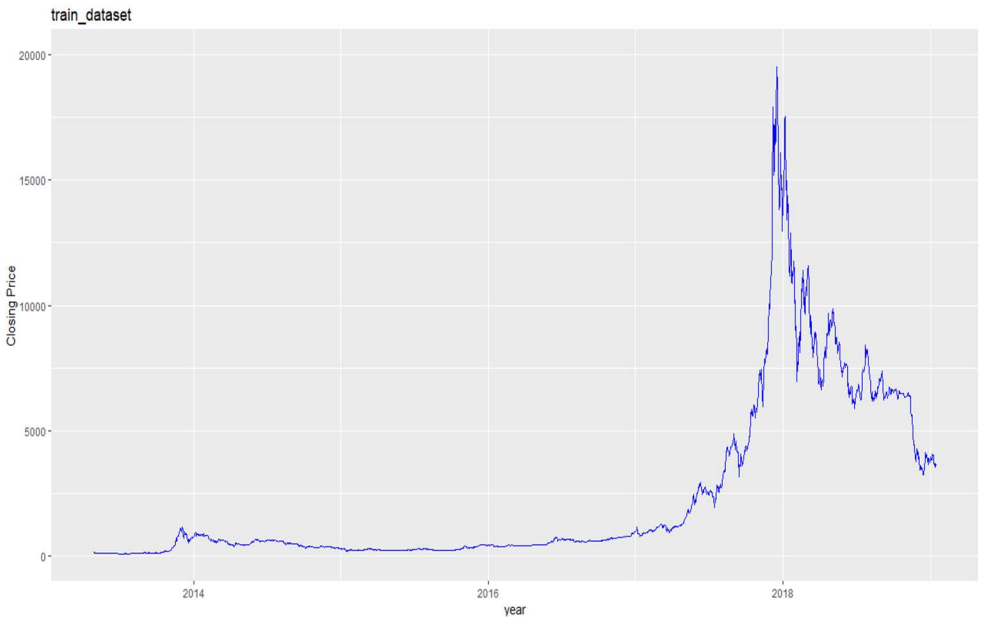

## Correlation Matrix

When working with data it is helpful to build a correlation matrix to describe data and the associations between variables. As we all know that the variables are correlated to each other, please refer below correlation matrix:

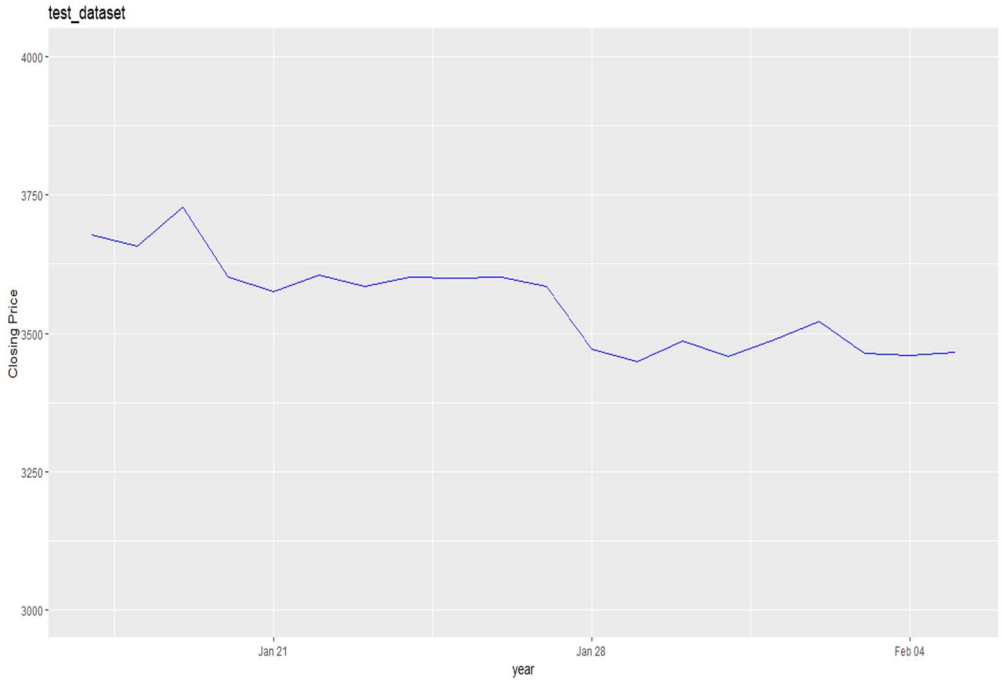


We have split the data into train and test to conduct our analysis and apply the model based on the training and testing data. Below is plot of both train and test dataset.

**Plot of Train dataset**

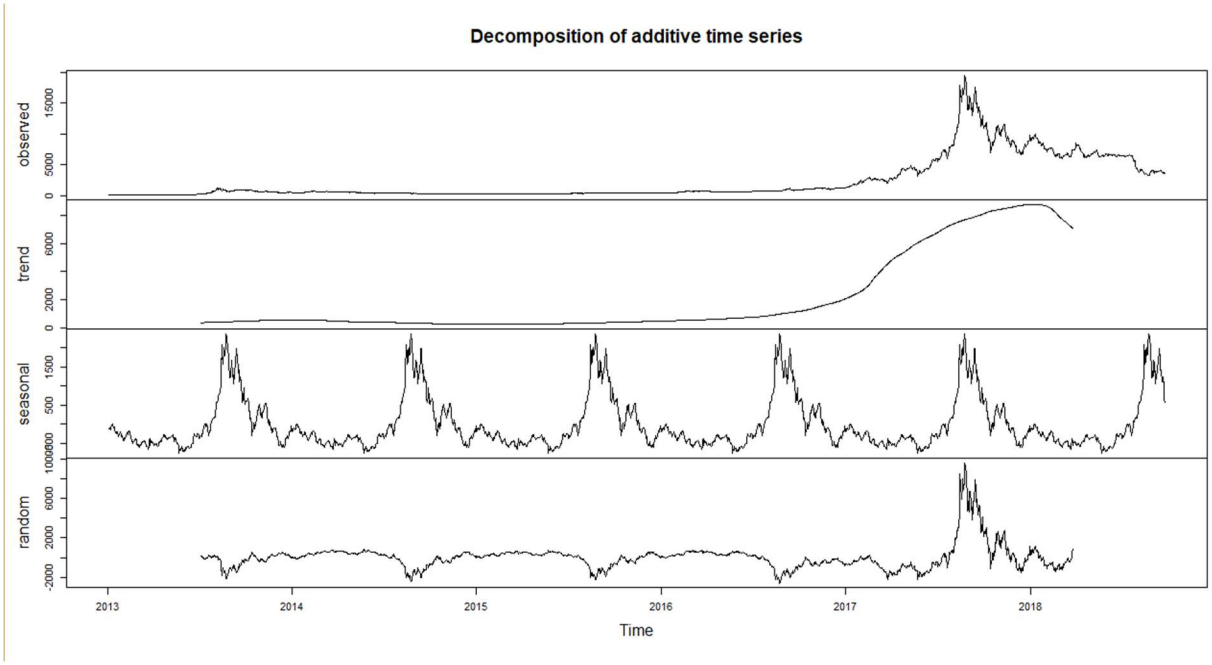



**Plot of Test dataset**



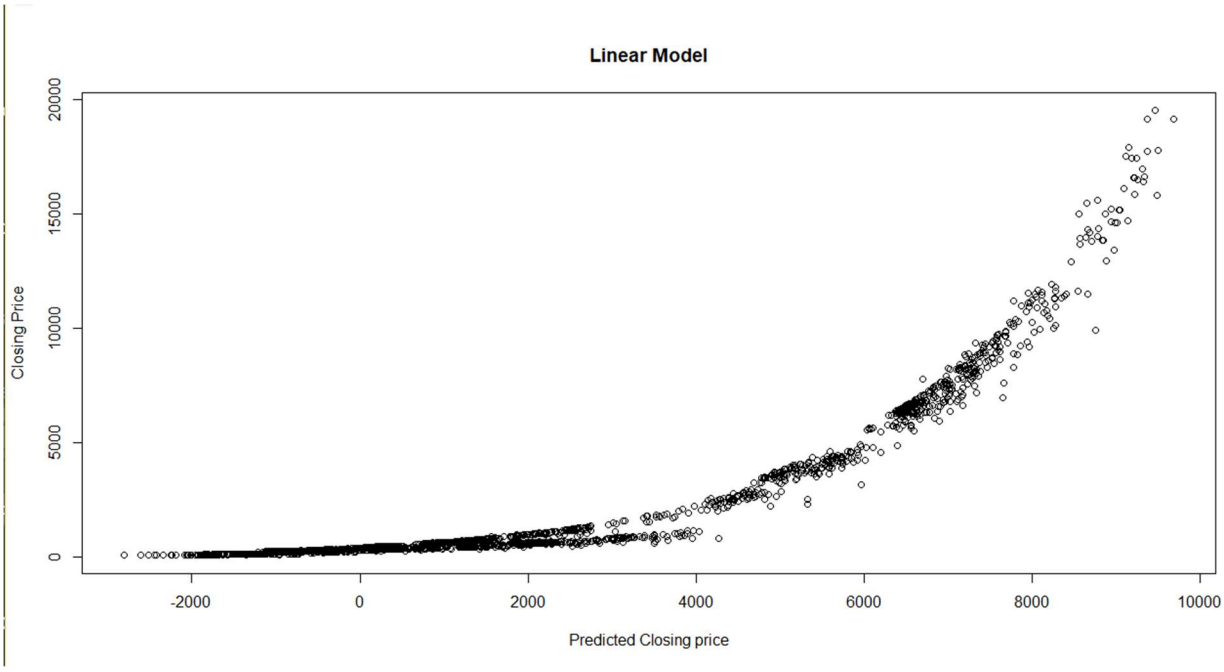
**Forecasting the model with closing prices of Bitcoin**

Time series decomposition is useful as it splits a time series into several components, each representing one of the underlying pattern categories. Often this is done to help understand the time series better, but it can also be used to improve forecasts.




Based on the above decomposition, it can be seen there is a uniform seasonal variation in the closing price of bitcoins over years and trend has been almost constant till the end of 2018.

We have also tried to plot a linear model to our data. Below is Linear model plot:



**Models selection**: Based on the closing price of bitcoin, we have analyzed the dataset with the help of forecasting models. Further, we have decomposed the dataset for understanding the variation in data over the period of time and seasonal trend in the data. We need to study the trends and seasonality involved in the closing price of Bitcoin. Based on the seasonal patterns, we can decide on which forecasting model to use. We found that data has a seasonal trend as well as multiplicative trend and its varying over the period of time, so based on the above factors, here are the below explained three forecasting algorithms used in predicting the price of bitcoins. Below are the three models explained in detail:

**1) Holt's Forecasting model**

Holt's model uses two parameters, one for the overall smoothing and the other for the trend smoothing equation. The method is also called double exponential smoothing or trend-enhanced exponential smoothing. Holt's model has three separate equations that work together to generate a final forecast. Exponential smoothing is a technique for smoothing time series data, exponential functions are used to assign exponentially decreasing weights overtime. Single exponential smoothing does not perform well when there is a trend in data. In such situation, we use 'double exponential smoothing or second-order exponential smoothing', which is a recursive application of an exponential smoothing twice. Basic idea is to consider the possibility of a series exhibiting some form of trend.

**2) ETS Forecasting Model**

Exponential triple smoothing is an algorithm which estimates the initial states and smoothing parameters of a forecast by optimizing the likelihood function to find the local minimum and is restricted within a parametric space to make sure data is forecastable. Exponential Smoothing is a technique to make forecasts by using a weighted mean of past values, wherein more recent values are given higher weights. Exponential smoothing is one of many window functions commonly applied to smooth data in signal processing, acting as low-pass filters to remove high frequency noise. (Wikipedia)

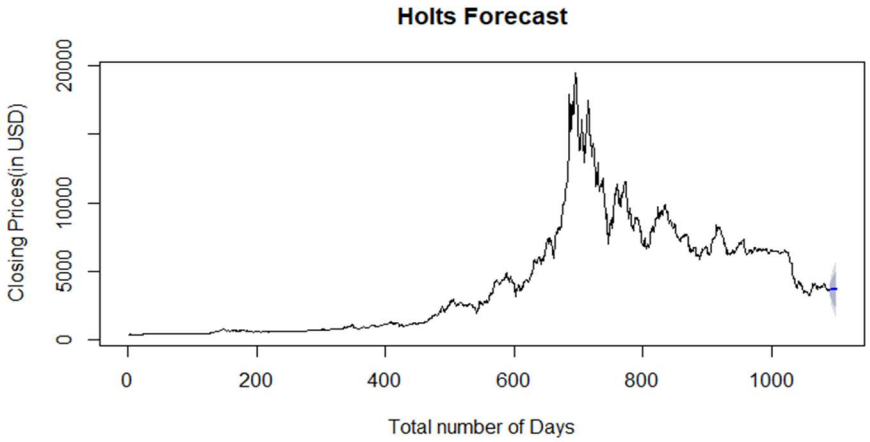
**3) ARIMA Forecasting Model**

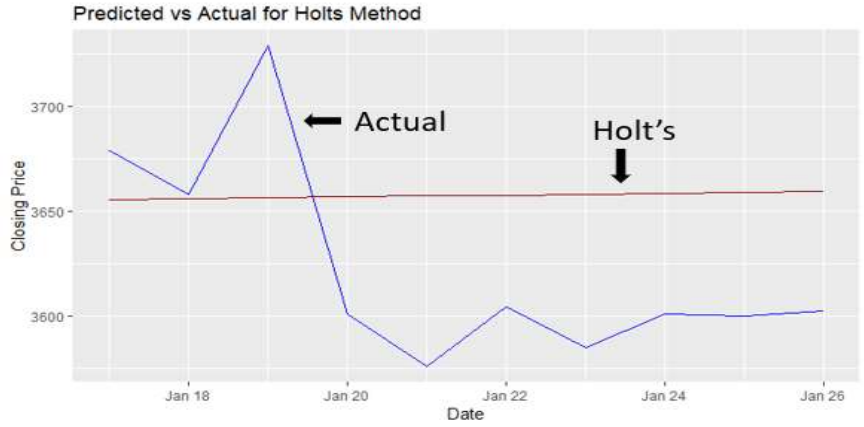
ARIMA models are, in theory, the most general class of models for forecasting a time series which can be made to be “stationary” by differencing (if necessary), perhaps in conjunction with nonlinear transformations such as logging or deflating (if necessary). A random variable that is a time series is stationary if its statistical properties are all constant over time. A stationary series has no trend, its variations around its mean have a constant amplitude, and it wiggles in a consistent fashion, i.e., its short-term random time patterns always look the same in a statistical sense. The latter condition means that its autocorrelations (correlations with its own prior deviations from the mean) remain constant over time, or equivalently, that its power spectrum remains constant over time. A random variable of this form can be viewed (as usual) as a combination of signal and noise, and the signal (if one is apparent) could be a pattern of fast or slow mean reversion, or sinusoidal oscillation, or rapid alternation in sign, and it could also have a seasonal component. An ARIMA model can be viewed as a “filter” that tries to separate the signal from the noise, and the signal is then extrapolated into the future to obtain forecasts.

## Results & Findings

### 1) Result and findings using Holt's Forecasting model:

Initially, we have constructed the model using Holt's algorithm and this model is used to predict the closing prices of bitcoins for next 10 days.



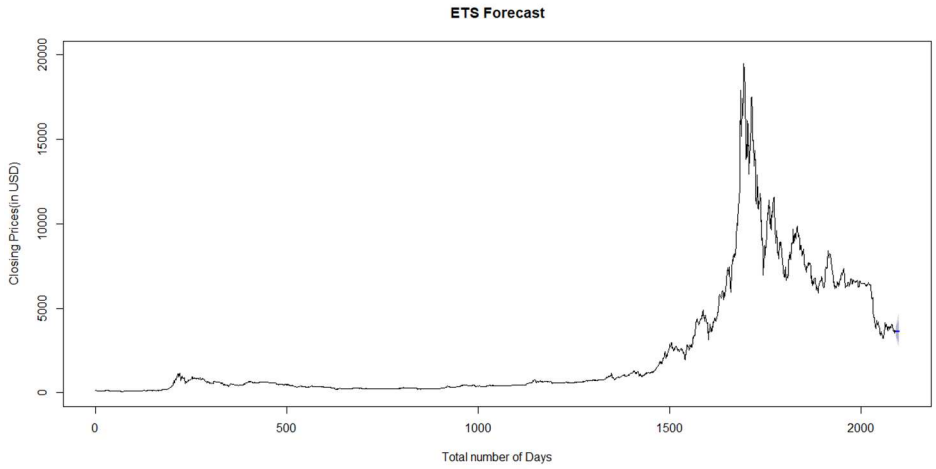

In the graphs, we can see the results, blue line is the actual line and red line the predicted line which we got after implementing the holt's method. As we know, Holt's model uses two parameters, one for the overall smoothing and the other for the trend smoothing equation. We can see red line is the straight line and it helped us in the smoothing of jagged line in data but it's not showing us the ups and down in the trend.

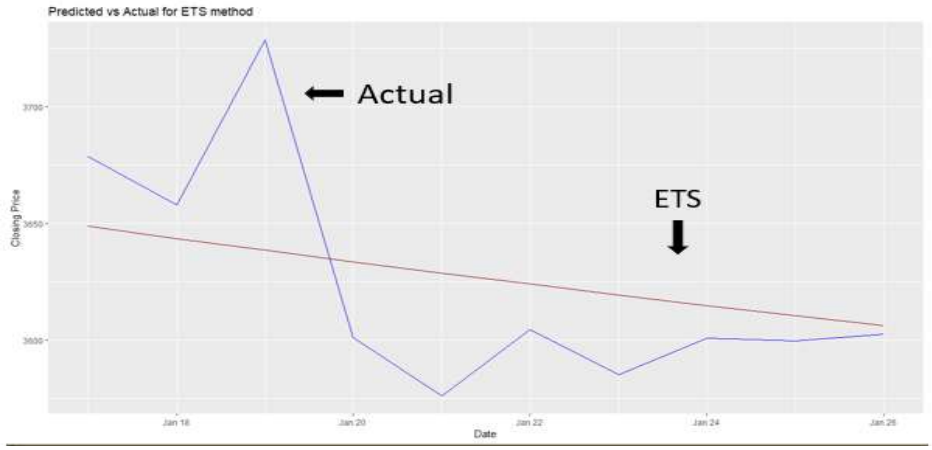
```
> accuracy(holt_df[,1], testdata) #accuracy
                ME     RMSE     MAE         MPE     MAPE
Test set -33.83528 57.9272 53.321 -0.9502374 1.474737
```

As we checked the accuracy of model, we got MAPE i.e. error rate of 1.47% which is good as its below 10%. But, we analyzed that we can improve the accuracy and also, we can get the line which is moving with the trend and also showing the fluctuation over the period of time. Hence, we decided to use other forecasting model i.e. ETS.

**2) Result and findings using ETS Forecasting model:**

Exponential Triple Smoothing, Triple exponential smoothing applies exponential smoothing three times, whenever our data shows both trends and seasonality (also called periodicity). There are two types of seasonality 'multiplicative' and 'additive' in nature. Our data shows multiplicative nature as, the seasonal component is expressed in relative terms (percentages), and the series is seasonally adjusted by dividing through by the Seasonal component.




In the graphs, we can see the results, blue line is the actual line and red line the predicted line which we got after implementing the ETS method. We can see the red line is not curving with the data, it is still possessing the linear trend.

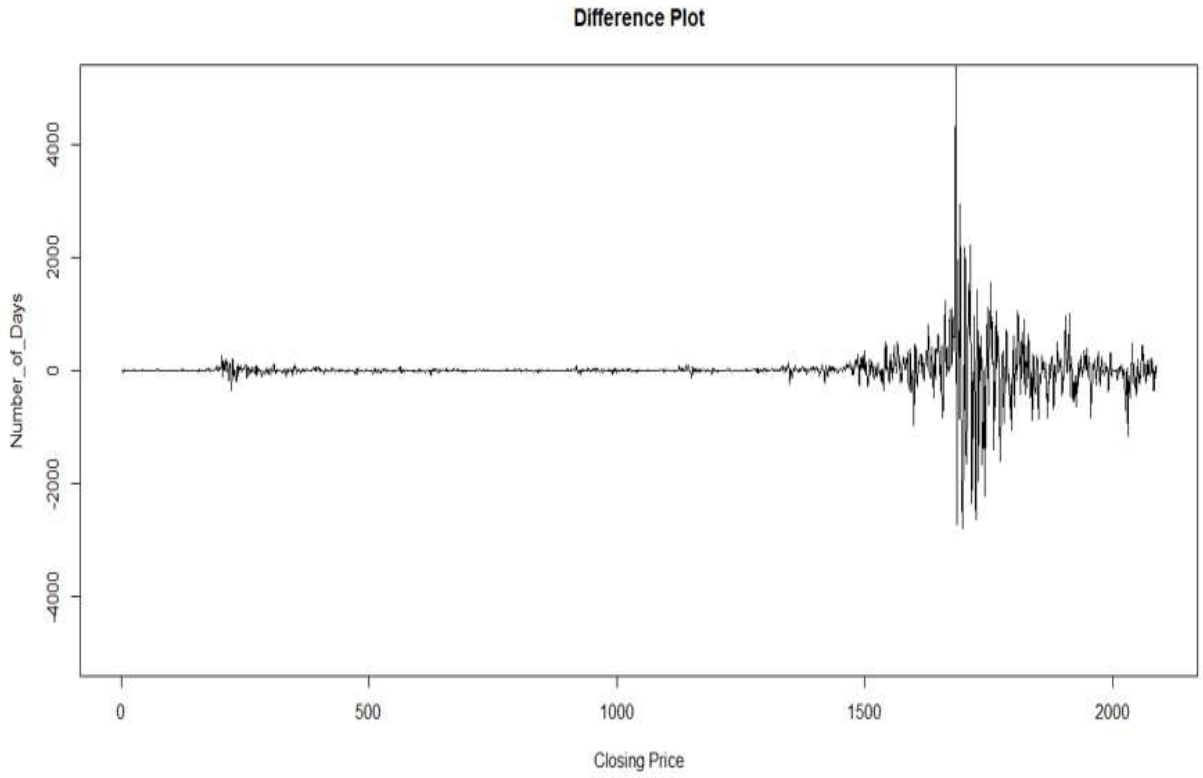
```
> accuracy(ETS_df[,1], testdata)
                    ME     RMSE      MAE         MPE      MAPE
Test set -3.235913 38.62286 30.12205 -0.1021931 0.8269502
```

As we checked the accuracy of model, we got MAPE i.e. error rate of 0.83% which is good as its below 10%, and also better then Holts method but we analyzed that we can improve the accuracy and also, we can get the line curving with our data and also which shows the fluctuation over the period of time. Hence, we decided to use other forecasting model i.e. ARIMA.

**3) Result and findings using AUTO-ARIMA and ARIMA Forecasting model:**

ARIMA stands for Auto Regressive Integrated Moving Average. There are seasonal and Non-seasonal ARIMA models that can be used for forecasting. Differencing is a method of transforming a non-stationary time series into a stationary one. This is an important step in preparing data to be used in an ARIMA model.

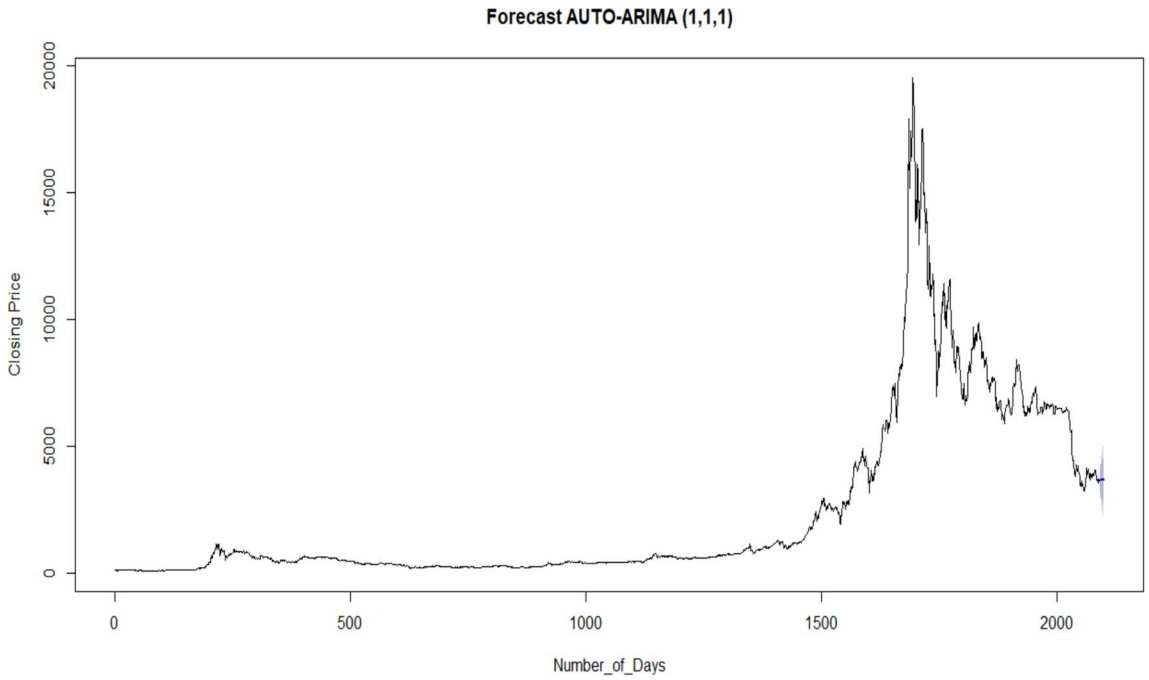
If we look at our data, in long term, the timing of these cycle is not predictable. Hence the series is stationary.

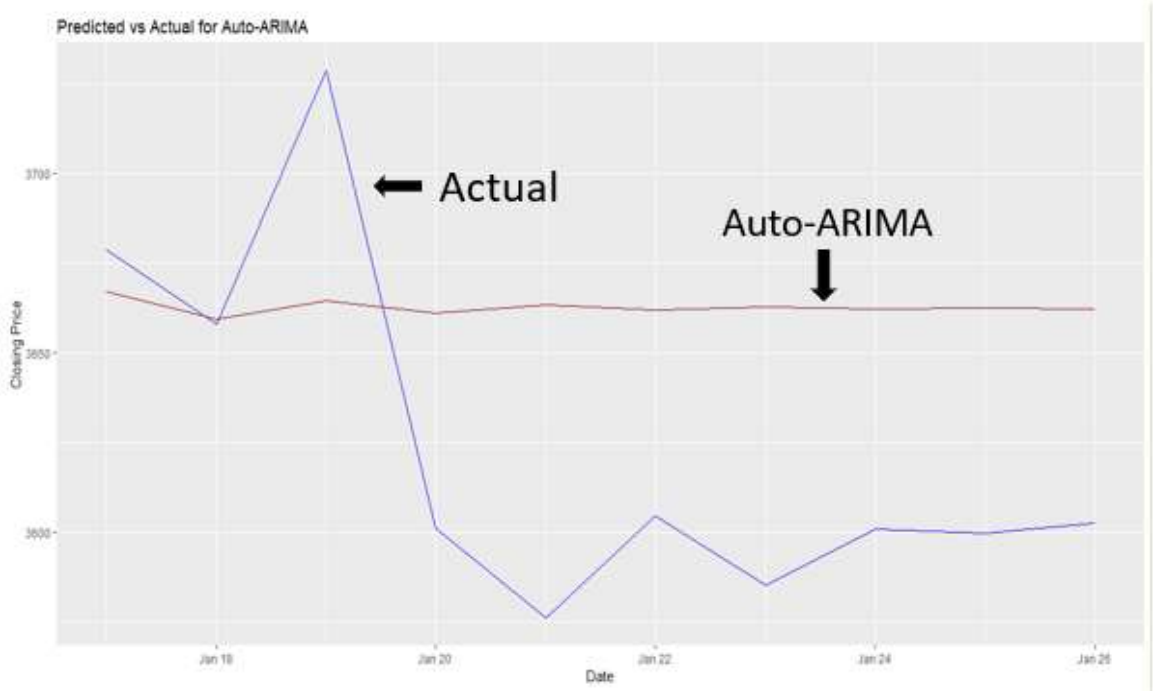



Autocorrelation refers to how correlated a time series is with its past values whereas the ACF is the plot used to see the correlation between the points, up to and including the lag unit. In ACF, the correlation coefficient is in the x-axis whereas the number of lags is shown in the y-axis. The Autocorrelation function plot will let you know how the given time series is correlated with itself.

After plotting the ACF plot we move to Partial Autocorrelation Function plots (PACF). A partial autocorrelation is a summary of the relationship between an observation in a time series with observations at prior time steps with the relationships of intervening observations removed.

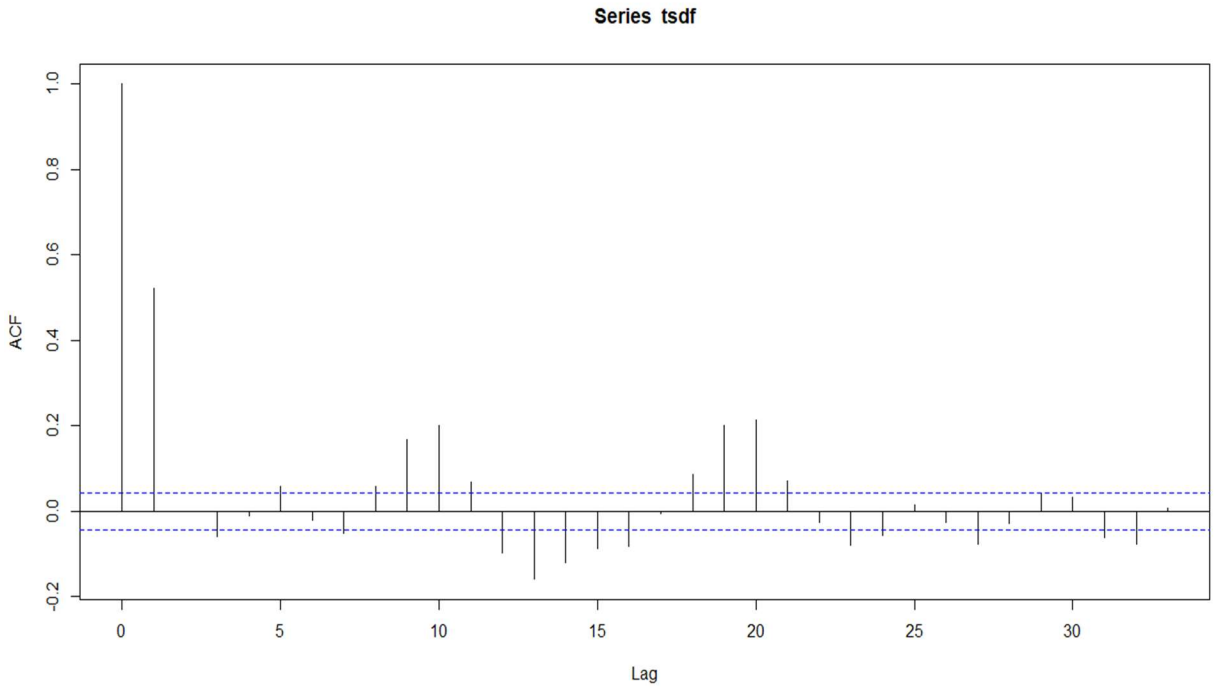
## A) Auto ARIMA (1,1,1)



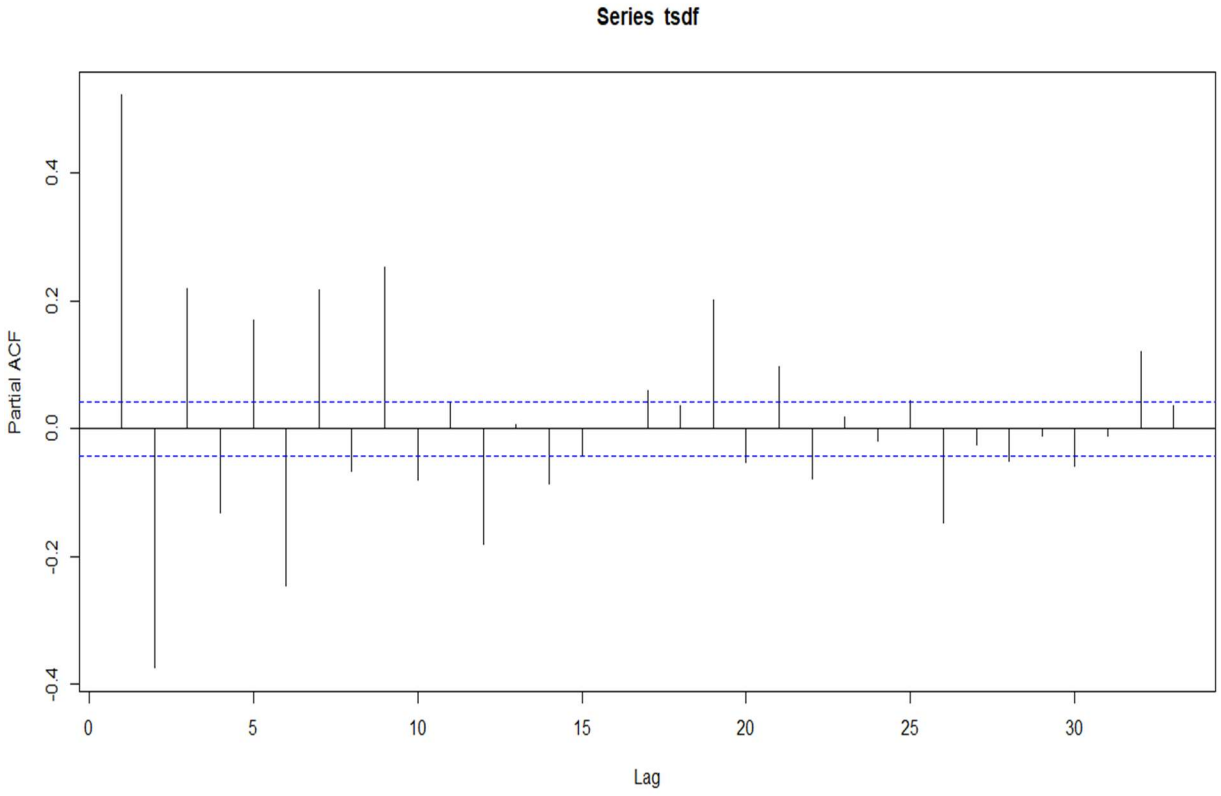




The mean accurate prediction error comes out to be 1.50% for Auto ARIMA which is good, but line is not showing the proper up and dip in the prices hence we decided to do further analysis using ACF, PACF graphs and adjusting the leg value manually to get more accurate results and graph.

## ACF ARIMA



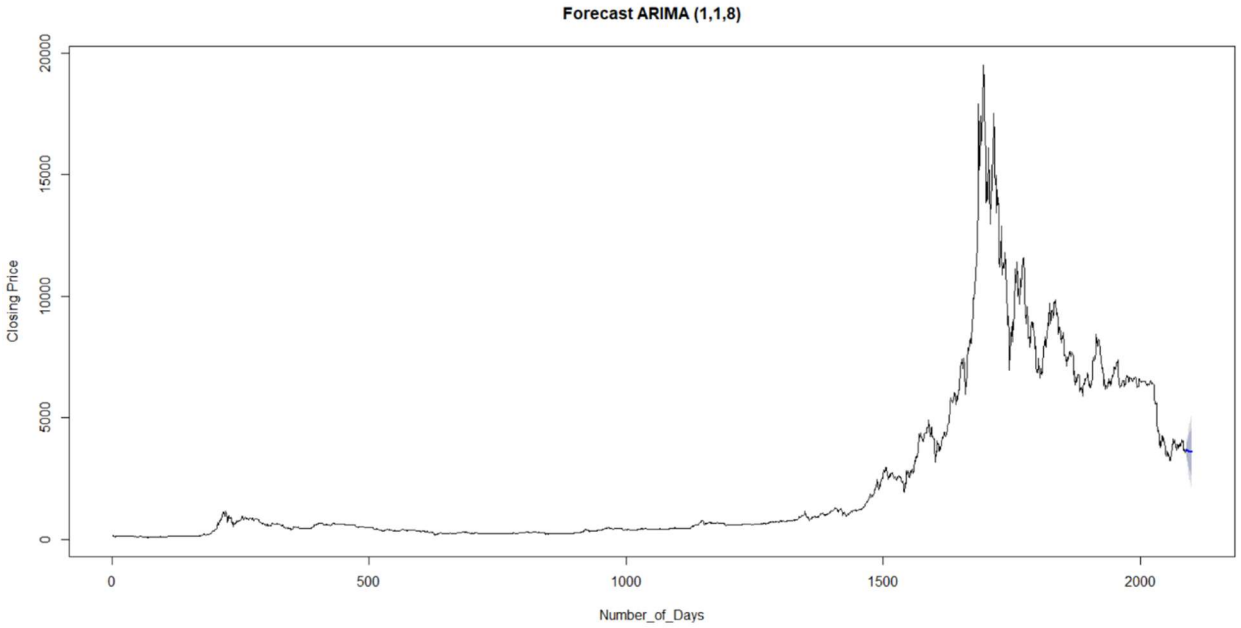

## PACF ARIMA

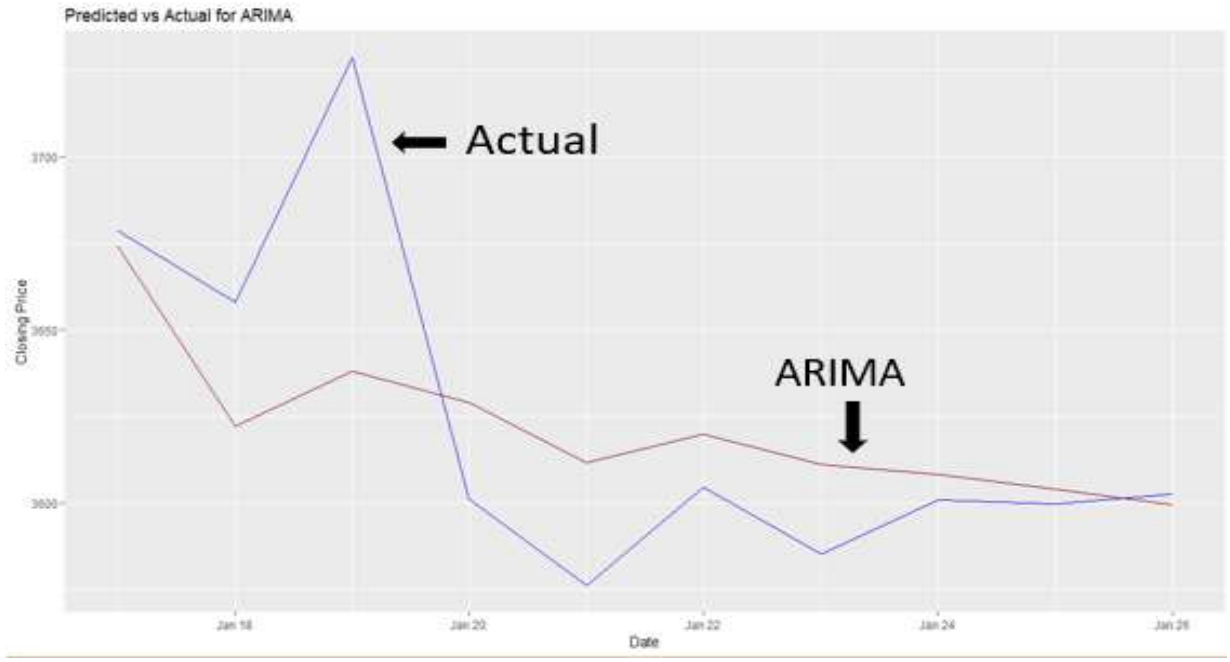


As we can see in above graphs, PACF plot drops off at lag n, then use an AR(n) model and if the drop in PACF is more gradual then we use the MA term. After plotting the ACF plot we move to Partial Autocorrelation Function plots (PACF). A partial autocorrelation is a summary of the relationship between an observation in a time series with observations at prior time steps with the relationships of intervening observations removed. The partial autocorrelation at lag k is the correlation that results after removing the effect of any correlations due to the terms at shorter lags.

As well as looking at the time plot of the data, the ACF plot is also useful for identifying non-stationary time series. For a stationary time, series, the ACF will drop to zero relatively quickly, while the ACF of non-stationary data decreases slowly. It divides the time series data into models based on time lags in an autoregressive model, the differencing and the order of the moving average model. R provides functionality between 2 kinds of model one is auto arima function which automatically fits the data to the best possible order and other one is ARIMA forecasting in which the order needs to be determined based on ACF and PACF plot. Both the models are created for forecasting and are compared for accuracy. After analyzing ACF and PACF graphs we decided to replace i-term with leg 8 which was leg 1 in Auto ARIMA model to get more accurate results.

**B) ARIMA (1,1,8)**

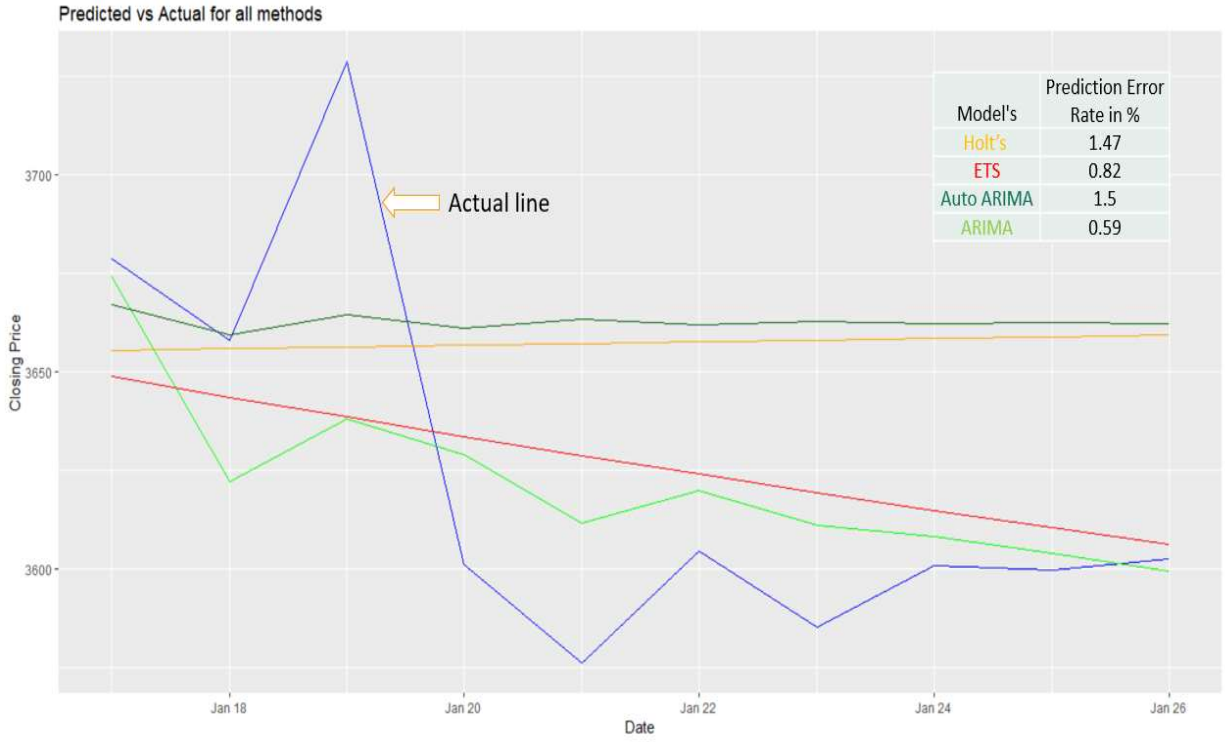



The mean accurate prediction error comes out to be 0.59% for ARIMA in which we set parameters by observing ACF and PACF graph. This can be treated as overfitted as well. But in the graph above we can see line is moving with the trend in the prices and showing the proper ups and dips in the prices. Also, the accuracy which we have derived from this ARIMA model is the best amongst all the three methods which we used.

## Conclusion

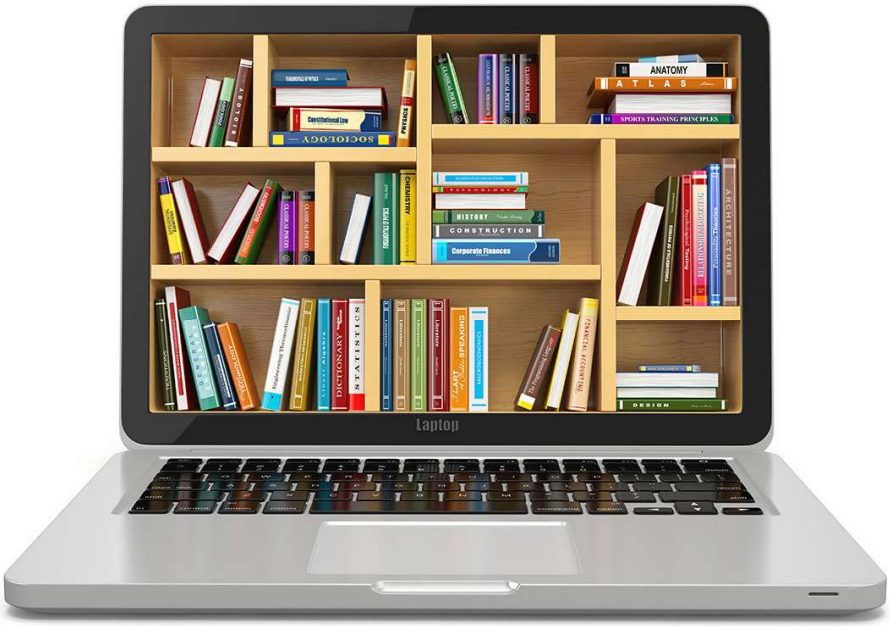
We tried to use different approaches for predicting the bitcoin prices using the time series forecasting. We first used the Holt's approach and we got the accurate prediction error of 1.47% which was good but the problem of using this model was that it was not able to predict the up and downs in the prices of bitcoins, so we shifted our way to second approach. The reason for using the Exponential Triple Smoothing (ETS) approach was the data, the data showed both trend and seasonality component. Using the ETS approach, we got the mean accurate prediction error 0.82%, which actually improved our accuracy. But, as we see the below plot, ETS predicted line was still not curving along with the actual line so due to this reason we used the ARIMA approach.




Using the ARIMA approach, we predicted 10 intervals based on selecting the parameters from ACF and PACF ARIMA model. We can see the green predicted line of ARIMA model in the above plot, here the mean prediction error comes out to be 0.59% for ARIMA in which we manually set the parameters by observing ACF and PACF model. This model has eventually got us the desired result and this model can be overfitted as well.

Finally, to conclude, the original and predicted time series for different approaches is plotted with mean error. Therefore, we were able to use different transformations and models to predict the closing price of bitcoin. As a group we understood that models may not predict the future Bitcoin price, but they certainly provide a better understanding.

## References




1) Bitcoin. https://en.wikipedia.org/wiki/Bitcoin

2) Arvindhan Rameshbabu, (2016). Forecasting of Bitcoin.

3) Time Series. https://en.wikipedia.org/wiki/Time_series

4) Swapna, (Jan 21, 2018). Cryptocurrency Data Analysis Using R.

5) Data cleansing. https://en.wikipedia.org/wiki/Data_cleansing

6) Peter Fuleky, (Oct 15, 2014). Time Series Decomposition.

7) Sangarshanan (2018). Time series Forecasting - ARIMA models

8) ARIMA models for time series forecasting. https://people.duke.edu/~rnau/411arim.htm

## Appendix

Below is the R script for Bitcoin Time Series Forecasting.

```
> library("bsts")
> library("car")
> library("caret")
> library("forecast")
> library("keras")
> library("MCMCpack")
> library("smooth")
> library("tensorflow")
> library("tseries")
> library("TTR")
> library("ggplot2")
> library("dplyr")
> library("rvest")
> library("corrplot")
> library("coinmarketcapr")
> library("formatR")
> library("yaml")
> library("googleVis")
> library("knitr")
> library("anytime")


> #scraping Bitcoin data
> url <- "https://coinmarketcap.com/currencies/bitcoin/historical-data/?start
=20130428&end=20190205"
> bitcoin_data <- url %>%
+   html() %>%
+   html_nodes(xpath='//*[@id="historical-data"]/div/div[2]/table') %>%
+   html_table()
> bitcoin_data <- bitcoin_data[[1]]
> head(bitcoin_data)
> summary(bitcoin_data)


> #scraping ethereum data
> url_2 <- "https://coinmarketcap.com/currencies/ethereum/historical-data/?st
art=20130428&end=20190205"
> ethereum_data <- url_2 %>%
+   html() %>%
+   html_nodes(xpath='//*[@id="historical-data"]/div/div[2]/table') %>%
+   html_table()
> ethereum_data <- ethereum_data[[1]]
> head(ethereum_data)
> summary(bitcoin_data)


> #formating date in standard format from character
> bitcoin_data$Date <- as.Date(anytime(bitcoin_data$Date))
> head(bitcoin_data)


> #formating market cap as numeric from character by replacing ',' from the d
ata
> bitcoin_data$`Market Cap` <- gsub(",","",bitcoin_data$`Market Cap`)
> bitcoin_data$`Market Cap` <- as.numeric(bitcoin_data$`Market Cap`)


> #formating volume as numeric from character by replacing ',' from the data
```

```
> bitcoin_data$Volume <- gsub(",","",bitcoin_data$Volume)
> bitcoin_data$Volume <- as.numeric(bitcoin_data$Volume)
> head(bitcoin_data)
> #calculating total number of NA values in each column
> colSums(is.na(bitcoin_data))


> #Difference between high and low on each day
> a <- matrix(c(0), nrow = 0, ncol = 1)
> for(i in 1:nrow(bitcoin_data)){
+   a <- rbind(a, bitcoin_data[i,3] - bitcoin_data[i,4])
+   i <- i + 1
+ }
> bitcoin_data <- cbind(bitcoin_data,a)
> head(bitcoin_data)
> summary(bitcoin_data$a)
> boxplot(bitcoin_data$a)


> #Volume has missing values#
> #Data Manipulation#
> fifty_avg <- round(mean(bitcoin_data$Volume[bitcoin_data$a < 50], na.rm = T
RUE), digits = 2)
> hun_avg <- round(mean(bitcoin_data$Volume[bitcoin_data$a > 50 & bitcoin_dat
a$a < 100], na.rm = TRUE), digits = 2)
> hf_avg <- round(mean(bitcoin_data$Volume[bitcoin_data$a > 100 & bitcoin_dat
a$a < 150], na.rm = TRUE), digits = 2)
> th_avg <- round(mean(bitcoin_data$Volume[bitcoin_data$a > 150 & bitcoin_dat
a$a < 350], na.rm = TRUE), digits = 2)
> for(i in 1:nrow(bitcoin_data)){
+   if(is.na(bitcoin_data[i,6])){
+     if(bitcoin_data$a[i] < 50){
+       bitcoin_data$Volume[i] <- fifty_avg
+     } else if(bitcoin_data$a[i] < 100){
+       bitcoin_data$Volume[i] <- hun_avg
+     } else if(bitcoin_data$a[i] < 150){
+       bitcoin_data$Volume[i] <- hf_avg
+     } else if(bitcoin_data$a[i] < 350){
+       bitcoin_data$Volume[i] <- th_avg
+     }else
+       print("Uncaught Title")
+   }
+ }
> bitcoin_data <- bitcoin_data[,-8] #removing column 'a'
> summary(bitcoin_data)


> #writing data frame into csv file
> class(bitcoin_data)
> write.table(bitcoin_data,file =  "bitcoin.csv",sep = ',',row.names = F)
> #data cleaning is not required for ethereum as it doesnt have any NA values
> colSums(is.na(ethereum_data))


> #Market Cap of top 3 cryptocurrencies
> ggplot(top_df, aes(x=top_df$id,y = top_df$market_cap_usd, fill=id)) +geom_b
ar(stat = "identity")+
+   xlab("Leading CC")+ylab("USD")+theme_minimal()+ggtitle("Market Cap of Lea
ding Cryptocurrency")
> #Variation in prices
> dfa<-data.frame(currency=c("Bitcoin","Ethereum","Ripple"),change1h=c(0.14,0
.14,0.19),change24h=c(0.09,-1.11,0.07),change7d=c(5.57,0.82,10.45))
```

```
> bar<-gvisColumnChart(dfa, xvar = "currency",yvar = c("change1h","change24h"
,"change7d"),options=list(title = "% change of prices", legend = "top", width
=1000, height=800))
> plot(bar)
> #Volume
> ggplot(top_df, aes(x=top_df$id,y = top_df$X24h_volume_usd, fill=id)) +geom_
bar(stat = "identity")+
+   xlab("Leading CC")+ylab("USD")+theme_minimal()+ggtitle("Volume")
> #Ploting bitcoin_data
> #Bitcoin Closing Price and Opening Price
> ggplot(bitcoin_data, aes(bitcoin_data$Date, bitcoin_data$`Close**`)) +
+   geom_line(color='blue') + scale_x_date("Year")+ ylim(0,20000) + ylab("Clo
sing Price")+
+   ggtitle("Bitcoin Closing Price")
> ggplot(bitcoin_data, aes(bitcoin_data$Date, bitcoin_data$`Open*`))+
+   geom_line(color='red') + scale_x_date("Year")+ ylim(0,15000) + ylab("Open
ing Price")+
+   ggtitle("Bitcoin Opening Price")


> #Boxplot and Hist of Closing Price
> breaks<-c(50,1000,3000,5000,10000,15000,20000)
> labels<-c("50-1000","1000-3000","3000-5000","5000-10000","10000-15000","150
00-20000")
> bins <- cut(bitcoin_data$`Close**`,breaks, include.lowest = T,right = F,lab
els = labels)
> plot(bins,col=3, main="Bitcoin Closing Price", xlab="Prices", ylab="Frequen
cy")
> boxplot(bitcoin_data$`Close**`,col=7, main="Boxplot of Bitcoin CP", xlab="C
losing Price",ylab="Prices" )
> #Comparing bitcoin Opening and Closing price
> ggplot(bitcoin_data, aes(bitcoin_data$Date))+ geom_line(aes(y = bitcoin_dat
a$`Open*`, colour="Open"))+
+   geom_line(aes(y = bitcoin_data$`Close**`, colour="Close"))+ scale_x_date(
"Year")+ ylim(0,20000) + ylab("Prices")+
+   ggtitle("Comparison of Bitcoin Prices")
> #Comparison of Bitcoin Market Cap and Volume
> breakv<-c(2.858e+05,2.858e+07,2.858e+09,2.858e+11)
> labelss<-c("2.858e+05-2.858e+07","2.858e+07-2.858e+09","2.858e+09-2.858e+11
")
> binss<-cut( bitcoin_data$Volume,breakv,include.lowest = T,right = F,labels
= labelss)
> boxplot(bitcoin_data$Volume,col=7, main="Boxplot of Bitcoin Volume", xlab="
Volume",ylab="Prices" )
> plot(binss,col=3,main="Bitcoin Volume", xlab="Prices", ylab="Frequency")
> breakmp<-c(7.784e+08,7.784e+09,7.784e+10,7.784e+11)
> labelsss<-c("7.784e+08-7.784e+09","7.784e+09-7.784e+10","7.784e+10-7.784e+1
1")
> binsss<-cut(bitcoin_data$`Market Cap`,breakmp,include.lowest = T,right = F,
labels =labelsss)
> plot(binsss,col=4, main="Bitcoin Market Cap", xlab="Prices", ylab="Frequenc
y")
> ggplot(bitcoin_data, aes(bitcoin_data$Date))+ geom_line(aes(y = bitcoin_dat
a$`Market Cap`, colour="Market Cap")) +
+   geom_line(aes(y = bitcoin_data$Volume, colour="Volume"))+ scale_x_date("Y
ear") +ylab("Prices")+
+   ggtitle("Comparing Bitcoin Market Cap and Volume")


> bitcoinext<- bitcoin_data[c(1:1279),]
> View(bitcoinext)
> bit_et<-cbind(bitcoinext,ethereum_data)
> colnames(ethereum_data)[colnames(ethereum_data)=="Date"]<-"Et.Date"
> colnames(ethereum_data)[colnames(ethereum_data)=="Open*"]<-"Et.Open"
```

```
> colnames(ethereum_data)[colnames(ethereum_data)=="High"]<-"Et.High"
> colnames(ethereum_data)[colnames(ethereum_data)=="Low"]<-"Et.Low"
> colnames(ethereum_data)[colnames(ethereum_data)=="Close**"]<-"Et.Close"
> colnames(ethereum_data)[colnames(ethereum_data)=="Volume"]<-"Et.Volume"
> colnames(ethereum_data)[colnames(ethereum_data)=="Market Cap"]<-"Et.Market
Cap"
> bit_et<-cbind(bitcoinext,ethereum_data)
> View(bit_et)
> ggplot(bit_et,aes(bit_et$Date,bit_et$Et.Close)) %>%
+   +geom_line()+scale_x_date("year")+ ylim(0,1500)+ylab("closing Price")+ggt
itle("Closing Price of Ethereum")
> ggplot(bit_et,aes(bit_et$Date,bit_et$Et.Open))+
+   geom_line() + scale_x_date("Year")+ ylim(0,1500) + ylab("Opening Price")+
ggtitle("Opening price of Ethereum")


> ggplot()+geom_line(data = bit_et,aes(Date, bit_et$`Open*`),color="green")+
+   geom_line(data = bit_et,aes(Date, bit_et$Et.Open),color="red")+ylab("Open
ing Price")+
+   xlab("Year")+ggtitle("Bitcoin OP vs Ethereum OP")
> ggplot()+geom_line(data = bit_et,aes(Date, bit_et$`Close**`), color="green"
)+
+   geom_line(data = bit_et,aes(Date, bit_et$Et.Close),color="red")+ylab("Clo
sing Price")+
+   xlab("Year")+ggtitle("Bitcoin CP vs Ethereum CP")
> ggplot(bitcoin_data, aes(bitcoin_data$Date))+ geom_line(aes(y = bitcoin_dat
a$High, colour="High"))+
+   geom_line(aes(y = bitcoin_data$Low, colour="Low"))+ scale_x_date("Year")+
ylim(0,20000) + ylab("Prices")+
+   ggtitle("Comparison of Bitcoin Trends")
> ggplot(bit_et, aes(bit_et$Date))+ geom_line(aes(y = bit_et$Et.High, colour=
"High"))+
+   geom_line(aes(y = bit_et$Et.Low, colour="Low"))+ scale_x_date("Year")+ yl
im(0,1500) + ylab("Prices")+
+   ggtitle("Comparison of Ethereum Trends")


> #correlation Matrix
> bitcoin_data %>%
+   select(`Open*`,`Close**`,High,Low,Volume,`Market Cap`) %>%
+   model.matrix(~.-1, .) %>%
+   cor(method = "spearman") %>%
+   corrplot(type="lower", method = "number", tl.col = "black", diag=FALSE, t
l.cex = 0.9, number.cex = 0.9)


> #spliting data into train and test set
> row.num <- c(nrow(bitcoin_data) : 21 )
> train_bitcoin <- bitcoin_data[row.num,]
> test_bitcoin <- bitcoin_data[20:1,]
> #plot of train dataset
> ggplot(train_bitcoin, aes(train_bitcoin$Date, train_bitcoin$`Close**`)) +
+   geom_line(color = 'blue') + scale_x_date("year")+ ylim(0,20000) + ylab("C
losing Price") +
+   ggtitle('train_dataset')
> #plot of test dataset
> ggplot(test_bitcoin, aes(test_bitcoin$Date, test_bitcoin$`Close**`)) +
+   geom_line(color = 'blue') + scale_x_date("year")+ ylim(3000,4000) + ylab(
"Closing Price") +
+   ggtitle('test_dataset')
```

```
> #forecasting model with closing prices of bitcoin
> bitcoin_time_series  <- ts(train_bitcoin$`Close**`,frequency = 365.25,start
= c(2013,4,27))
> bitcoin_time_series_decompose <- decompose(bitcoin_time_series)
> plot(bitcoin_time_series_decompose)
> #lets plot linear model to our data
> bitcoin_data_lm  <- data.frame(close=bitcoin_data$`Close**`,
+                                open=log(bitcoin_data$`Open*`+1),
+                                high=log(bitcoin_data$High),
+                                low=log(bitcoin_data$Low+1),
+                                market=log(bitcoin_data$`Market Cap`+1))
> fit <- step(lm(close ~ open  + high
+                + low + market, data=bitcoin_data_lm))
> summary(fit)
> plot(fitted(fit),type = "p", bitcoin_data_lm$close,ylab="Closing Price", xl
ab="Predicted Closing price", main="Linear Model")


> #Initially model is constructed using Holt's algorithm and this model is
> #used to predict the forecast prices of bitcoins for next 10 days.
> dim(train_bitcoin)
> holt_model <- holt(train_bitcoin[1000:2090,'Close**'],type="additive", damp
ed =F, h=10)
> holt_forecast <- forecast(holt_model,h = 10)
> holt_df <- as.data.frame(holt_forecast)
> #Forecast plot
> plot(holt_forecast, ylab = 'Closing Prices(in USD)', xlab = 'Total number o
f Days' , main = 'Holts Forecast')
> holtfdf <- cbind(test_bitcoin[1:10,], holt_df[,1])
> testdata <- test_bitcoin[1:10,5]
> accuracy(holt_df[,1], testdata) #accuracy
> ggplot() + geom_line(data = holtfdf, aes(Date, holtfdf[,5]), color = "blue"
) + ylab('Closing Price') +
+   geom_line(data = holtfdf, aes(Date, holtfdf[,8]), color = "Dark Red") +
+   ggtitle('Predicted vs Actual for Holts Method')


> ETS <- ets((train_bitcoin[,'Close**']),allow.multiplicative.trend=TRUE)
> ETS_forecast <- forecast(ETS, h = 10)
> ETS_df <- as.data.frame(ETS_forecast)
> plot(forecast(ETS, h =10), ylim = c(0,20000), xlab = 'Closing Prices(in USD
)',
+              ylab = 'Total number of Days' ,main = 'ETS Forecast')
> ETS_p <- predict(ETS, n.ahead = 10, prediction.interval = T, level = 0.95)
> testdata <- test_bitcoin[1:10,5]
> accuracy(ETS_df[,1], testdata)
> etsfdf <- cbind(test_bitcoin[1:10,], ETS_df[,1])
> ggplot() + geom_line(data = etsfdf, aes(Date, etsfdf[,5]), color = "blue")
+ ylab('Closing Price') +
+   geom_line(data = etsfdf, aes(Date, etsfdf[,8]), color = "Dark Red") +
+   ggtitle('Predicted vs Actual for ETS method')


> tsdf <- diff(train_bitcoin[,5], lag = 2)
> tsdf <- tsdf[!is.na(tsdf)]
> adf.test(tsdf)
> plot(tsdf,type = "l", xlab = 'Closing Price',ylab='Number_of_Days',ylim = c
(-5000, 5000), main='Difference Plot')


> #Auto- and Cross- Covariance and -Correlation Function Estimation
> #plots the estimates
> acf(tsdf)
```

```
> #pacf is used for partial auto correlation function
> pacf(tsdf)


> #fit best ARIMA model
> auto.arima(train_bitcoin[,5])
> bitcoin_time_series_forecast_auto <- forecast(auto.arima(train_bitcoin[,5])
, h=10)
> plot(bitcoin_time_series_forecast_auto, ylab = 'Closing Price', xlab = 'Num
ber_of_Days', main='Forecast AUTO-ARIMA (1,1,1)')
> bts_f_df_auto <- as.data.frame(bitcoin_time_series_forecast_auto)
> testdata <- test_bitcoin[1:10,5]
> accuracy(bts_f_df_auto[,1],testdata)
> gegefct_auto <- cbind(test_bitcoin[1:10,], bts_f_df_auto[,1])
> ggplot() + geom_line(data = gegefct_auto, aes(Date, gegefct_auto[,5]), colo
r = "blue") + ylab('Closing Price') +
+   geom_line(data = gegefct_auto, aes(Date, gegefct_auto[,8]), color = "Dark
Red") +
+   ggtitle('Predicted vs Actual for Auto-ARIMA')
> #predicting 10 interval based on selecting parameters from ACF and P-ACF AR
IMA model
> bitcoin_time_series_forecast <- forecast(arima(train_bitcoin[,5], order = c
(1,1,8)), h=10)
> arima(train_bitcoin[,5], order = c(1,1,8))
> bitcoin_time_series_forecast
> plot(bitcoin_time_series_forecast)
> bts_f_df <- as.data.frame(bitcoin_time_series_forecast)
> testdata <- test_bitcoin[1:10,5]
> accuracy(bts_f_df[,1],testdata)
> gegefct <- cbind(test_bitcoin[1:10,], bts_f_df[,1])
> plot(bitcoin_time_series_forecast,ylab = 'Closing Price', xlab = 'Number_of
_Days', main='Forecast ARIMA (1,1,8)')
> ggplot() + geom_line(data = gegefct, aes(Date, gegefct[,5]), color = "blue"
) + ylab('Closing Price') +
+   geom_line(data = gegefct, aes(Date, gegefct[,8]), color = "Dark Red") +
ggtitle('Predicted vs Actual for ARIMA')
> all <- cbind(test_bitcoin[1:10,],bts_f_df[,1],ETS_df[,1],holt_df[,1], bts_f
_df_auto[,1])
> head(all)
> ggplot() + geom_line(data = all, aes(Date, all[,5]), color = "blue") + ylab
('Closing Price') +
+   geom_line(data = all, aes(Date, all[,8]), color = "green") +
+   geom_line(data = all, aes(Date, all[,9]), color = "red") +
+   geom_line(data = all, aes(Date, all[,10]), color = "orange") +
+   geom_line(data = all, aes(Date, all[,11]), color = "dark green") +
+   ggtitle('Predicted vs Actual for all methods')
```