**ECN 580-01 Project: Using Historical Trading data for predicting stock prices by the ARIMA model**

Shivangi Gupta

A25266618

**Description of Research question**

Historical Trading data, which are inevitably associated with the framework of causality both financially and theoretically, were widely used to predict stock market values. The aim of this study was to investigate the performances of forecasting stock prices by historical trading data from Yahoo finance using R package (GetBatchSymbols) over a timespan of 10 years. In addition, Auto-correlation function and Partial Auto-correlation function were used to see if the series are correlated with its lags and finally ARIMA model is used for predicting stock prices. The stocks in focus for this project were AAPL (Apple), GOOGL (Alphabet Inc.), AMZ (Amazon), DIS (Disney) and WMT (Walmart corporations).

**Formulation of Hypothesis**

The only hypothesis used for this research was to check for stationarity. A stationary time series means a time series without trend, one having a constant mean and variance over time, which makes it easy for predicting values.

Null Hypothesis: H0 = time series has a unit root that it is a non-stationary process.

H1 = time series does not have a unit root that it is a stationary process.

**Description of Methods**

1. **Testing for stationarity:** To test for stationarity, used the Augmented Dickey-Fuller unit root test. The p-value resulting from the ADF test has to be less than 0.05 or 5% for a time series to be stationary. If the p-value is greater than 0.05 or 5%, you conclude that the time series has a unit root which means that it is a non-stationary process.
2. **Differencing:** To convert a non-stationary process to a stationary process, we apply the differencing method. Differencing a time series means finding the differences between consecutive values of a time series data. For this project, I used “first order differencing”.

1. **Auto-correlation and Partial Auto-correlation function:** The plots are useful visual tool

for determining whether a series is stationary or not. These plots can also be used to choose order parameters for ARIMA model but in this project, it is used to check the stationarity of the dataset. ACF plots display correlation between a series and its lags. Partial autocorrelation plots (PACF), as the name suggests, display correlation between a variable and its lags that is not explained by previous lags. If the series is correlated with its lags then, generally, there are some trend or seasonal components and therefore its statistical properties are not constant over time.

1. **Training and Test data:** The data is divided into two parts. It comprises of training data which is 80% of the total data and test data which is the remaining 20%. The training dataset is used to find the best fitted ARIMA model and the test data is used to check the accuracy of the forecasted values from that fitted model.
2. **ARIMA model:** ARIMA stands for Autoregressive Integrated Moving Average. ARIMA is also known as Box-Jenkins approach. Box and Jenkins claimed that non-stationary data can be made stationary by differencing the series, Yt.. The model combines three basic models:

* **AutoRegression (AR)** – In auto-regression the values of a given time series data are regressed on their own lagged values, which is indicated by the “p” value in the model.
* **Differencing (I-for Integrated)** – This involves differencing the time series data to remove the trend and convert a non-stationary time series to a stationary one. This is indicated by the “d” value in the model. If d = 1, it looks at the difference between two time series entries, if d = 2 it looks at the differences of the differences obtained at d =1, and so forth.
* **Moving Average (MA)** – The moving average nature of the model is represented by the “q” value which is the number of lagged values of the error term.

This model is called Autoregressive Integrated Moving Average or ARIMA (p, d, q) of Yt. In this project I have used the auto.arima function in R, which finds the best fit for your data on its own, and the user doesn’t have to find and define the (p, d, q) parameter.

1. **Forecasting:**  Estimated the best fitted ARIMA model on the training data set using auto.arima function in R and then used that fitted model to forecast the values of the test data set using a forecasting function.
2. **Error Calculation:** In the end, computed standard error to check the accuracy of the forecasted values (predicted stock value) in comparison to test data (actual stock price). Two methods were used for error calculation:

* Absolute error = mean [abs {(actual value – forecasted value)/actual value}]
* RMSE = sqrt [mean {(actual value – forecasted value) ^2}]

**Quality of Analysis: Hypothesis Testing and ARIMA modeling**

**Testing for stationarity**

The Augmented Dickey-Fuller Test was used to check for stationarity. The following data shows the ADF test on the stock price, log of stock price and the first difference on the log of stock price, in order to check for stationarity.

**ADF Test on Apple Stock**

|  |
| --- |
|  |
| **Figure 1: It shows the ADF test on the closing price of Apple stock. As the p-value > 0.05, hence the time series has a unit root which means that it is a non-stationary process.** |
|  |
| **Figure 2: It shows the ADF test on the log of closing price of Apple stock. As the p-value > 0.05, hence the time series has a unit root which means that it is a non-stationary process.** |
|  |
| **Figure 3: It shows the ADF test on the first differences of the log of closing price of Apple stock. As the p-value < 0.05, hence the time series does not have a unit root which means that it is a stationary process.** |

**ADF test on Google Stock**

|  |
| --- |
|  |
| **Figure 4: It shows the ADF test on the closing price of Google stock. As the p-value > 0.05, hence the time series has a unit root which means that it is a non-stationary process.** |
|  |
| **Figure 5: It shows the ADF test on the log of closing price of Google stock. As the p-value > 0.05, hence the time series has a unit root which means that it is a non-stationary process.** |
|  |
| **Figure 6: It shows the ADF test on the first differences of the log of closing price of Google stock. As the p-value < 0.05, hence the time series does not have a unit root which means that it is a stationary process.** |

**ADF Test on Walmart Stock**

|  |
| --- |
|  |
| **Figure 7: It shows the ADF test on the closing price of Walmart stock. As the p-value > 0.05, hence the time series has a unit root which means that it is a non-stationary process.** |
|  |
| **Figure 8: It shows the ADF test on the log of closing price of Walmart stock. As the p-value > 0.05, hence the time series has a unit root which means that it is a non-stationary process.** |
|  |
| **Figure 9: It shows the ADF test on the first differences of the log of closing price of Walmart stock. As the p-value < 0.05, hence the time series does not have a unit root which means that it is a stationary process.** |

**ADF Test on Disney Stock**

|  |
| --- |
|  |
| **Figure 10: It shows the ADF test on the closing price of Disney stock. As the p-value > 0.05, hence the time series has a unit root which means that it is a non-stationary process.** |
|  |
| **Figure 11: It shows the ADF test on the log of closing price of Disney stock. As the p-value > 0.05, hence the time series has a unit root which means that it is a non-stationary process.** |
|  |
| **Figure 12: It shows the ADF test on the first differences of the log of closing price of Disney stock. As the p-value < 0.05, hence the time series does not have a unit root which means that it is a stationary process.** |

**ADF Test on Amazon stock**

|  |
| --- |
|  |
| **Figure 13: It shows the ADF test on the closing price of Amazon stock. As the p-value > 0.05, hence the time series has a unit root which means that it is a non-stationary process.** |
|  |
| **Figure 14: It shows the ADF test on the log of closing price of Amazon stock. As the p-value > 0.05, hence the time series has a unit root which means that it is a non-stationary process.** |
|  |
| **Figure 15: It shows the ADF test on the first differences of the log of closing price of Amazon stock. As the p-value < 0.05, hence the time series does not have a unit root which means that it is a stationary process.** |

**ARIMA Model**

The auto.arima function finds the best fitted model for the training data, which in this project is the log of closing price of the stock data.

**ARIMA on Apple Stock**

|  |
| --- |
|  |
| **Figure 16: The figure above shows the function finding the best fitted model for the training data. The best model found for Apple Stock ARIMA (0,1,0) with drift** |
|  |
| **Figure 17: Summary of Best Fitted ARIMA Model for Apple Stock** |

**ARIMA on Google Stock**

|  |
| --- |
|  |
| **Figure 18: The figure above shows the function finding the best fitted model for the training data. The best model found for Google Stock ARIMA (0,1,0) with drift** |
|  |
| **Figure 19: Summary of Best Fitted ARIMA Model for Google Stock** |

**ARIMA on Walmart Stock**

|  |
| --- |
|  |
| **Figure 20: The figure above shows the function finding the best fitted model for the training data. The best model found for Walmart Stock ARIMA (0,1,0) with drift** |
|  |
| **Figure 21: Summary of Best Fitted ARIMA Model for Walmart Stock** |

**ARIMA on Disney Stock**

|  |
| --- |
|  |
| **Figure 22: The figure above shows the function finding the best fitted model for the training data. The best model found for Disney Stock ARIMA (0,1,0) with drift** |
|  |
| **Figure 23: Summary of Best Fitted ARIMA Model for Disney Stock** |

**ARIMA on Amazon Stock**

|  |
| --- |
|  |
| **Figure 24: The figure above shows the function finding the best fitted model for the training data. The best model found for Amazon Stock ARIMA (0,1,0) with drift** |
|  |
| **Figure 25: Summary of Best Fitted ARIMA Model for Amazon Stock** |

**RESULTS**

|  |
| --- |
|  |
| **Figure 26: Plot of closing price vs Date** |
|  |
| **Figure 27: Plot of Closing price vs Date for Apple, Disney and Walmart** |

From figure 26 and 27, random walk can be observed in the stock prices of all company and can be removed by using differences to remove it.

|  |
| --- |
|  |
| **Figure 28: Time series plot of Apple stock price vs Time** |
|  |
| **Figure 29: Time series plot of Google stock price vs Time** |
|  |
| **Figure 30: Time series plot of Walmart stock price vs Time** |
|  |
| **Figure 31: Time series plot of Disney stock price vs Time** |
|  |
| **Figure 32: Time series plot of Amazon stock price vs Time** |

**ACF and PACF plots**

|  |  |
| --- | --- |
|  |  |
| **Figure 33: ACF plot of Apple Stock** | **Figure 34: PACF plot of Apple Stock** |
|  |  |
| **Figure 35: ACF plot of Google Stock** | **Figure 36: PACF plot of Google Stock** |
|  |  |
| **Figure 37: ACF plot of Walmart Stock** | **Figure 38: PACF plot of Walmart Stock** |
|  |  |
| **Figure 39: ACF plot of Disney Stock** | **Figure 40: PACF plot of Disney Stock** |
|  |  |
| **Figure 41: ACF plot of Amazon Stock** | **Figure 42: PACF plot of Amazon Stock** |

It can be observed from all the ACF and PACF plots, that the correlation between the series decreases as the number of lags increases, showing that the statistical properties of the data is constant over time.

**Accuracy Results from ARIMA model fit**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Company** | **ARIMA model** | **ME** | **RMSE** | **MAE** | **MPE** | **MAPE** | **MASE** | **ACF1** |
| **Apple** | ARIMA (0,1,0)  With drift | 6.751829e-06 | 0.0356 | 0.0278 | 0.0076 | 0.677 | 0.087 | -0.025 |
| **Google** | ARIMA (0,1,0) | 0.0036 | 0.0332 | 0.0244 | 0.0598 | 0.413 | 0.138 | -0.026 |
| **Walmart** | ARIMA (0,1,0) | 0.0008 | 0.0206 | 0.0154 | 0.0185 | 0.371 | 0.147 | -0.066 |
| **Disney** | ARIMA(0,1,0) (1,0,0)[52] with drift | 3.35962e-06 | 0.0311 | 0.0226 | 0.0031 | 0.587 | 0.103 | -0.108 |
| **Amazon** | ARIMA(0,1,0) (2,0,0)[52] with drift | 0.00013 | 0.045 | 0.0326 | 0.0002 | 0.599 | 0.106 | -0.071 |

**Table 1 : Accuracy results from ARIMA best model fit for each stock**

From the table, it can be observed that all the accuracies are smaller than 1, hence the model is good fit for the data.

**Forecasts Plot**

|  |
| --- |
|  |
| **Figure 43: Forecast Plot for Apple Stock** |
|  |
| **Figure 44: Forecast Plot for Google Stock** |
|  |
| **Figure 45: Forecast Plot for Walmart Stock** |
|  |
| **Figure 46: Forecast Plot for Disney Stock** |
|  |
| **Figure 47: Forecast Plot for Amazon Stock** |

**Accuracy of Forecasted Values in comparison to test data**

|  |  |  |
| --- | --- | --- |
| **COMPANY** | **Absolute Error** | **Root Mean Squared Error** |
| **Apple** | 0.11116 | 30.2125 |
| **Google** | 0.203478 | 245.078919 |
| **Walmart** | 0.193743 | 20.09764 |
| **Disney** | 0.33087 | 39.39119 |
| **Amazon** | 0.13765 | 294.4022 |

**Table 2: Error between Actual data and Forecasted data values.**

**CONCLUSION**

The ARIMA model was successful in predicting the stock prices, but due to huge Root Mean Squared Error between Actual and Predicted Data value for Google and Amazon, it means there can be other factors that might be contributing to change in prices that should be taken into account while using the ARIMA model for better prediction. Overall, the model performed good and can be good starting point in stock price prediction.

**STRUCTURE OF THE PAPER**

The paper I replicated was “Using Internet Search Trends and Historical Trading Data for Predicting Stock Markets by the Least Squares Support Vector Regression Model”. The paper used Google Trends to get the Historical trading data over a time span of 5 years. It used Correlation Feature selection method to select independent variables used Least squares support vector regression for predicting stock prices. In addition, the mean absolute percentage error (MAPE) and mean absolute error (MAE) were used to measure the performance of LSSVR models.

**LANGUAGE**

R was used as language for this project.

**#----------------------------------------------------CODE------------------------------------------------------------------**

**#BatchGetSymbols downloads financial data.**

library(BatchGetSymbols)

library(esquisse)

library(forecast)

library(tseries)

library(MASS)

library(lubridate)

library(Metrics)

library(quantmod)

library(ggplot2)

**#this gives you 1 year of data, use Sys.Date()-3653 for 10 years of data (3650 + 2 or 3 leap years)**

**#you can set first.date & last.date to anything you want**

**#if you choose specific dates enter them in quotes in the following format: "YYYY-MM-DD"**

**#set dates**

first.date <- Sys.Date()-3653

last.date <- as.Date('2019-04-12')

freq.data <- 'weekly'

**#set tickers**

**#AAPL<-Apple, GOOGL<-Alphabet Inc.**

**#WMT<- Walmart, DIS<-The Walt Disney Company, AMZN<-Amazon.com Inc.**

tickers <- c('AAPL','GOOGL','WMT','DIS','AMZN')

**#download the data: if you have a lot this can take time**

l.out <- BatchGetSymbols(tickers = tickers,

first.date = first.date,

last.date = last.date,

freq.data = freq.data)

**#l.out is a list of 2 data frames**

**#the prices are in the second one so extract it into a separate data frame to use**

stockdata <- l.out[[2]]

**#Plotting stock data vs Date**

ggplot(data = stockdata) +

aes(x = ref.date, y = price.close, color = ticker) +

geom\_line() +

labs(title = 'Closing Price vs Date',

x = 'Date',

y = 'Closing Price') +

theme\_minimal()

**#Plotting stock data vs Date (excluding Amazon and Google)**

esquisser(stockdata)

**#---------------------------Apple Sock------------------------------------------------------------------**

**#Subsettung the stock data based on tickers**

apple\_data <- subset(stockdata, ticker=="AAPL", select = c(ref.date,price.close))

apple\_price <- apple\_data$price.close

apple\_ts <- ts(apple\_price, start = decimal\_date(ymd("2009-04-13")), frequency = 365.25/7)

**#Plotting the data sets**

plot(apple\_ts, main = "Apple Stock Closing price vs Time", ylab = "Closing price" )

**#convert to ln format**

apple\_lnprice <- log(apple\_price)

**#Moving average on ln of stock price**

apple\_difflnprice <- diff(apple\_lnprice,1)

**#Dickey-Fuller Test**

adf.test(apple\_price)

adf.test(apple\_lnprice)

adf.test(apple\_difflnprice)

**#ACF, PACF**

acf(apple\_lnprice, lag.max=50, main="ACF plot of Apple stock")

pacf(apple\_lnprice, lag.max=50, main="PACF plot of Apple stock")

**#breaking the dataset into training and testing**

breakpoint <- floor(dim(apple\_data)[1]\*0.8)

**#Divide into training and test data**

apple\_train <- apple\_lnprice[1:breakpoint]

apple\_test <- apple\_lnprice[breakpoint:dim(apple\_data)[1]]

**#Time series and auto.arima on ln price**

apple\_pricearima <- ts(apple\_train, start = decimal\_date(ymd("2009-04-13")), frequency = 365.25/7)

apple\_fit <- auto.arima(apple\_pricearima, allowdrift = TRUE, trace = TRUE, test = "adf", ic = "bic")

summary(apple\_fit)

**#Forecasted values from ARIMA**

apple\_forecastedvalues <- forecast(apple\_fit, h=106)

plot(apple\_forecastedvalues, include= 100, xlab = "Time",ylab = "log(closing price)")

**#Converting the log values to exponential**

apple\_value <- as.numeric(apple\_forecastedvalues$mean)

apple\_predictedval <- exp(apple\_value)

apple\_actual\_value <- exp(apple\_test)

**#calculating the error between predicted and test data**

apple\_rmse <- rmse(apple\_actual\_value,apple\_predictedval)

apple\_error <- mean(abs((apple\_actual\_value - apple\_predictedval)/apple\_actual\_value))

**#---------------------------------------------Google Data------------------------------------------**

**#Subsettung the stock data based on tickers**

google\_data <- subset(stockdata, ticker=="GOOGL", select = c(ref.date,price.close))

google\_price <- google\_data$price.close

google\_ts <- ts(google\_price, start = decimal\_date(ymd("2009-04-13")), frequency = 365.25/7)

**#Plotting the data sets**

plot(google\_ts, main = "Google Stock Closing price vs Time", ylab = "Closing price" )

**#convert to ln format**

google\_lnprice <- log(google\_price)

**#Moving average on ln of stock price**

google\_difflnprice <- diff(google\_lnprice,1)

**#Dickey-Fuller Test**

adf.test(google\_price)

adf.test(google\_lnprice)

adf.test(google\_difflnprice)

**#ACF, PACF**

acf(google\_lnprice, lag.max=50, main="ACF plot of Google stock")

pacf(google\_lnprice, lag.max=50, main="PACF plot of Google stock")

**#breaking the dataset into training and testing**

breakpoint <- floor(dim(google\_data)[1]\*0.8)

**#Divide into training and test data**

google\_train <- google\_lnprice[1:breakpoint]

google\_test <- google\_lnprice[breakpoint:dim(google\_data)[1]]

**#Time series and auto.arima on ln price**

google\_pricearima <- ts(google\_train, start = decimal\_date(ymd("2009-04-13")), frequency = 365.25/7)

google\_fit <- auto.arima(google\_pricearima, allowdrift = TRUE, test = "adf", trace = TRUE, ic = "bic")

summary(google\_fit)

**#Forecasted values from ARIMA**

google\_forecastedvalues <- forecast(google\_fit, h=106)

plot(google\_forecastedvalues, include= 100, xlab = "Time",ylab = "log(closing price)")

**#Converting the log values to exponential**

google\_value <- as.numeric(google\_forecastedvalues$mean)

google\_predictedval <- exp(google\_value)

google\_actual\_value <- exp(google\_test)

**#calculating the error between predicted and test data**

google\_error <- mean(abs((google\_actual\_value - google\_predictedval)/google\_actual\_value))

google\_rmse <- rmse(google\_actual\_value,google\_predictedval)

**#----------------------------------------------Walmart Data----------------------------------------**

**#Subsetting the stock data based on tickers**

walmart\_data <- subset(stockdata, ticker=="WMT", select = c(ref.date,price.close))

wmt\_price <- walmart\_data$price.close

wmt\_ts <- ts(wmt\_price, start = decimal\_date(ymd("2009-04-13")), frequency = 365.25/7)

**#Plotting the data sets**

plot(wmt\_ts, main = "Walmart Stock Closing price vs Time", ylab = "Closing price" )

**#convert to ln format**

wmt\_lnprice <- log(wmt\_price)

**#Moving average on ln of stock price**

wmt\_difflnprice <- diff(wmt\_lnprice,1)

**#Dickey-Fuller Test**

adf.test(wmt\_price)

adf.test(wmt\_lnprice)

adf.test(wmt\_difflnprice)

**#ACF, PACF**

acf(wmt\_lnprice, lag.max=50, main="ACF plot of Walmart stock")

pacf(wmt\_lnprice, lag.max=50, main="PACF plot of Walmart stock")

**#breaking the dataset into training and testing**

breakpoint <- floor(dim(walmart\_data)[1]\*0.8)

**#Divide into training and test data**

wmt\_train <- wmt\_lnprice[1:breakpoint]

wmt\_test <- wmt\_lnprice[breakpoint:dim(walmart\_data)[1]]

**#Time series and auto.arima on ln price**

wmt\_pricearima <- ts(wmt\_train, start = decimal\_date(ymd("2009-04-13")), frequency = 365.25/7)

wmt\_fit <- auto.arima(wmt\_pricearima, allowdrift = TRUE, trace = TRUE, test = "adf", ic = "bic")

summary(wmt\_fit)

**#Forecasted values from ARIMA**

wmt\_forecastedvalues <- forecast(wmt\_fit, h=106)

plot(wmt\_forecastedvalues, include= 100, xlab = "Time",ylab = "log(closing price)")

**#Converting the log values to exponential**

wmt\_value <- as.numeric(wmt\_forecastedvalues$mean)

wmt\_predictedval <- exp(wmt\_value)

wmt\_actual\_value <- exp(wmt\_test)

**#calculating the error between predicted and test data**

wmt\_error <- mean(abs((wmt\_actual\_value - wmt\_predictedval)/wmt\_actual\_value))

wmt\_rmse <- rmse(wmt\_actual\_value,wmt\_predictedval)

**#---------------------------------------------Disney data------------------------------------------**

**#Subsetting the stock data based on tickers**

disney\_data <- subset(stockdata, ticker=="DIS", select = c(ref.date,price.close))

dis\_price <- disney\_data$price.close

dis\_ts <- ts(dis\_price, start = decimal\_date(ymd("2009-04-13")), frequency = 365.25/7)

**#Plotting the data sets**

plot(dis\_ts, main = "Disney Stock Closing price vs Time", ylab = "Closing price" )

**#convert to ln format**

dis\_lnprice <- log(dis\_price)

**#Moving average on ln of stock price**

dis\_difflnprice <- diff(dis\_lnprice,1)

**#Dickey-Fuller Test**

adf.test(dis\_price)

adf.test(dis\_lnprice)

adf.test(dis\_difflnprice)

**#ACF, PACF**

acf(dis\_lnprice, lag.max=50, main="ACF plot of Disney stock")

pacf(dis\_lnprice, lag.max=50, main="PACF plot of Disney stock")

**#breaking the dataset into training and testing**

breakpoint <- floor(dim(disney\_data)[1]\*0.8)

**#Divide into training and test data**

dis\_train <- dis\_lnprice[1:breakpoint]

dis\_test <- dis\_lnprice[breakpoint:dim(disney\_data)[1]]

**#Time series and auto.arima on ln price**

dis\_pricearima <- ts(dis\_train, start = decimal\_date(ymd("2009-04-13")), frequency = 365.25/7)

dis\_fit <- auto.arima(dis\_pricearima, allowdrift = TRUE, trace = TRUE, test = "adf", ic = "bic")

summary(dis\_fit)

**#Forecasted values from ARIMA**

dis\_forecastedvalues <- forecast(dis\_fit, h=106)

plot(dis\_forecastedvalues, include= 100, xlab = "Time",ylab = "log(closing price)")

**#Converting the log values to exponential**

dis\_value <- as.numeric(dis\_forecastedvalues$mean)

dis\_predictedval <- exp(dis\_value)

dis\_actual\_value <- exp(dis\_test)

**#calculating the error between predicted and test data**

dis\_error <- mean(abs((dis\_actual\_value - dis\_predictedval)/dis\_actual\_value))

dis\_rmse <- rmse(dis\_actual\_value,dis\_predictedval)

**#--------------------------------------------Amazon data-------------------------------------------**

**#Subsetting the stock data based on tickers**

amazon\_data <- subset(stockdata, ticker=="AMZN", select = c(ref.date,price.close))

amz\_price <- amazon\_data$price.close

amz\_ts <- ts(amz\_price, start = decimal\_date(ymd("2009-04-13")), frequency = 365.25/7)

**#Plotting the data sets**

plot(amz\_ts, main = "Amazon Stock Closing price vs Time", ylab = "Closing price" )

**#convert to ln format**

amz\_lnprice <- log(amz\_price)

**#Moving average on ln of stock price**

amz\_difflnprice <- diff(amz\_lnprice,1)

**#Dickey-Fuller Test**

adf.test(amz\_price)

adf.test(amz\_lnprice)

adf.test(amz\_difflnprice)

**#ACF, PACF**

acf(amz\_lnprice, lag.max=50, main="ACF plot of Amazon stock")

pacf(amz\_lnprice, lag.max=50, main="PACF plot of Amazon stock")

**#breaking the dataset into training and testing**

breakpoint <- floor(dim(amazon\_data)[1]\*0.8)

**#Divide into training and test data**

amz\_train <- amz\_lnprice[1:breakpoint]

amz\_test <- amz\_lnprice[breakpoint:dim(amazon\_data)[1]]

**#Time series and auto.arima on ln price**

amz\_pricearima <- ts(amz\_train, start = decimal\_date(ymd("2009-04-13")), frequency = 365.25/7)

amz\_fit <- auto.arima(amz\_pricearima, allowdrift = TRUE, trace = TRUE, test = "adf", ic = "bic")

summary(amz\_fit)

**#Forecasted values from ARIMA**

amz\_forecastedvalues <- forecast(amz\_fit, h=106)

plot(amz\_forecastedvalues, include= 100, xlab = "Time",ylab = "log(closing price)")

**#Converting the log values to exponential**

amz\_value <- as.numeric(amz\_forecastedvalues$mean)

amz\_predictedval <- exp(amz\_value)

amz\_actual\_value <- exp(amz\_test)

**#calculating the error between predicted and test data**

amz\_error <- mean(abs((amz\_actual\_value - amz\_predictedval)/amz\_actual\_value))

amz\_rmse <- rmse(amz\_actual\_value,amz\_predictedval)