OR538 Project Report

Report for project for *Fall 2016*
as part of OR538 Analytics for Financial Engineering and Econometrics
Data Analytics Engineering Program
Department of Systems Engineering and Operations Research (SEOR)
Volgenau School of Engineering
George Mason University

Instructor: Prof: Kuo Chu Chang

By
Ashok Vardhan Kari
Sudhamini Guda
Vamshidar Reddy Chaulapally

TABLE OF CONTENTS

1. Introduction
2. Purpose of the project
3. Portfolio Selection
4. Approach
5. Extracting Trends
6. Model Building
   - 6.1 Holt-Winters
   - 6.2 Nueral Networks
   - 6.3 Arima
   - 6.4 Time-Series Linear Model
   - 6.5 STL (Seasonal Decomposition of Time Series by Lowess)
7. Predictions based on models
   - 7.1 Predicitons on Actual Trend
   - 7.2 Predicitons on Polynomial Trend
   - 7.3 Predictions on STL
8. Choosing the accuracy metric
9. Performance on Amazon stock example
10. Learning Achieved
11. References
12. Appendix

1. Introduction:

Stock market prediction is trying to determine the future price of a stock or a future or other derivative traded on an exchange. While a successful prediction of stock can yield significant profits, we have a Random Walk Theory which claims that stock prices cannot be predicted using the historical prices. It is however observed that against these fallacies, trading companies like JP Morgan, Morgan Stanley and Goldman Sachs have consistently made profits based on investment predictions.

2. Purpose of the project:

The purpose of this project is to develop a useful prediction system in forecasting stock prices. We have approached this purpose by developing a model that is suitable to predict for any type of stock and any number of stocks at a time.

3. Portfolio:

Since the project aims to make prediction for any kind of stocks in a portfolio, much importance is not given to the selection of stocks or the correlation between them. We randomly chose ten stocks from our main source Yahoo finance. For the information purpose the list is as shown below.

1. A
2. F
3. ABC
4. INTC
5. KO
6. CSCO
7. XOM
8. COF
9. HPQ
10. AMZN

Our training data consists of the stock prices from January 2000 to December 2013 and our test data is from January 2014 to November 2016. We have made a brief prediction on stock prices from December 2016 to December 2019.

4. Approach:

We have used R extensively for our project. The important packages we have used are "forecast" and "fpp"

5. Extracting Trends:

After converting into time-series data, we extract the three constituent trends in the data namely,

1. Random Trend
2. Polynomial Trend
3. Seasonal Trend

Here random trend is nothing but the actual trend if the stock for the given period of time. The polynomial trend is the type of trend that represents large data with lots of fluctuations. Each year market tends to repeat certain seasonal trends. These seasonal trends affect individual stocks and the stock market as a whole. When an investor have a thorough understanding of these trends work they are able to gain a slight advantage when it comes to trading and investing. Hence it is important to extract the seasonal trend to bring normality in the data.

Given a data set with the closing prices of stock, we can extract different kinds of trend by fitting polynomials. We have extracted a polynomial trend by fitting all the stock's closing data with a polynomial of seventh degree. The second trend extracted is STL trend, which is by applying STL to dataset and taking only trend data points and avoiding seasonality and remainder data points.

6. Model Building:

We built 7 models and applied them to different trends. Following are those models:

6.1 Holtwinters (2 Models):

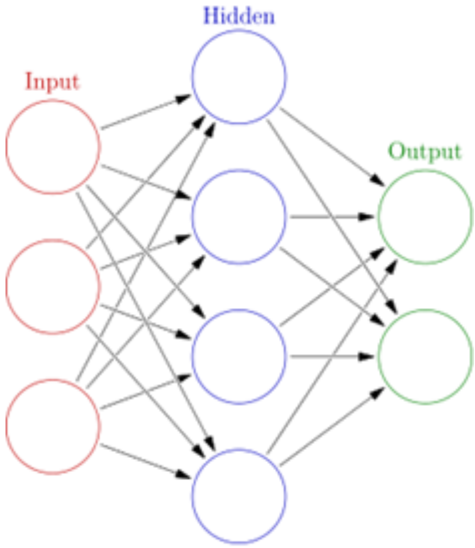
Holt (1957) and Winters (1960) extended Holt's method to capture seasonality. The Holt-Winters seasonal method comprises the forecast equation and three smoothing equations — one for the level l_t, one for trend bt, and one for the seasonal component denoted by st, with smoothing parameters α, β* and γ. We use mm to denote the period of the seasonality, i.e., the number of seasons in a year. For example, for quarterly data m=4, and for monthly data m=12.

There are two variations to this method that differ in the nature of the seasonal component. The additive method is preferred when the seasonal variations are roughly constant through the series, while the multiplicative method is preferred when the seasonal variations are changing proportional to the level of the series. With the additive method, the seasonal component is expressed in absolute terms in the scale of the observed series, and in the level equation the series is seasonally adjusted by subtracting the seasonal component. Within each year the seasonal component will add up to approximately zero. With the multiplicative method, the seasonal component is expressed in relative terms (percentages) and the series is seasonally adjusted by dividing through by the seasonal component. Within each year, the seasonal component will sum up to approximately m.

We have also built another holtwinter model without considering seasonality by keeping gamma=FALSE (considering Simple Exponential Smoothing).

6.2 Neural Network (1 Model):

Neural Networks are a machine learning framework that attempts to mimic the learning pattern of natural biological neural networks. Biological neural networks have interconnected neurons with dendrites that receive inputs, then based on these inputs they produce an output signal through an axon to another neuron.



6.3 ARIMA (2 Models):

The AR part of ARIMA indicates that the evolving variable of interest is regressed on its own lagged (i.e., prior) values. The MA part indicates that the regression error is actually a linear combination of error terms whose values occurred contemporaneously and at various times in the past. The I (for "integrated") indicates that the data values have been replaced with the difference between their values and the previous values (and this differencing process may have been performed more than once). The purpose of each of these features is to make the model fit the data as well as possible.
A nonseasonal ARIMA model is classified as an "ARIMA(p,d,q)" model, where:

- **p** is the number of autoregressive terms,
- **d** is the number of nonseasonal differences needed for stationarity, and
- **q** is the number of lagged forecast errors in the prediction equation.

We have built two ARIMA models, ARIMA(15,3,3) and auto arima.

6.4 Time Series Linear Model (1 Model):

Time series linear model(tslm) is used to fit linear models to time series including trend and seasonality components. tslm is largely a wrapper for lm() except that it allows variables "trend" and "season" which are created on the fly from the time series characteristics of the data. The variable "trend" is a simple time trend and "season" is a factor indicating the season (e.g., the month or the quarter depending on the frequency of the data).

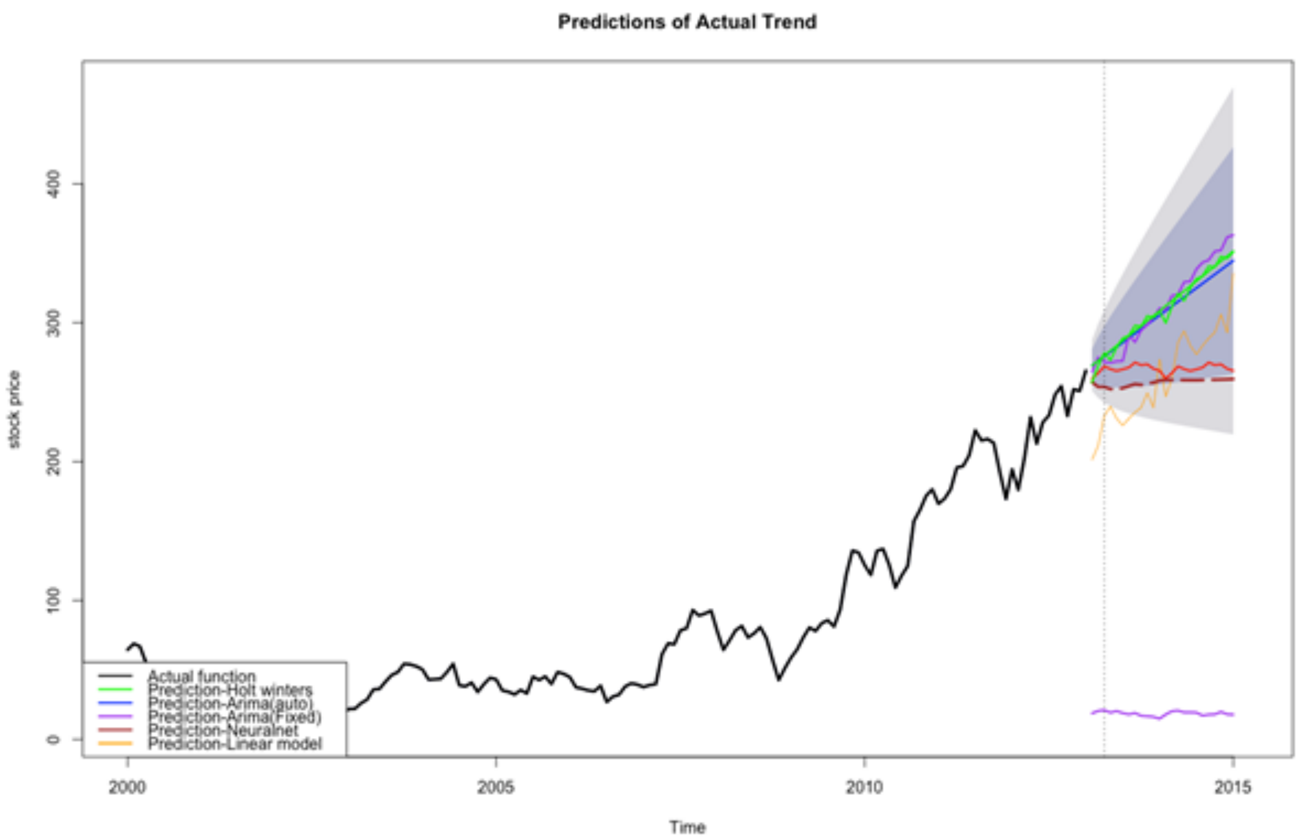
6.5 STL (Seasonal Decomposition of Time Series by Loess) (1 Model):

The Seasonal Trend Decomposition using Loess (STL) is an algorithm that was developed to help to divide up a time series into three components namely: the trend, seasonality and remainder. The methodology was presented by Robert Cleveland, William Cleveland, Jean McRae and Irma Terpenning in the Journal of Official Statistics in 1990. The STL is available within R via the **stl** function.

The seasonal component is found by *loess* smoothing the seasonal subseries; if s.window = "periodic" smoothing is effectively replaced by taking the mean. The seasonal values are removed, and the remainder smoothed to find the trend. The overall level is removed from the seasonal component and added to the trend component. This process is iterated a few times. The remainder component is the residuals from the seasonal plus trend fit.

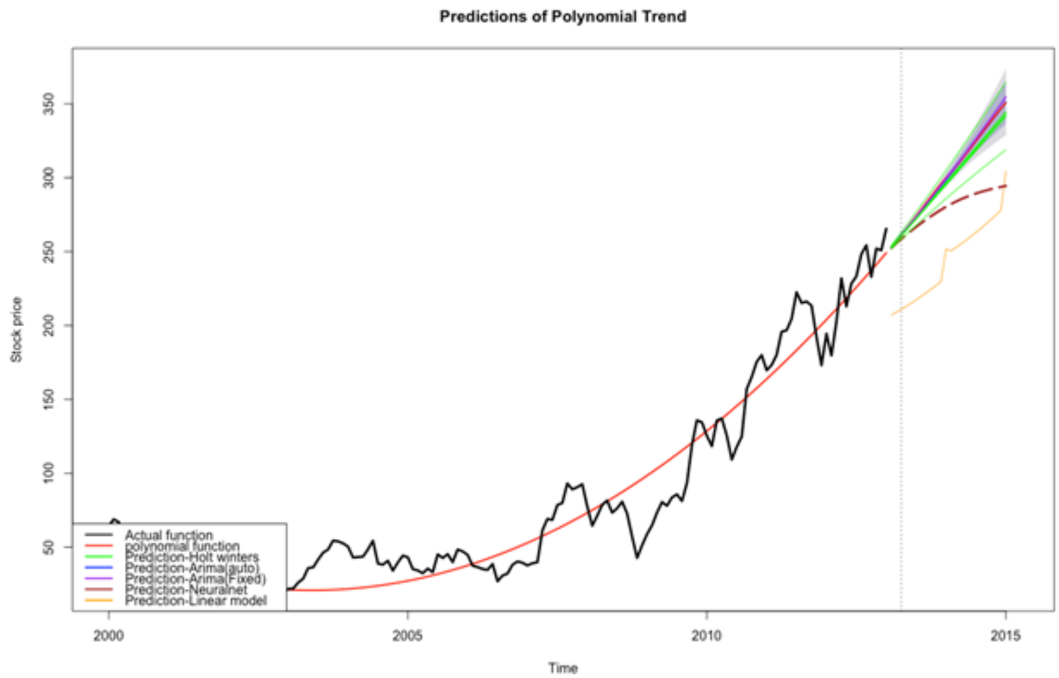
7. Predictions based on Models:

7.1 Predictions on Actual Trend:



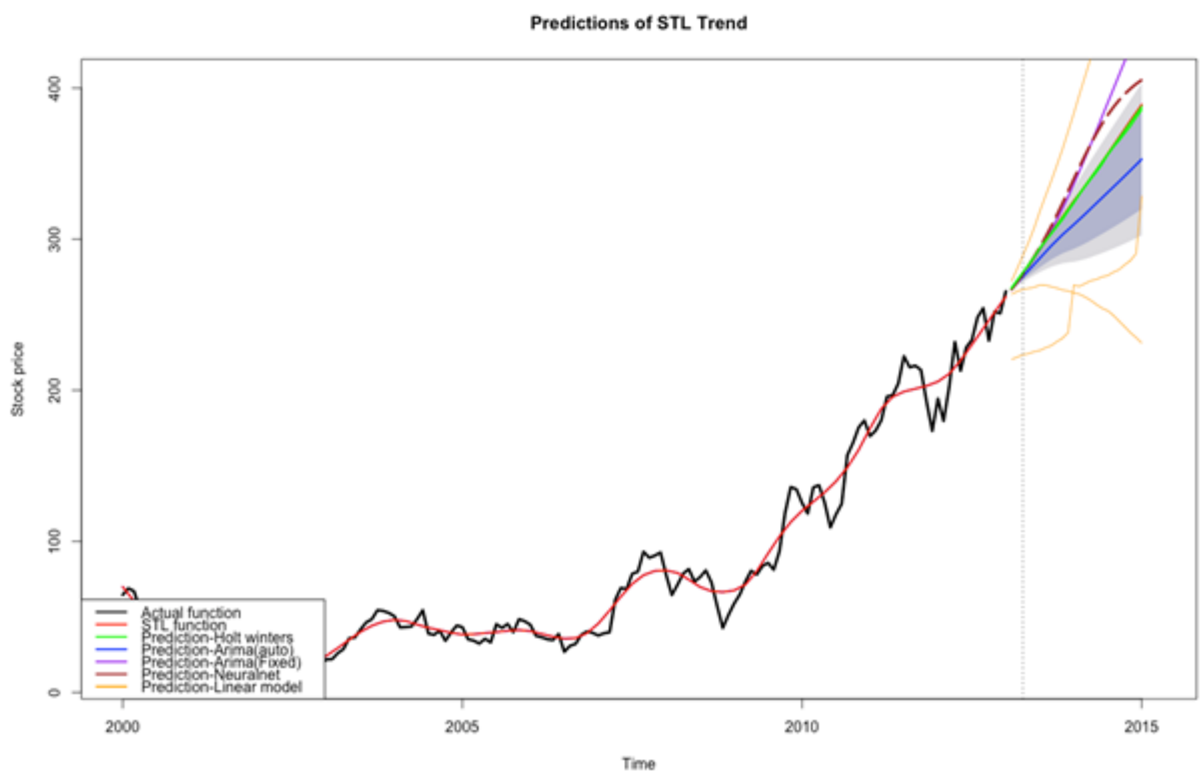
The above generated graph plots the predictions of all the seven models on actual amazon stock data. These predictions are performed using forecast() function for a span of 2 years (2013-2015).

7.2 Prediction of polynomial Trend:



The above generated graph plots the predictions of all seven models on the polynomial trend data which is generated by fitting seventh degree polynomial into actual data. . These predictions are performed using forecast() function for a span of 2 years (2013-2015).

7.3 Prediction of STL Trend:



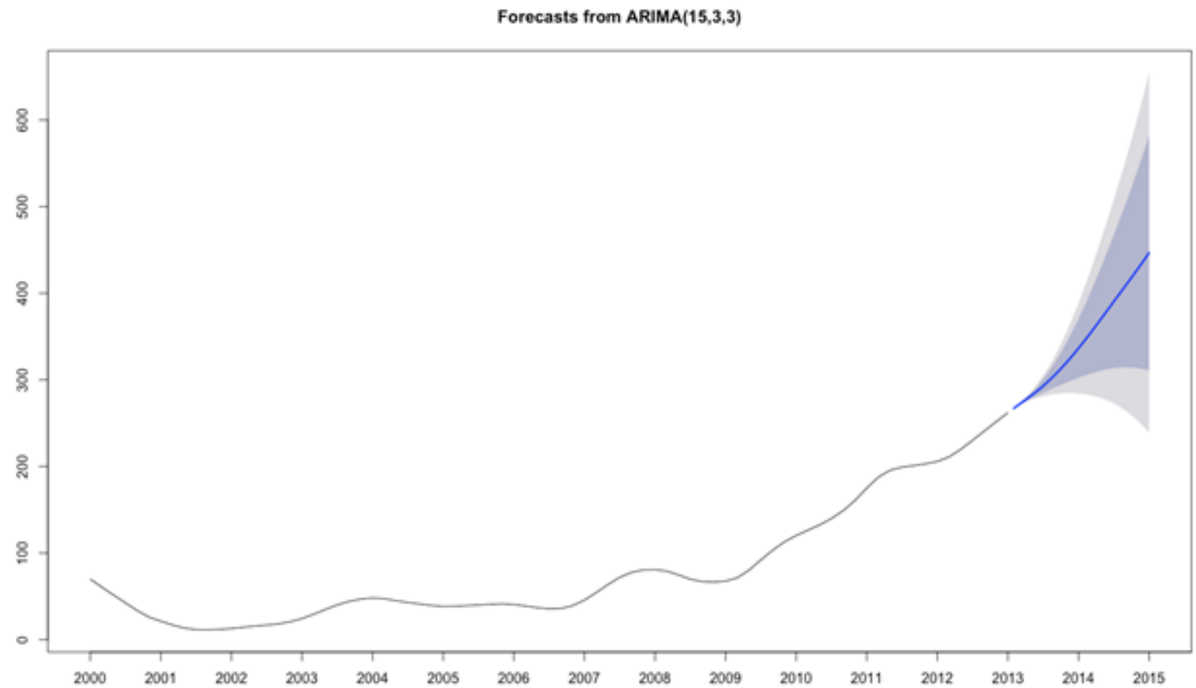
The above generated graph is the predictions of all the seven models on the STL trend values (excluding seasonality and remainder) which we have taken by applying stl() function on actual amazon stock data. These predictions are performed using forecast() function for a span of 2 years (2013-2015).

8. Accuracy Metrics:

We have built 21 models (7 on each of 3 trend types) and which will produce 21 MAE values and choose least value and gives out corresponding model number. Then we used that model on that particular stock and predicted next three years of monthly prices.

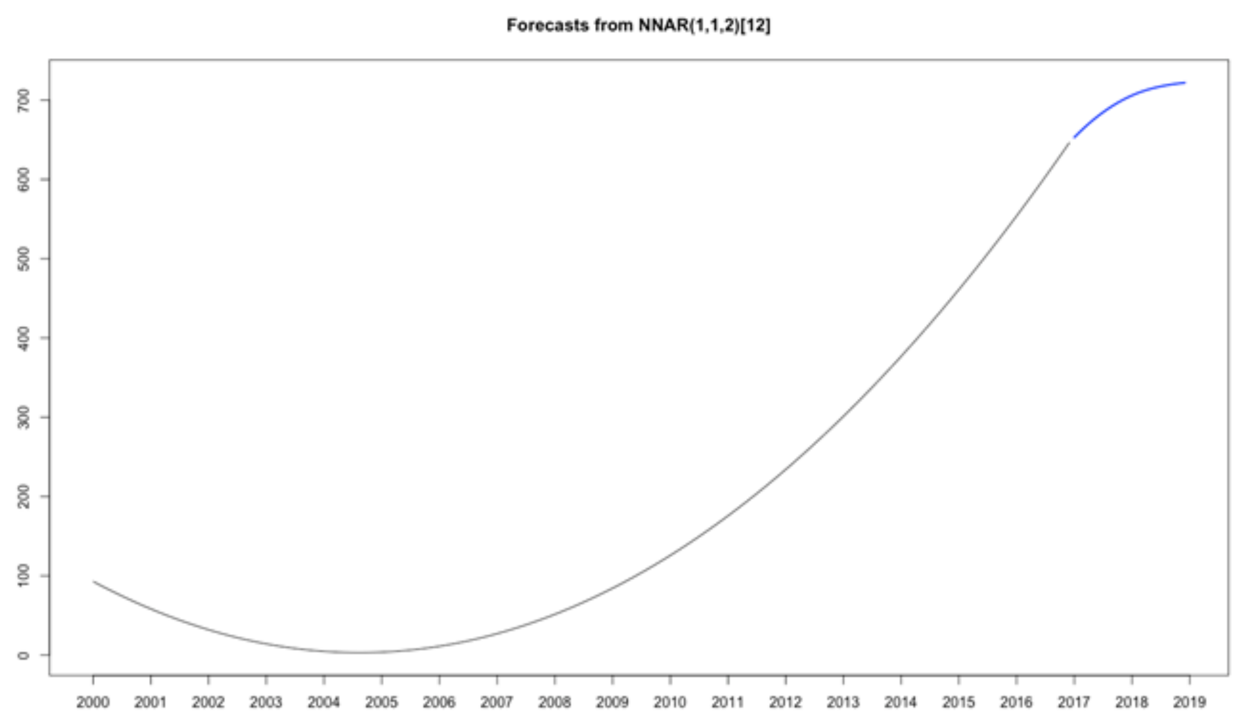
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
[1,]	295.26775	297.27778	295.26775	38.711631	38.711631	5.1604254	0.3474166	7.832836
[2,]	499.96301	501.30862	499.96301	65.659668	65.659668	198.1284343	0.4246262	13.221596
[3,]	339.95886	341.57544	339.95886	44.604919	44.604919	7.9510316	0.3653700	8.983592
[4,]	339.20574	340.98355	339.20574	44.493440	44.493440	7.8899353	0.3672809	8.991489
[5,]	335.88731	337.68015	335.88731	44.056575	44.056575	7.6366215	0.3661065	8.904247
[6,]	305.87930	307.81969	305.87930	40.110571	40.110571	5.4545793	0.3462986	8.107718
[7,]	74.32781	83.90427	74.32781	9.637662	9.637662	0.6502585	0.3445419	2.195470
[8,]	327.53924	329.35038	327.53924	42.960772	42.960772	6.4543238	0.3560782	8.679418
[9,]	268.66184	270.86418	268.66184	35.206910	35.206910	4.2301100	0.3448809	7.139681
[10,]	492.55055	493.79920	492.55055	64.692873	64.692873	Inf	0.4189462	13.023614
[11,]	328.19945	330.23987	328.19945	43.025109	43.025109	6.0965965	0.4234732	8.741390
[12,]	274.16020	276.30302	274.16020	35.930373	35.930373	4.4329088	0.3456660	7.282997
[13,]	270.55022	272.74236	270.55022	35.454509	35.454509	4.3024748	0.3471349	7.190553
[14,]	323.18860	325.03418	323.18860	42.387010	42.387010	6.1956729	0.3563039	8.566384
[15,]	328.38570	330.74214	328.38570	43.042544	43.042544	5.4109773	0.3769882	8.803190
[16,]	375.48752	377.11810	375.48752	49.270505	49.270505	7.1351960	0.3802497	9.945861
[17,]	347.71351	349.60390	347.71351	45.603336	45.603336	5.9902381	0.3827951	9.229276
[18,]	434.50766	436.03497	434.50766	57.035612	57.035612	19.2453876	0.4194357	11.509630
[19,]	339.29436	341.06198	339.29436	44.507026	44.507026	7.1461774	0.3622403	8.990154
[20,]	343.56778	345.31582	343.56778	45.069674	45.069674	7.4201097	0.3635361	9.102389
[21,]	292.50083	294.52922	292.50083	38.348025	38.348025	4.8742245	0.3453961	7.759105
[22,]	394.63560	396.20739	394.63560	51.791436	51.791436	12.1073235	0.3862482	10.448751

9. Forecasting Amazon on Train Data (2013-2015):



10. Predicting Amazon monthly stock till 2019:

Here we choose the best model i.e. ARIMA(15,3,3) using polynomial trend and forecasted for next three years (2017-2019) for amazon stock.



Learning Achieved:

In doing this project we have learned how to deal with time series data i.e. checking stationarity and reading ACF(Auto Correlation Factor) charts. Learned different types of model applications like ARIMA, NeuralNet, Holt-winters, and STL.Normalizing data is very important before applying Neural net. Depending on your dataset, avoiding normalization may lead to useless results or to a very difficult training process (most of the times the algorithm will not converge before the number of maximum iterations allowed).

Results:

Stocks	**Best Trend**	**Best Model**
A	Actual Trend	Neural Net
F	STL Trend	Neural Net
ABC	Actual Trend	ARIMA(15,3,3)
INTC	Polynomial Trend	Holtwinters
KO	Actual Trend	STL
CSCO	Polynomial Trend	Holtwinters
XOM	Polynomial Trend	Time series Linear Model
COF	STL Trend	STL
HPQ	Polynomial Trend	Time series Linear Model
AMZN	STL Trend	ARIMA(15,3,3)

REFERENCES:

1. Book: Statistics and Data Analysis for Financial Engineering 2nd Edition by David Ruppert and David S. Matteson
2. Data: Yahoo Finance www.finance.yahoo.com

3. Blog: https://www.r-bloggers.com/fitting-a-neural-network-in-r-neuralnet-package/
4. Article: http://quantlego.com/howto/holt-winters-smoothing-and-forecast/
5. Article: http://finance.zacks.com/seasonal-stock-market-trends-5830.html
6. Paper: Stock Market Regression Using Regression and Polynomial Models http://lnunno.github.io/assets/docs/ml_paper.pdf

APPENDIX:

R Codes

```
library(forecast)
library(fpp)
Portfolio = list()
BestPrediction = vector()
Portfolio[[1]]=read.csv("http://ichart.finance.yahoo.com/table.csv?s=A&a=00&b=1&c=2000&d=03&e=
1&f=2017&g=m&ignore=.csv", stringsAsFactors = FALSE)
Portfolio[[2]]=read.csv("http://ichart.finance.yahoo.com/table.csv?s=F&a=00&b=1&c=2000&d=03&e=
1&f=2017&g=m&ignore=.csv", stringsAsFactors = FALSE)
Portfolio[[3]]=read.csv("http://ichart.finance.yahoo.com/table.csv?s=ABC&a=00&b=1&c=2000&d=03
&e=1&f=2017&g=m&ignore=.csv", stringsAsFactors = FALSE)
Portfolio[[4]]=read.csv("http://ichart.finance.yahoo.com/table.csv?s=INTC&a=00&b=1&c=2000&d=03
&e=1&f=2017&g=m&ignore=.csv", stringsAsFactors = FALSE)
Portfolio[[5]]=read.csv("http://ichart.finance.yahoo.com/table.csv?s=KO&a=00&b=1&c=2000&d=03&
e=1&f=2017&g=m&ignore=.csv", stringsAsFactors = FALSE)
Portfolio[[6]]=read.csv("http://ichart.finance.yahoo.com/table.csv?s=CSCO&a=00&b=1&c=2000&d=0
3&e=1&f=2017&g=m&ignore=.csv", stringsAsFactors = FALSE)
Portfolio[[7]]=read.csv("http://ichart.finance.yahoo.com/table.csv?s=XOM&a=00&b=1&c=2000&d=03
&e=1&f=2017&g=m&ignore=.csv", stringsAsFactors = FALSE)
Portfolio[[8]]=read.csv("http://ichart.finance.yahoo.com/table.csv?s=COF&a=00&b=1&c=2000&d=03
&e=1&f=2017&g=m&ignore=.csv", stringsAsFactors = FALSE)
Portfolio[[9]]=read.csv("http://ichart.finance.yahoo.com/table.csv?s=HPQ&a=00&b=1&c=2000&d=03
&e=1&f=2017&g=m&ignore=.csv", stringsAsFactors = FALSE)
Portfolio[[10]]=read.csv("http://ichart.finance.yahoo.com/table.csv?s=AMZN&a=00&b=1&c=2000&d=
03&e=1&f=2017&g=m&ignore=.csv", stringsAsFactors = FALSE)

x<-
c("fitr","NETfitr","HWStockr","HWStockr_ng","autofit2","fit12","fit2","stlStock1","stlStock2","stlStoc
kr","NETfit2","HWStock2","HWStock2_ng","autofit1","fitlr","fitl1","fitl2","HWStock1_ng","NETfit1",
"HWStock1","HWStock1","HWStock1")
y <- 1:22

models <- data.frame(y,x)
```

```
 findBestPrediction <- function(Stock)
 {
   # Convert into timeseries object
   tsStock=ts(rev(Stock$Close),start = c(2000,1),frequency = 12)

   # Create train and Test sets of the input stocks
   train <- window(tsStock, end=2013)
   test <- window(tsStock, start=2014)

   #Generalize function as polynomial trend  (TREND="tsStocktrend1)
   tl = seq(2000,2013,length=length(train))
   tl2 = tl^7
   polyStock = lm(train ~ tl + tl2)
   tsStocktrend1=ts(polyStock$fit,start=c(2000, 1),frequency=12)
   plot(train,lw=2,col="blue",xlim=c(2000,2013))
   lines(tsStocktrend1,lw=2,col="red")
   abline(v=2013.25,lty=3)

   #Decompose a timeseries into seasonal, trend and irregular components
   #Second generalised trend function (tsStocktrend2)
   stlStock = stl(train,s.window="periodic")
   plot(stlStock,col="blue",lwd=2)
   tsStocktrend2 = stlStock$time.series[,2]
   plot(forecast(stlStock))
   abline(v=2013.25,lty=3)
   #
   plot(train,lwd=3)
   lines(tsStocktrend1,col="purple",lwd=2)
   lines(tsStocktrend2,col="red",lwd=2)
   abline(v=2013.25,lty=3)
legend("topright",legend=c("Actual","polynomialtrend","STLTrend"),col=c("black","purple","red"),lwd=
2)

   #Start Predicting
   # Based on Polynomial function
   HWStock1_ng = HoltWinters(tsStocktrend1,gamma=FALSE)
   HWStock1 = HoltWinters(tsStocktrend1)
   NETfit1 <- nnetar(tsStocktrend1)
   autofit1 = auto.arima(tsStocktrend1)
   fit12 <- Arima(tsStocktrend1, order=c(1,0,0), list(order=c(2,1,0), period=12), method="CSS")
   fitl1 <- tslm(tsStocktrend1 ~ trend + season, lambda=0)
   stlStock1 = stl(tsStocktrend1,s.window="periodic")

plot(forecast(autofit1,h=24),xlim=c(2000,2015.2),lwd=2,col="red",xlab="Time",ylab="Stock
price",main="Predictions of Polynomial Trend")
```

```
  lines(forecast(stlStock1,h=24)$mean,col="red",lw=2)
  lines(train,lw=3)
  lines(forecast(fitl1,h=24)$mean,col="orange")
  lines(forecast(NETfit1,h=24)$mean,lw=3,lty="longdash",col="brown")
  lines(predict(HWStock1_ng,n.ahead=24),lw=2,col="green")
  lines(forecast(fit12,h=24)$mean,lw=2,col="purple")
  lines(predict(HWStock1,n.ahead = 24,prediction.interval = T,level=0.95)[,1],lw=2,col="green")
  lines(predict(HWStock1,n.ahead = 24,prediction.interval = T,level=0.95)[,2],col="green")
  lines(predict(HWStock1,n.ahead = 24,prediction.interval = T,level=0.95)[,3],col="green")
  legend("bottomleft",legend=c("Actual function","polynomial function","Prediction-Holt
winters","Prediction-Arima(auto)","Prediction-Arima(Fixed)","Prediction-Neuralnet","Prediction-Linear
model"),col=c("black","red","green","blue","purple","brown","orange"),lw=2)
  abline(v=2013.25,lty=3)

  #Based on STL Function
  HWStock2_ng = HoltWinters(tsStocktrend2,gamma=FALSE)
  HWStock2 = HoltWinters(tsStocktrend2)
  NETfit2 <- nnetar(tsStocktrend2)
  autofit2 = auto.arima(tsStocktrend2)
  fit2 <- Arima(tsStocktrend2, order=c(15,3,3),method="CSS")
  fitl2 <- tslm(tsStocktrend2 ~ trend + season, lambda=0)
  #fit22=arima(tsStocktrend2,order = c(1,0,0),list(order=c(2,1,0),period=12))
  stlStock2 = stl(tsStocktrend2,s.window="periodic")

plot(forecast(autofit2,h=24),xlim=c(2000,2015.2),lwd=2,col="blue",xlab="Time",ylab="Stock
price",main="Predictions of STL Trend")
  lines(train,lw=3)
  lines(forecast(stlStock2,h=24)$mean,col="red",lw=2)
  lines(forecast(fitl2,h=24)$mean,col="orange")
  lines(forecast(fit2,h=24)$mean,lw=2,col="purple")
  #lines(forecast(fit22,h=24)$mean,lw=2,col="purple")
  lines(tsStocktrend2,lw=2,col="red")
  lines(forecast(NETfit2,h=24)$mean,lw=3,lty="longdash",col="brown")
  lines(predict(HWStock2,n.ahead = 24),lw=2,col="green")
  lines(predict(HWStock2_ng,n.ahead=24),lw=2,col="green")

  lines(predict(HWStock2,n.ahead = 24,prediction.interval = T,level=0.95)[,2],col="orange")
  lines(predict(HWStock2,n.ahead = 24,prediction.interval = T,level=0.95)[,3],col="orange")
legend("bottomleft",legend=c("Actualfunction","STLfunction","Prediction-Holt winters","Prediction-
Arima(auto)","Prediction-Arima(Fixed)","Prediction-Neuralnet","Prediction-Linear
model"),col=c("black","red","green","blue","purple","brown","orange"),lw=2)
  abline(v=2013.25,lty=3)

  #Based on actual function
```

```
  HWStockr_ng=HoltWinters(train,gamma = FALSE)
  HWStockr=HoltWinters(train)
  NETfitr=nnetar(train)
  autofitr=auto.arima(train)
  fitr=arima(train,order=c(15,3,3), method = "CSS")
  #fitr2=arima(train,order=c(1,0,0),list(order=c(2,1,0),period=12))
  fitlr=tslm(train~trend+season,lambda = 0)
  stlStockr=stl(train,s.window = "periodic")

plot(forecast(autofitr,h=24),xlim=c(2000,2015.2),lw=2,col="blue",xlab="Time",ylab="stock
price",main="Predictions of Actual Trend")
  lines(forecast(fitlr, h=24)$mean, col="orange")
  lines(forecast(stlStockr, h=24)$mean, col="red", lw=2)
  lines(forecast(fitr,h=24)$mean,lw=2,col="purple")
  lines(forecast(fitr2,h=24)$mean,lw=2,col="purple")
  lines(train,lw=3)
  lines(forecast(NETfitr,h=24)$mean,lw=3,lty="longdash",col="brown")
  lines(predict(HWStockr,n.ahead=24),lw=2,col="green")
  lines(predict(HWStockr_ng,n.ahead=24),lw=2,col="green")
  abline(v=2013.25,lty=3)
legend("bottomleft",legend=c("Actualfunction","Prediction-Holt                    winters","Prediction-
Arima(auto)","Prediction-Arima(Fixed)","Prediction-Neuralnet","Prediction-Linear
model"),col=c("black","green","blue","purple","brown","orange"),lw=2)

  predfitr =  window(forecast(fitr,h=39)$mean)
  # cat("4")
  #predfitr2  =   window(forecast(fitr2,h=39)$mean, start=2014)
  # cat("5")
  predNETfitr =  window(forecast(NETfitr,h=39)$mean)
  # cat("6")
  predHWStockr =  window(predict(HWStockr,n.ahead=39))
  # cat("7")
  predHWStockr_ng =  window(predict(HWStockr_ng,n.ahead=39))
  # cat("8")
  predautofit2 =   window(forecast(autofit2,h=39)$mean)
  # cat("9")
  predfit12 =  window(forecast(fit12,h=39)$mean)
  # cat("10")
  predfit2 = window(forecast(fit2,h=39)$mean)
  # cat("11")
  #predfit22 =  window(forecast(fit22,h=39)$mean, start=2014)
  # cat("12")
  predstlStock1 = window(forecast(stlStock1, h=39)$mean)
  # cat("13")
```

```
predstlStock2 =  window(forecast(stlStock2, h=39)$mean)
# cat("14")
predstlStockr =  window(forecast(stlStockr, h=39)$mean)
# cat("15")
predNETfit2 =  window(forecast(NETfit2,h=39)$mean)
predHWStock2 =  window(predict(HWStock2,n.ahead=39))
predHWStock2_ng =  window(predict(HWStock2_ng,n.ahead=39))
predautofit1 =  window(forecast(autofit1,h=39)$mean)
# cat("after autofit")
predfitlr = window(forecast(fitlr, h=39)$mean)
predfitl1 =   window(forecast(fitl1, h=39)$mean)
predfitl2 = window(forecast(fitl2, h=39)$mean)
predNETfit1 =  window(forecast(NETfit1,h=39)$mean)
predHWStock1_ng =  window(predict(HWStock1_ng,n.ahead=39))
predHWStock11 =  window(predict(HWStock1, n.ahead = 39, prediction.interval = T, level = 0.95)[,1])
predHWStock12 =  window(predict(HWStock1, n.ahead = 39, prediction.interval = T, level = 0.95)[,2])
predHWStock13 =  window(predict(HWStock1, n.ahead = 39, prediction.interval = T, level = 0.95)[,3])

forecasts <- lapply(list(predfitr, predNETfitr, predHWStockr, predHWStockr_ng, predautofit2, predfit12, predfit2, predstlStock1, predstlStock2, predstlStockr, predNETfit2, predHWStock2, predHWStock2_ng, predautofit1, predfitlr , predfitl1, predfitl2, predNETfit1,  predHWStock1_ng, predHWStock11, predHWStock12, predHWStock13), forecast, 12)
acc <- lapply(forecasts, function(f){
  accuracy(f, test)[2,,drop=FALSE]
})
acc <- Reduce(rbind, acc)
row.names(acc) <- names(forecasts)
#acc <- acc[order(acc[,'MASE']),]
round(acc, 2)

# Mean Absolute Errors of the 25 predictions are stored here
mae = matrix(NA, 25, length(test)+1)

# Calculate MAE
for(i in 1:length(test))
{
  mae[1,i] <- abs(predfitr[i]-test[i])
  mae[2,i] <- abs(predNETfitr[i]-test[i])
  mae[3,i] <- abs(predHWStockr[i]-test[i])
  mae[4,i] <- abs(predHWStockr_ng[i]-test[i])
  mae[5,i] <- abs(predautofit2[i]-test[i])
  mae[6,i] <- abs(predfit12[i]-test[i])
  mae[7,i] <-abs(predfit2[i]-test[i])
  mae[8,i] <- abs(predstlStock1[i]-test[i])
```

```
    mae[9,i] <- abs(predstlStock2[i]-test[i])
    mae[10,i] <- abs(predstlStockr[i]-test[i])
    mae[11,i] <- abs(predNETfit2[i]-test[i])
    mae[12,i] <- abs(predHWStock2[i]-test[i])
    mae[13,i] <- abs(predHWStock2_ng[i]-test[i])
    mae[14,i] <- abs(predautofit1[i]-test[i])
    mae[15,i] <- abs(predfitlr[i]-test[i])
    mae[16,i] <-  abs(predfitl1[i]-test[i])
    mae[17,i] <- abs(predfitl2[i]-test[i])
    mae[18,i] <- abs(predNETfit1[i]-test[i])
    mae[19,i] <- abs(predHWStock1_ng[i]-test[i] )
    mae[20,i] <- abs(predHWStock11[i]-test[i])
    mae[21,i] <- abs(predHWStock12[i]-test[i])
    mae[22,i] <- abs(predHWStock13[i]-test[i])
  }

  # Sum all Errors
  for(i in 1:22)
  {
    mae[i,36] = mean(mae[i,1:35])
  }
  # Find best Prediction
  best = which.min(mae[1:22,36])
  cat(" - winning model ID:", best, "\n")
  for(i in 1:22)
  {
    if (best == models[i,1])
    {
      fact = forecast(eval(as.name(paste(models[i,2]))))
      plot(fact,axes=FALSE)
      axis(side=1,at=seq(2000,2019,by=1))
      axis(side=2)
      box()
    }else{}
  }
  print(acc)
  return (best)
}

for (i in 1:length(Portfolio))
{
  BestPrediction[i] = findBestPrediction(Portfolio[[i]])
}
```