STOCK PRICE PREDICTION (NETFLIX)

GROUP MEMBERS

Madhur Nagaraj

Akhil Patil

Vipul Pawar

Pranab Singh

BACKGROUND

Netflix, Inc. is an American media-services provider headquartered in Los Gatos, California, founded in 1997 by Reed Hastings and Marc Randolph in Scotts Valley, California. The company's primary business is its subscription-based streaming OTT service which offers online streaming of a library of films and television programs, including those produced in-house.

AIM

By performing closing price analysis for investors and traders we can help them make buying and selling decisions. We study and evaluate past data and prognosticate the future closing price. By doing so, investors and traders can gain an edge in the markets by making informed decisions.

Importing the Libraries

```
library(ggplot2)
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(data.table)
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

library(tidyr)
library(corrplot)

## corrplot 0.84 loaded

library(forecast)
library(tseries)
library(TSA)

##
## Attaching package: 'TSA'

## The following objects are masked from 'package:stats':
##
##     acf, arima

## The following object is masked from 'package:utils':
##
##     tar

library(tibble)
library(TTR)

s_data <- read.csv("D:/Downloads/all_stocks_5yr.csv")
attach(s_data)
s_data = na.omit(s_data)
summary(s_data)

##          date              open               high               low
##  2017-12-05:   505   Min.   :   1.62   Min.   :   1.69   Min.   :   1.50
##  2017-12-06:   505   1st Qu.:  40.22   1st Qu.:  40.62   1st Qu.:  39.83
##  2017-12-07:   505   Median :  62.59   Median :  63.15   Median :  62.02
##  2017-12-08:   505   Mean   :  83.02   Mean   :  83.78   Mean   :  82.26
##  2017-12-11:   505   3rd Qu.:  94.37   3rd Qu.:  95.18   3rd Qu.:  93.54
##  2017-12-12:   505   Max.   :2044.00   Max.   :2067.99   Max.   :2035.11
##  (Other)   :615999
##      close              volume                Name
##  Min.   :   1.59   Min.   :       101   A       :  1259
##  1st Qu.:  40.24   1st Qu.:   1070351   AAL     :  1259
##  Median :  62.62   Median :   2082165   AAP     :  1259
##  Mean   :  83.04   Mean   :   4321892   AAPL    :  1259
##  3rd Qu.:  94.41   3rd Qu.:   4284550   ABBV    :  1259
##  Max.   :2049.00   Max.   :618237630   ABC     :  1259
##                                          (Other):611475
```

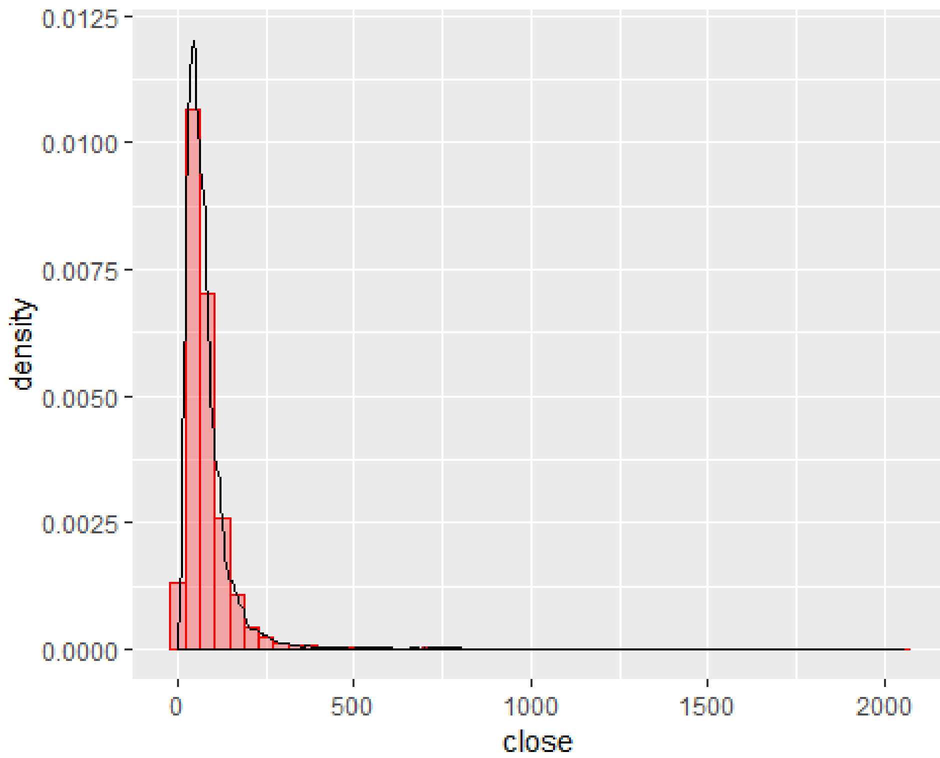
Data Cleaning

Here we remove the NA values and stock data of other companies thereby reducing the data to the stock prices of the NFLX company. We also perform necessary conversion of dates into specific date formats which will ease application of models.

```
s_data[is.na(s_data)] <- 0
summary(s_data)

##          date             open              high              low
##  2017-12-05:   505   Min.   :   1.62   Min.   :   1.69   Min.   :   1.50
##  2017-12-06:   505   1st Qu.:  40.22   1st Qu.:  40.62   1st Qu.:  39.83
##  2017-12-07:   505   Median :  62.59   Median :  63.15   Median :  62.02
##  2017-12-08:   505   Mean   :  83.02   Mean   :  83.78   Mean   :  82.26
##  2017-12-11:   505   3rd Qu.:  94.37   3rd Qu.:  95.18   3rd Qu.:  93.54
##  2017-12-12:   505   Max.   :2044.00   Max.   :2067.99   Max.   :2035.11
##  (Other)   :615999
##      close             volume               Name
##  Min.   :   1.59   Min.   :       101   A      :  1259
##  1st Qu.:  40.24   1st Qu.:  1070351   AAL    :  1259
##  Median :  62.62   Median :  2082165   AAP    :  1259
##  Mean   :  83.04   Mean   :  4321892   AAPL   :  1259
##  3rd Qu.:  94.41   3rd Qu.:  4284550   ABBV   :  1259
##  Max.   :2049.00   Max.   :618237630   ABC    :  1259
##                                         (Other):611475

s_data$date = as.Date(s_data$date)

str(s_data)

## 'data.frame':    619029 obs. of  7 variables:
##  $ date  : Date, format: "2013-02-08" "2013-02-11" ...
##  $ open  : num  15.1 14.9 14.4 14.3 14.9 ...
##  $ high  : num  15.1 15 14.5 14.9 15 ...
##  $ low   : num  14.6 14.3 14.1 14.2 13.2 ...
##  $ close : num  14.8 14.5 14.3 14.7 14 ...
##  $ volume: int  8407500 8882000 8126000 10259500 31879900 15628000 1135440
0 14725200 11922100 6071400 ...
##  $ Name  : Factor w/ 505 levels "A","AAL","AAP",..: 2 2 2 2 2 2 2 2 2 2 ..
.
##  - attr(*, "na.action")= 'omit' Named int  82950 165735 165858 205077 2398
33 434380 434503 478595 558214 581907 ...
##   ..- attr(*, "names")= chr  "82950" "165735" "165858" "205077" ...
```

Distribution of Closing price

```
library(ggplot2)
ggplot(s_data, aes(close)) + geom_histogram(bins = 50, aes(y = ..density..),
col = "red", fill = "red", alpha = 0.3) + geom_density()# + xlim(c(0, 1000))
```



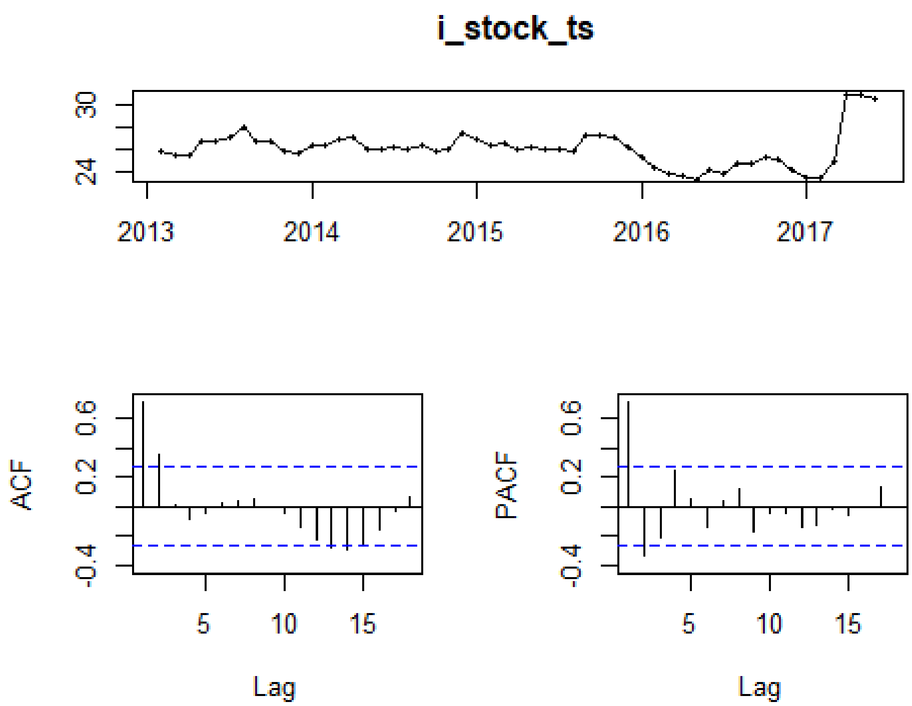
Filtering the Data for only Ticker NFLX (Netflix)

```
#library(stringr)
i_stock =filter(s_data, s_data$Name == "NFLX")
```

Creating the Time Series

```
i_stock_ts = ts(i_stock$close , start = c(2013,2) , end=c(2017,6), frequency
= 12)

train1 <- ts(i_stock$close, start = c(2013,2) , end= c(2016,6), frequency = 1
2)



test1 <- ts(i_stock$close,start = c(2016,6) , end=c(2017,6) ,frequency = 12)

tail(train1,10)

##           Jan     Feb     Mar     Apr     May     Jun Jul Aug     Sep
## 2015                                                          27.2300
## 2016 25.2414 24.2485 23.8128 23.5228 23.2943 24.1943
##           Oct     Nov     Dec
## 2015 27.1771 27.0400 26.0614
## 2016

head(test1,10)
```

```
##          Jan     Feb     Mar Apr May     Jun     Jul     Aug     Sep
## 2016                                25.8528 25.4128 25.4214 26.6098
## 2017 26.7314 26.7357 25.6943
##          Oct     Nov     Dec
## 2016 26.7714 27.0731 28.0643
## 2017

tsdisplay(i_stock_ts)
```



CHECK FOR STATIONARITY

KPSS TEST

```
#Stationarity
library(urca)

kpss.test(train1, null = c("Level"))

##
##  KPSS Test for Level Stationarity
##
## data:  train1
## KPSS Level = 0.59046, Truncation lag parameter = 1, p-value =
## 0.0235

kpss.test(i_stock_ts, null = c("Trend"))
```

```
##
##  KPSS Test for Trend Stationarity
##
## data:  i_stock_ts
## KPSS Trend = 0.12887, Truncation lag parameter = 1, p-value =
## 0.08173
```

The null hypothesis is rejected at the 95% confidence intervals.

A major disadvantage for the KPSS test is that it has a high rate of Type I errors (it tends to reject the null hypothesis too often). If attempts are made to control these errors (by having larger p-values), then that negatively impacts the test's power. One way to deal with the potential for high Type I errors is to combine the KPSS with an ADF test. If the result from both tests suggests that the time series in stationary, then it probably is.

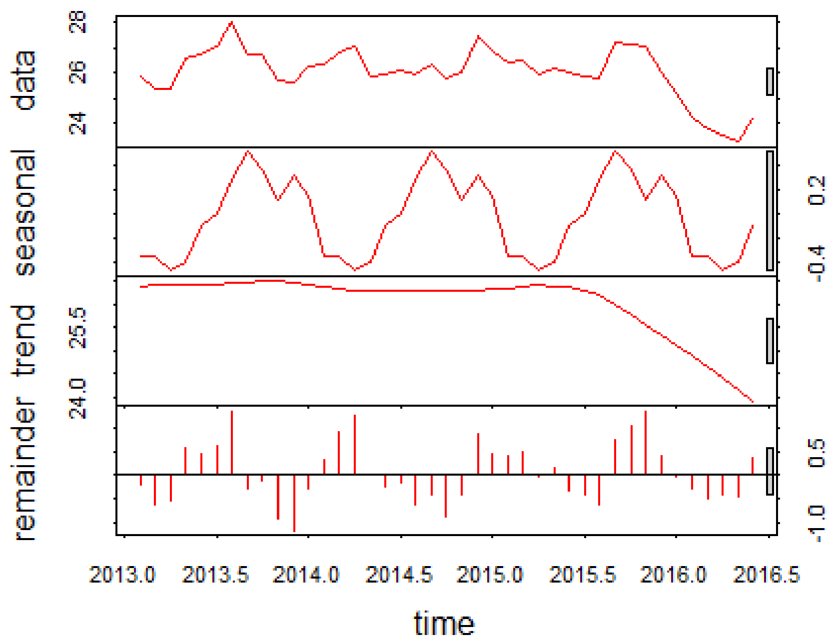
ADF TEST

```
adf.test(i_stock_ts)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  i_stock_ts
## Dickey-Fuller = -2.4758, Lag order = 3, p-value = 0.3829
## alternative hypothesis: stationary
```

The ADF Test Suggests that the data is stationary.

```
#Decomposing Time Series

i_tscomponents <- stl(train1,window(100))
plot(i_tscomponents, col = "red")
```



The first component is the original data, the second component is the seasonality, which indicates there is presence of seasonality in the dataset. The third picture has trend, indicates that there is no trend in the data for long term.

To check if the differencing is required for stationarity

```
a <- ndiffs(train1)
a

## [1] 1
```

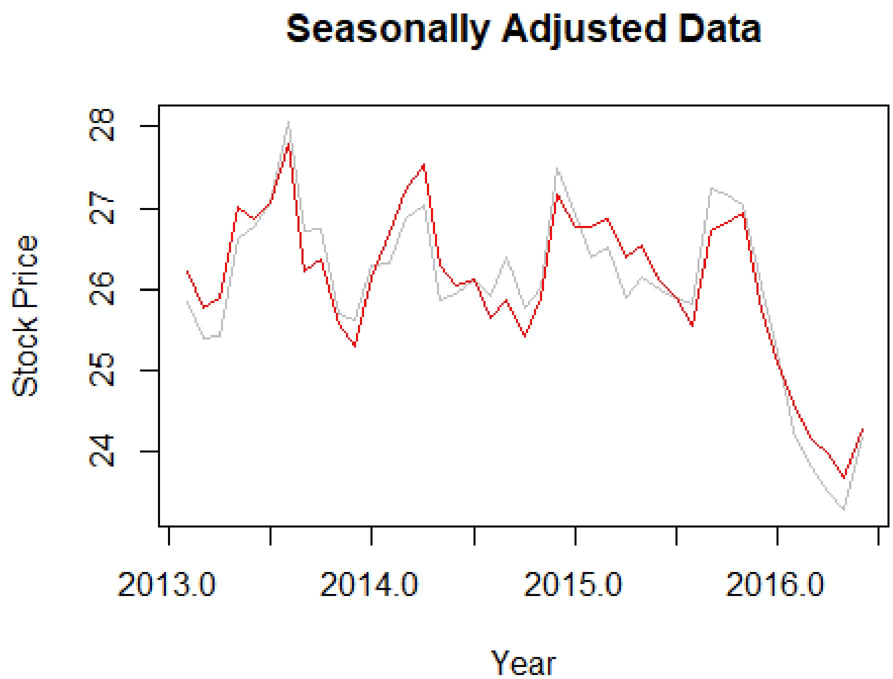
The result indicates that we need to have first order differencing to make our data stationary i.e. since our data has seasonal component it will be removed in first differencing.

SEASONALLY ADJUSTING THE DATA

It is a statistical technique designed to even out periodic swings in statistics or movements in supply and demand related to changing seasons. Seasonal adjustments provide a clearer view of non-seasonal changes in data that would otherwise be overshadowed by the seasonal differences.

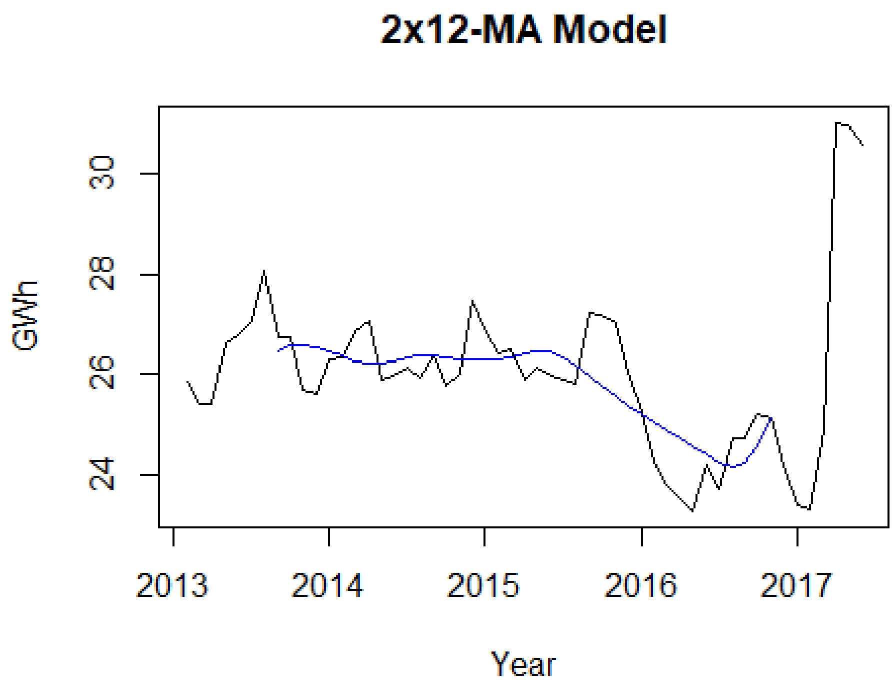
```
plot(train1, col="grey",
     main="Seasonally Adjusted Data",
```

```
     xlab="Year", ylab="Stock Price")
lines(seasadj(i_tscomponents),col="red",ylab="Seasonally adjusted")
```



2x12-MA Model

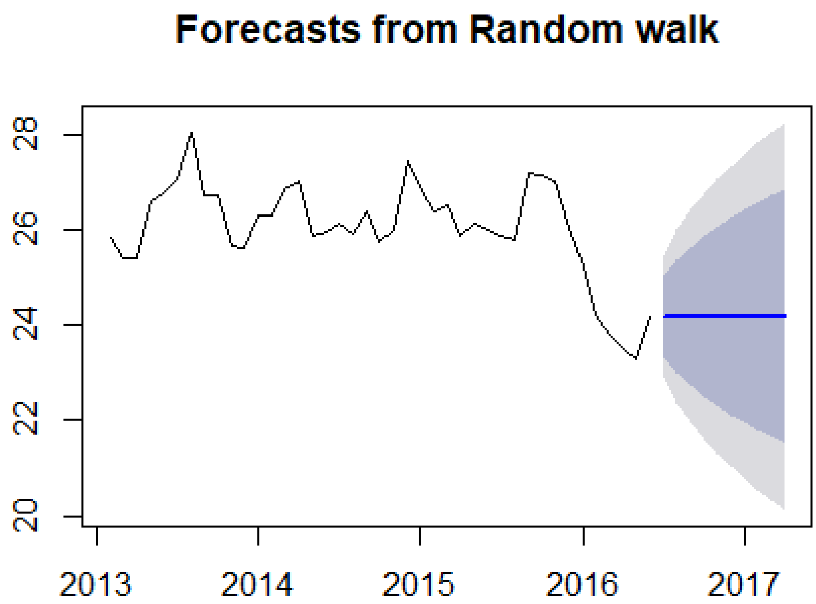
```
plot(i_stock_ts, main="2x12-MA Model",
     ylab="GWh", xlab="Year")
lines(ma(ma(i_stock_ts,2),12),col="blue")
```



RANDOM WALK MODEL

One of the simplest models, yet the random walk model is widely used in the area of finance. A common and serious departure from random behavior is called a random walk. Random Walk Model. By definition, a series is said to follow a random walk if the first differences are random. We will perform the random walk model just to get a base line for accuracy

```
rrr <- rwf(train1)
plot(rrr)
```




```
accuracy(rrr,test1)

##                       ME      RMSE       MAE        MPE     MAPE      MASE
## Training set -0.0414625 0.6586683 0.4983325 -0.1972482 1.909186 0.4413834
## Test set      2.2188300 2.3643380 2.2188300  8.3137559 8.313756 1.9652636
##                       ACF1 Theil's U
## Training set -0.0006188997        NA
## Test set      0.4741675557  3.192681
```

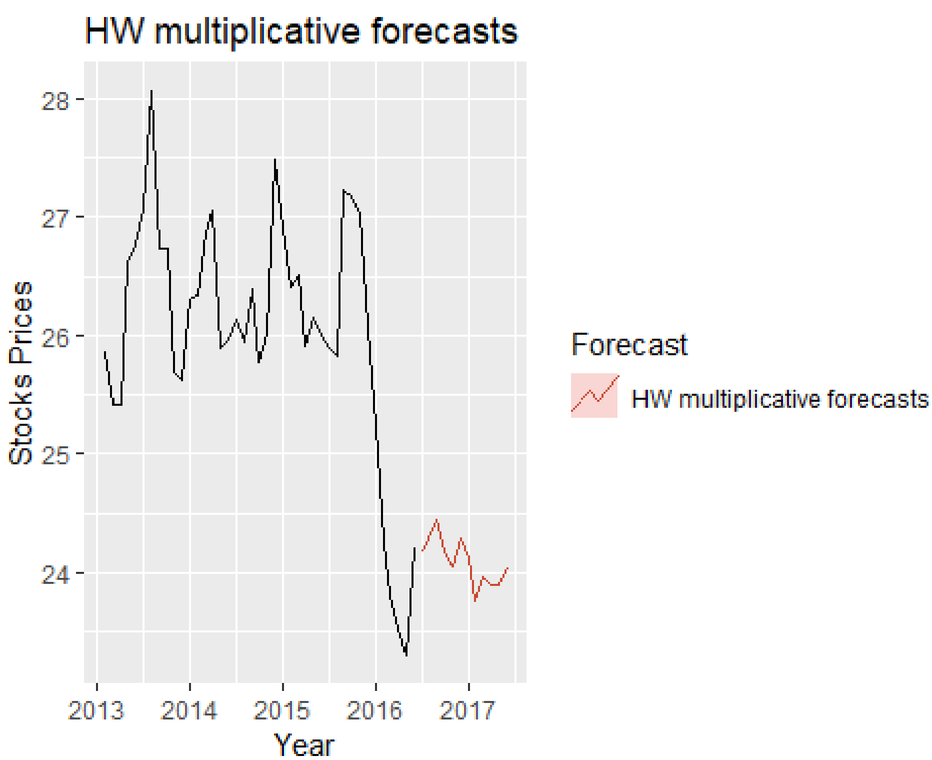
The RMSE of the training set is 0.66. This is okay, but we need to compare it to the test set to get a better idea of the performance of the model. We can see that the accuracy when compared to the test set is bad.

HOLT WINTER FORECASTING

The Holt-Winters seasonal method comprises the forecast equation and three smoothing equations — one for the level ℓt, one for the trend but, and one for the seasonal component set, with corresponding smoothing parameters α , β∗ and γ. We use m to denote the frequency of the seasonality, i.e., the number of seasons in a year. For example, for monthly data m=12.

HOLT WINTER MULTIPLICATIVE FORECASTING

```r
fit2 <- hw(train1,seasonal="multiplicative",damped = TRUE,h=12)
autoplot(train1) +
  autolayer(fit2, series="HW multiplicative forecasts",PI=FALSE) +
  xlab("Year") +
  ylab("Stocks Prices") +
  ggtitle("HW multiplicative forecasts") +
  guides(colour=guide_legend(title="Forecast"))
```



FINDING THE ACCURACY OF HW MULTIPLICATIVE

```r
accuracy(fit2 , test1)
```

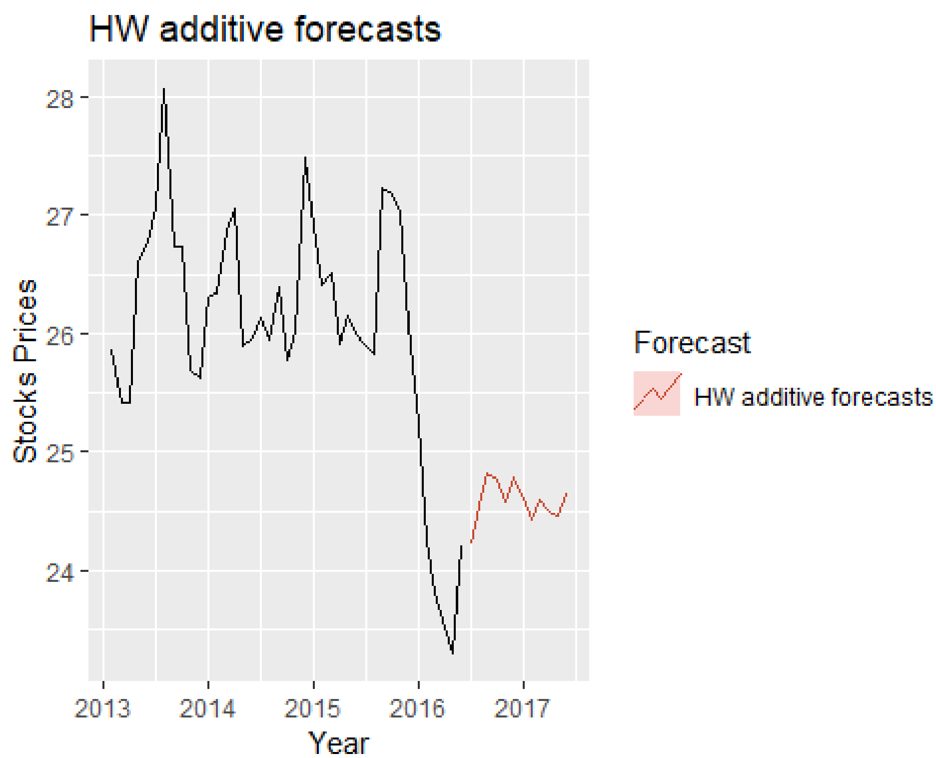
```
##                       ME      RMSE       MAE        MPE     MAPE      MASE
## Training set -0.02168769 0.6299839 0.5024867 -0.1246858 1.925994 0.4450628
## Test set      2.30935238 2.4243932 2.3093524  8.6797304 8.679730 2.0454411
##                    ACF1 Theil's U
## Training set 0.09991679        NA
## Test set     0.48848279   3.45057
```

HOLT WINTER ADDITIVE FORECASTING

It the estimates of the seasonal indices used for forecasting come from the final year of the sample.

```r
fit1 <- hw(train1,seasonal="additive",h=12)
autoplot(train1) +
```

```
  autolayer(fit1, series="HW additive forecasts", PI=FALSE) +
  xlab("Year") +
  ylab("Stocks Prices") +
  ggtitle("HW additive forecasts") +
  guides(colour=guide_legend(title="Forecast"))
```



FINDING THE ACCURACY OF HW ADDITIVE

```
accuracy(fit1 , test1)

##                       ME      RMSE       MAE        MPE     MAPE      MASE
## Training set -0.08869146 0.6265457 0.4858814 -0.3760216 1.866118 0.4303552
## Test set      1.81925235 1.9364778 1.8192524  6.8285446 6.828545 1.6113494
##                    ACF1 Theil's U
## Training set 0.03689511        NA
## Test set     0.46596081  2.737915
```

The plot and summary show some forecasted values. For the model there is the RMSE is 0.63 which is good..

Because both methods have exactly the same number of parameters to estimate, we can compare the training RMSE from both models. In this case, the method with additive seasonality fits the data best. This was to be expected, as the time plot shows that the seasonal variation in the data increases as the level of the series increases. This is also reflected in the two sets of forecasts; the forecasts generated by the method with the additive seasonality display larger and increasing seasonal variation as the level of the forecasts increases compared to the forecasts generated by the method with multiplicative seasonality.

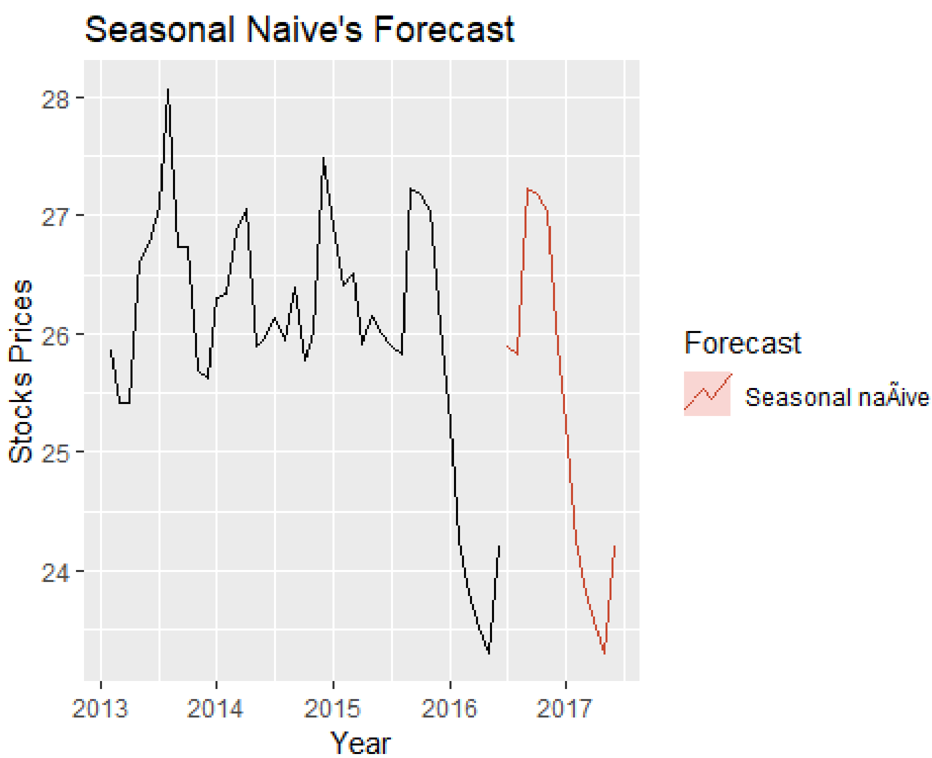
SEASONAL NAIVE FORECASTING

The seasonal naïve, as the names suggests will be just a repetition of the previous seasonal values. We expect it to perform poorly given the variation in the data.

```
seasonal_naive_forecast = snaive(train1,12)
```

Plotting the results of the seasonal forecast

```
autoplot(train1)+
  #autolayer(test1)+
  autolayer(seasonal_naive_forecast, series="Seasonal naÃive", PI=FALSE) +
  xlab("Year") +
  ylab("Stocks Prices") +
  ggtitle("Seasonal Naive's Forecast") +
  guides(colour=guide_legend(title="Forecast"))
```



ACCURACY OF SEASONAL NAIVE

```
accuracy(seasonal_naive_forecast , test1)

##                      ME     RMSE      MAE       MPE     MAPE     MASE
## Training set -0.4390655 1.386221 1.129024 -1.894763 4.448038 1.000000
## Test set      1.1009583 1.705108 1.420758  4.151204 5.377735 1.258395
##                   ACF1 Theil's U
```

```
## Training set 0.6774016        NA
## Test set     0.7292349  2.471859
```

As we expected, the seasonal naïve model just forecasted the previous year values. This results in an RMSE of 1.39 which is higher than the other models.

FITTING AN ARIMA MODEL

Now we fit an ARIMA model to our data. We are fitting ARIMA using auto.arima to find the best fit for our data. function so we won't require to difference the time series before applying to the function.

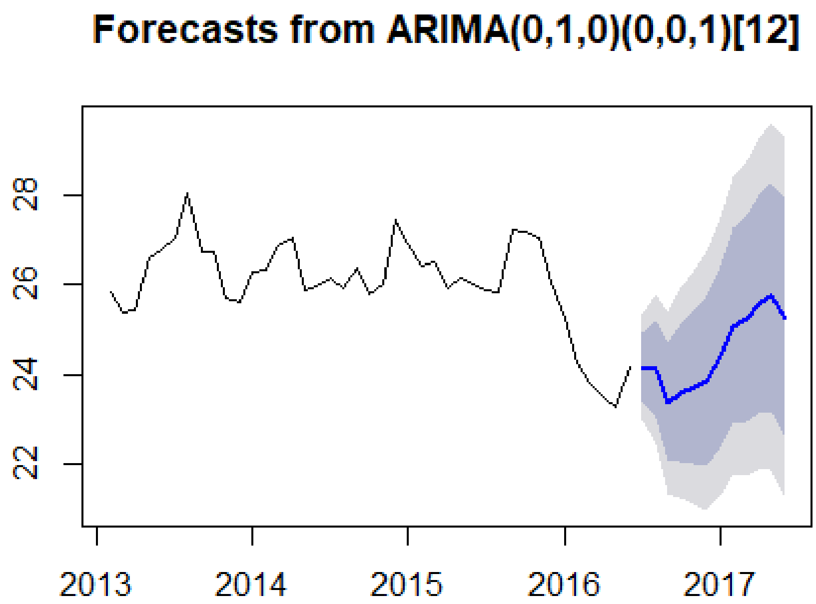
```
i_tsarima <- auto.arima(train1)
i_tsarima
```

```
## Series: train1
## ARIMA(0,1,0)(0,0,1)[12]
##
## Coefficients:
##          sma1
##       -0.5732
## s.e.   0.3228
##
## sigma^2 estimated as 0.354:  log likelihood=-37.82
## AIC=79.64   AICc=79.96   BIC=83.01
```

```
library(forecast)
fit.arima <- forecast(i_tsarima, h=12)
```

Plotting the Forecasts

```
plot(fit.arima)
```



FINDING THE ACCURACY OF THE ARIMA MODEL

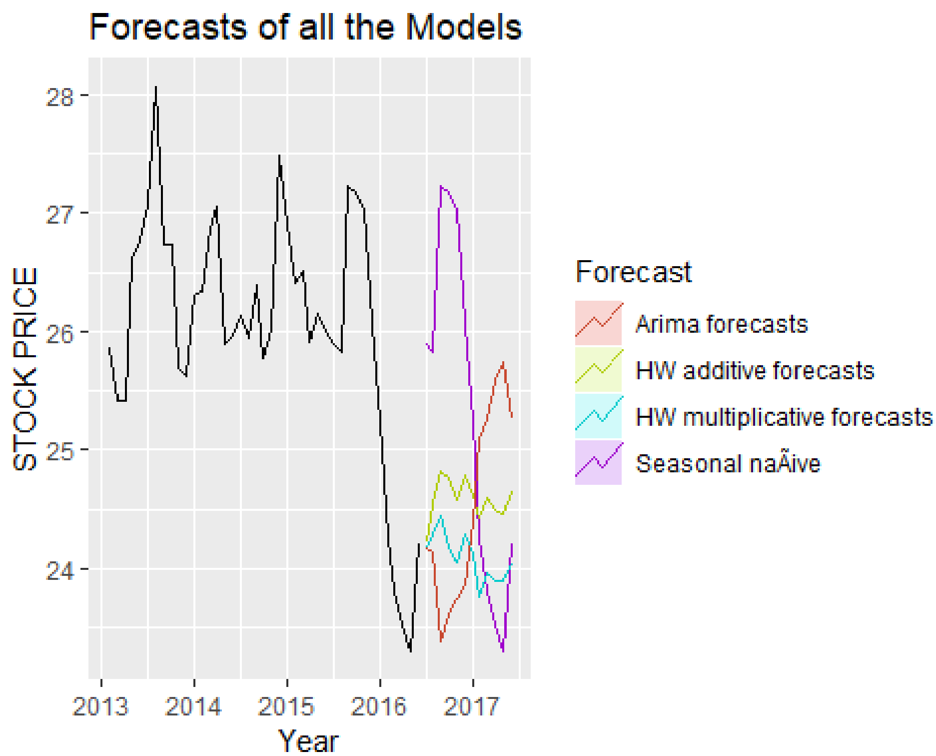
```
accuracy(fit.arima , test1)

##                        ME     RMSE       MAE        MPE     MAPE     MASE
## Training set -0.03770884 0.580315 0.4310208 -0.1817749 1.656678 0.381764
## Test set      1.88174779 2.280202 1.8817478  7.0220918 7.022092 1.666703
##                     ACF1 Theil's U
## Training set 0.01900108        NA
## Test set     0.70605125  3.237533
```

The auto.arima has returned `ARIMA(0,1,0)(0,0,1)[12]` as the best fir model with an AIC score of 79.64. We have tried the model . the fitted arima model indicates first order difference in Non-Seasonal component & first order of Moving Average for smoothing the curve in Seasonal component with 12 seasons.

PLOTTING ALL THE MODELS TO VIEW THE PERFORMANCE

```
autoplot(train1)+
  autolayer(seasonal_naive_forecast, series="Seasonal naÃive", PI=FALSE) +
  autolayer(fit1, series="HW additive forecasts", PI=FALSE) +
  autolayer(fit2, series="HW multiplicative forecasts",PI=FALSE) +
  autolayer(fit.arima, series="Arima forecasts",PI=FALSE) +
  xlab("Year") +
  ylab("STOCK PRICE") +
```

```
ggtitle("Forecasts of all the Models") +
guides(colour=guide_legend(title="Forecast"))
```



FINDING THE ACCURACY OF ALL THE MODELS

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
RANDOM WALK	-0.041	0.658	0.498	-0.197	1.90	0.44	-0.0006
HW (MUL)	-0.021	0.629	2.30	-0.12	1.925	0.445	0.099
HW (ADD)	-0.088	0.626	0.48	-0.376	1.866	0.430	0.036
S.NAIVE	-0.439	1.386	1.129	-1.894	4.44	1.00	0.677
ARIMA	-0.037	0.58	0.431	-0.181	1.656	0.381	0.01

CONCLUSION

We saw that initially the data was stationary and had seasonality but using Arima model we could make the data stationary without any differencing. Even though we got the model with a low RMSE compared to other models. Now this model can be used to predict good enough. May be this model can be improved with more data which includes a larger proportion of training set.

CHALLENGES

There are lots of analyzing methods, we must use the same standard to judge their usefulness. Then we select the best model for forecast the future based on what criteria we prefer. For comprehensive comparisons, we will apply diversified ways to show the advantages and disadvantages of varied models but not just using limited methods. So, the overall calculation is huge amount of work. Everyone need to be consistent with each other

REFERENCES

1. https://otexts.org/fpp2/