Project

Ngone Lo

06/12/2020

```r
setwd("~/Classes/FALL 2020 CLASSES/B.INF2167 R4DS/Assignments/Group Project/Codes and Datasets Used")
```

Predicting the daily "Low Stock Price" to find the number to look for in order to determine when to buy stock on a daily basis.

Given today's data (stock and pharma sales), can we predict the "low stock price" for tomorrow? We will use:

1. Linear Regression
2. KNN Regression
3. Random Forest
4. SVM 6: Ensemble Model
5. Simple Moving Average Method
6. Auto-ARIMA (auto-regressive integrated moving average)
7. Average of Previous 5 Days Method

#Libraries needed

```r
#install.packages('RCurl')
#install.packages('MASS')
#install.packages('leaps')
#install.packages('corrplot')
#install.packages('caret')
#install.packages("FNN")
#install.packages("mlbench)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(RCurl) # getURL
```

```
##
## Attaching package: 'RCurl'

## The following object is masked from 'package:tidyr':
##
##     complete
```

```
library(MASS) # stepwise regression
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(leaps) # all subsets regression
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(FNN)
library(mlbench)
library(rcompanion)
library(e1071)
```

#Import Merged Dataset and change column names of drugs (pharma data)

```
data<-read.csv("daily_sales_stock.csv")
data<-drop_na(data)
colnames(data)[3:10]<-c("Med4RheumArth","Med4OstArth", "Aspirin","Ibuprofen", "Med4Tension", "Med4Sleep
str(data)
```

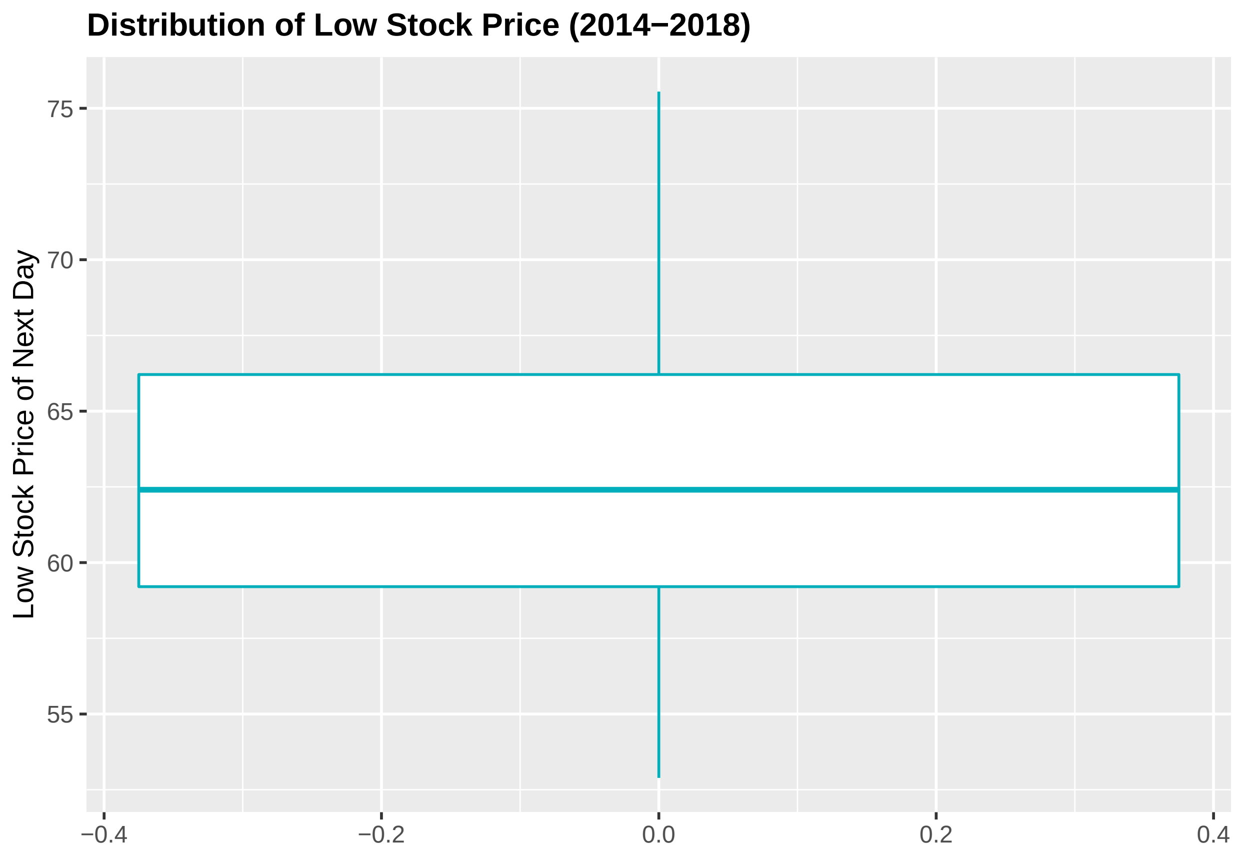
```
## 'data.frame':    1258 obs. of  18 variables:
##  $ X             : int  1 2 5 6 7 8 9 12 13 14 ...
##  $ date          : chr  "2014-01-02" "2014-01-03" "2014-01-06" "2014-01-07" ...
##  $ Med4RheumArth : num  0 8 5 0 5.33 7 5 7.34 6 4 ...
##  $ Med4OstArth   : num  3.67 4 1 0 3 1.68 2 7.66 1.33 2.34 ...
##  $ Aspirin       : num  3.4 4.4 4.5 0 10.5 8 2 6.2 12.3 5 ...
##  $ Ibuprofen     : num  32.4 50.6 21.7 0 26.4 25 53.3 52 33.7 26.7 ...
##  $ Med4Tension   : num  7 16 16 0 19 16 15 9 6 12 ...
##  $ Med4Sleep     : num  0 0 2 0 1 0 2 0 1 2 ...
##  $ Meds4Asthma   : num  0 20 6 0 10 3 0 7 0 3 ...
##  $ Meds4Allergy  : num  2 4 2 0 0 2 2 1 2 3 ...
##  $ hour          : int  248 276 276 276 276 276 276 276 276 276 ...
##  $ weekday_name  : chr  "Thursday" "Friday" "Monday" "Tuesday" ...
##  $ open_mean     : num  58.2 56.9 57.3 57.3 57.3 ...
##  $ high_mean     : num  58.4 57.4 57.6 57.7 57.6 ...
##  $ low_mean      : num  57.9 56.9 56.8 57.1 57 ...
##  $ close_mean    : num  58.1 57.2 57 57.4 57.2 ...
##  $ adj_close_mean: num  45.8 45.3 45.2 45.5 45.4 ...
##  $ volume_mean   : num  5787717 5177143 8423543 8769257 8656971 ...
```

#creating the column/variable low_price_next_day (Low Price of stock of next day) as our dependent variable

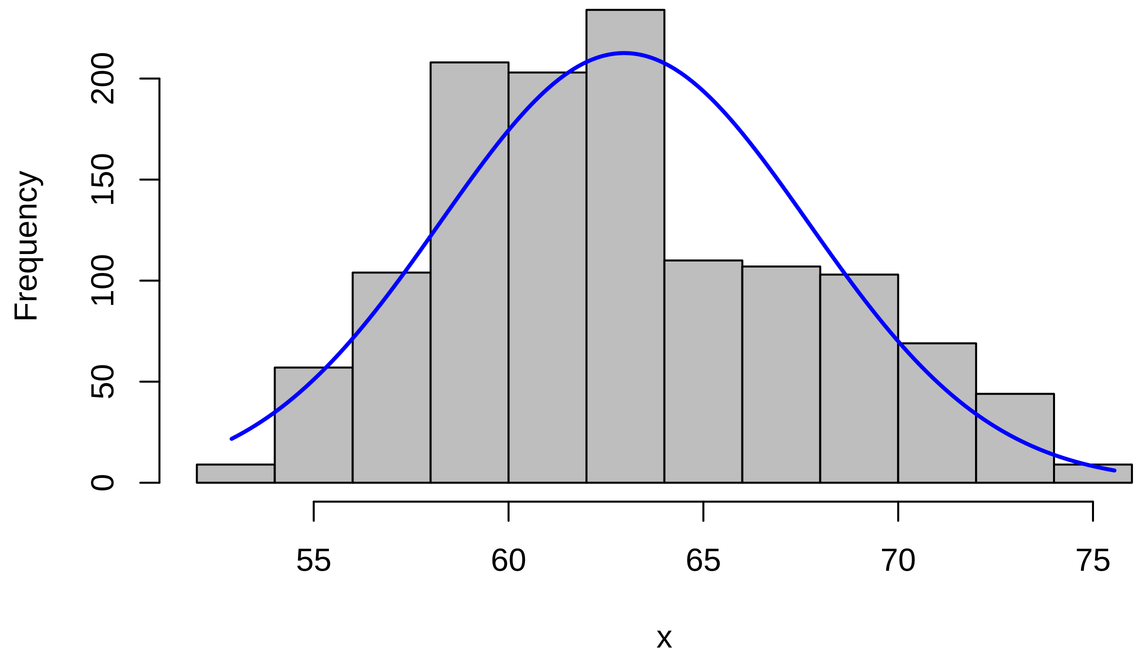
```r
shift <- function(x, n){
  c(x[-(seq(n))], rep(NA, n))}

data$low_price_next_day<-shift(data$low_mean,1)
data<-drop_na(data)
```

#Explore Dependent (Target) Varibale: low_price_next_day (Low Price of stock of next day)

```r
ggplot(data=data)+
  geom_boxplot(mapping = aes(x = low_price_next_day), color="#00AFBB") +
  coord_flip()+
  xlab("Low Stock Price of Next Day") +
  ggtitle("Distribution of Low Stock Price (2014-2018)") +
  theme(plot.title = element_text(size=12, face = "bold"))
```



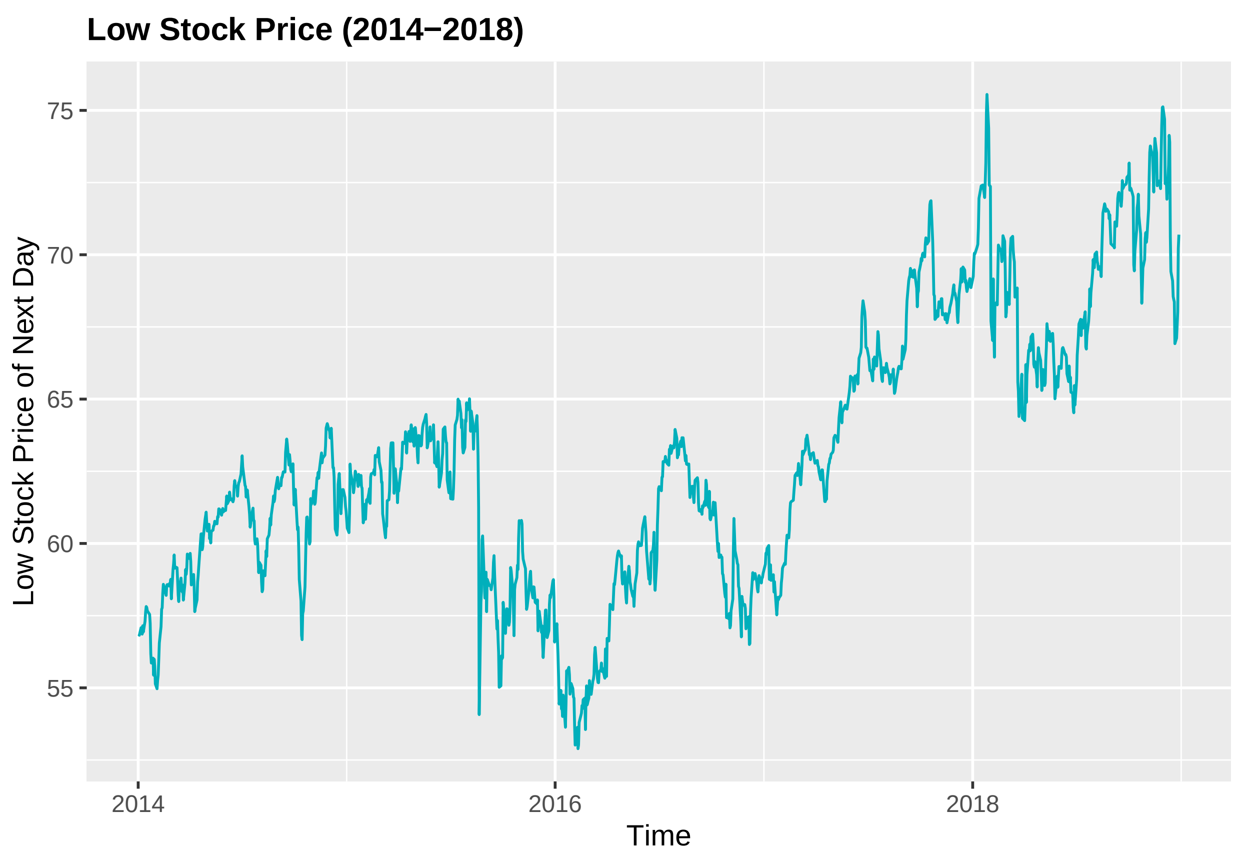
```
plotNormalHistogram(data$low_price_next_day)
```



```r
paste("Skewness of DV (next day low price):" ,skewness(data$low_price_next_day))
```

```
## [1] "Skewness of DV (next day low price): 0.365709614365617"
```

```r
ggplot(data = data, aes(x = as.Date(date), y = low_price_next_day))+
  geom_line(color = "#00AFBB")+
  xlab("Time")+
  ylab("Low Stock Price of Next Day")+
  ggtitle("Low Stock Price (2014-2018)") +
  theme(plot.title = element_text(size=12, face = "bold"))
```



The target variable is fairly symmetrical/normally distributed.

#Data Preprocessing (of IVs) ##Structure of Dataset

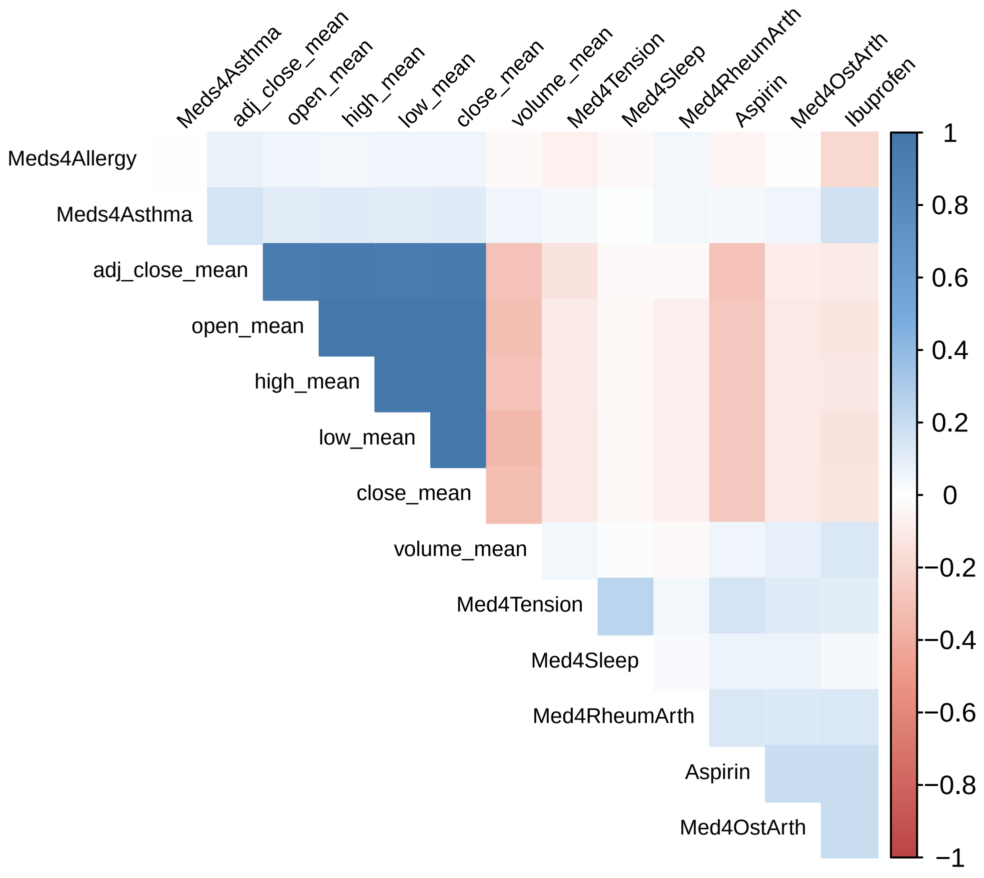
```
data<-data[,-c(1,11)] #drop index(X)  and hour columns
str(data)
```

```
## 'data.frame':    1257 obs. of  17 variables:
##  $ date             : chr  "2014-01-02" "2014-01-03" "2014-01-06" "2014-01-07" ...
##  $ Med4RheumArth     : num  0 8 5 0 5.33 7 5 7.34 6 4 ...
##  $ Med4OstArth       : num  3.67 4 1 0 3 1.68 2 7.66 1.33 2.34 ...
##  $ Aspirin           : num  3.4 4.4 4.5 0 10.5 8 2 6.2 12.3 5 ...
##  $ Ibuprofen         : num  32.4 50.6 21.7 0 26.4 25 53.3 52 33.7 26.7 ...
##  $ Med4Tension       : num  7 16 16 0 19 16 15 9 6 12 ...
##  $ Med4Sleep         : num  0 0 2 0 1 0 2 0 1 2 ...
##  $ Meds4Asthma       : num  0 20 6 0 10 3 0 7 0 3 ...
##  $ Meds4Allergy      : num  2 4 2 0 0 2 2 1 2 3 ...
##  $ weekday_name      : chr  "Thursday" "Friday" "Monday" "Tuesday" ...
##  $ open_mean         : num  58.2 56.9 57.3 57.3 57.3 ...
##  $ high_mean         : num  58.4 57.4 57.6 57.7 57.6 ...
##  $ low_mean          : num  57.9 56.9 56.8 57.1 57 ...
##  $ close_mean        : num  58.1 57.2 57 57.4 57.2 ...
##  $ adj_close_mean    : num  45.8 45.3 45.2 45.5 45.4 ...
##  $ volume_mean       : num  5787717 5177143 8423543 8769257 8656971 ...
##  $ low_price_next_day: num  56.9 56.8 57.1 57 56.9 ...
```

Correlation Analysis

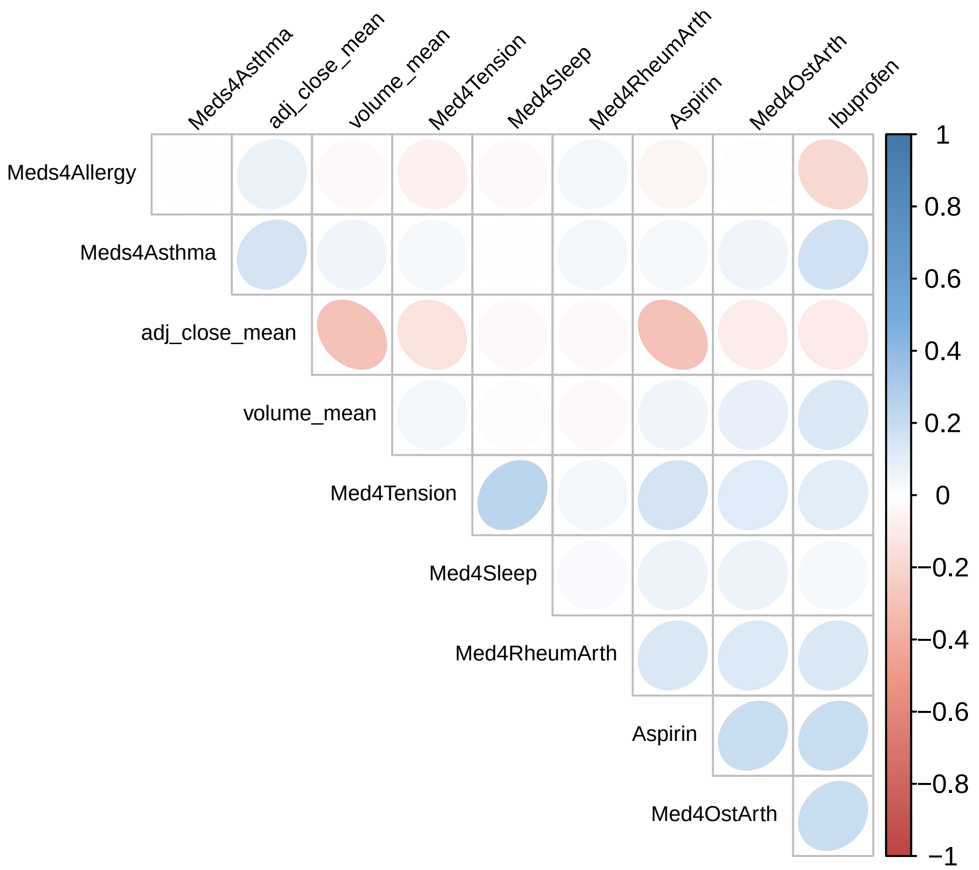
###Identify highly correlated "numeric" variables

```
###draw correlation matrix of the numeric independent variables only
num_data <- data[,-c(1,10,17)] ### remove date and weekday_name as well as low_price_next_day because i
correlationMatrix <- cor(num_data, method = "pearson")
col <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD", "#4477AA"))
corrplot(correlationMatrix, method="color", col=col(200),
         type="upper", order="hclust",
         tl.col="black", tl.srt=45, tl.cex= 0.7, #Text label color and rotation
         # Combine with significance
         sig.level = 0.01,
         # hide correlation coefficient on the principal diagonal
         diag=FALSE
)
```



###Remove highly correlated numeric IVs (0.6)

```
highlyCorrelated <- findCorrelation(correlationMatrix, cutoff=0.6)
noncor <- num_data[,-highlyCorrelated]  #keep only those not highly
correlationMatrix2 <- cor(noncor, method = "pearson")  ### only numeric vars
col <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD", "#4477AA"))
corrplot(correlationMatrix2, method="ellipse", col=col(200),
         type="upper", order="hclust",
         tl.col="black", tl.srt=45, tl.cex= 0.7, #Text label color and rotation
         # Combine with significance
         sig.level = 0.01,
         # hide correlation coefficient on the principal diagonal
         diag=FALSE
)
```



##Normalizing numeric IVs

```
normalize <- function(x) {
                return ((x - min(x)) / (max(x) - min(x))) }

noncor_n <- as.data.frame(lapply(noncor, normalize))

data_n<-cbind(noncor_n,data[,c(1,10,17)])
```

##Features Selection by Backward Elimination

```
full <- lm(low_price_next_day~.,data=data_n[,-11])
stepB <- stepAIC(full, direction= "backward", trace=TRUE)
```

```
## Start:  AIC=973.62
## low_price_next_day ~ Med4RheumArth + Med4OstArth + Aspirin +
##     Ibuprofen + Med4Tension + Med4Sleep + Meds4Asthma + Meds4Allergy +
##     adj_close_mean + volume_mean + weekday_name
##
```

```
##                  Df Sum of Sq     RSS    AIC
## - weekday_name    4       1.8  2664.7  966.5
## - Med4OstArth     1       1.9  2664.8  972.5
## - Meds4Allergy    1       3.5  2666.4  973.3
## <none>                         2662.9  973.6
## - Med4Sleep       1       7.3  2670.2  975.1
## - Aspirin         1      11.6  2674.6  977.1
## - Meds4Asthma     1      19.2  2682.1  980.7
## - Med4Tension     1      25.9  2688.9  983.8
## - Ibuprofen       1      35.6  2698.5  988.3
## - Med4RheumArth   1      44.4  2707.3  992.4
## - volume_mean     1      88.2  2751.1 1012.6
## - adj_close_mean  1   19610.0 22272.9 3641.4
##
## Step:  AIC=966.45
## low_price_next_day ~ Med4RheumArth + Med4OstArth + Aspirin +
##     Ibuprofen + Med4Tension + Med4Sleep + Meds4Asthma + Meds4Allergy +
##     adj_close_mean + volume_mean
##
##                  Df Sum of Sq     RSS    AIC
## - Med4OstArth     1       2.0  2666.7  965.4
## - Meds4Allergy    1       3.2  2667.9  966.0
## <none>                         2664.7  966.5
## - Med4Sleep       1       7.0  2671.6  967.7
## - Aspirin         1      12.2  2676.9  970.2
## - Meds4Asthma     1      18.8  2683.4  973.3
## - Med4Tension     1      26.6  2691.2  976.9
## - Ibuprofen       1      35.7  2700.4  981.2
## - Med4RheumArth   1      43.6  2708.2  984.8
## - volume_mean     1      88.1  2752.7 1005.3
## - adj_close_mean  1   19628.8 22293.5 3634.6
##
## Step:  AIC=965.39
## low_price_next_day ~ Med4RheumArth + Aspirin + Ibuprofen + Med4Tension +
##     Med4Sleep + Meds4Asthma + Meds4Allergy + adj_close_mean +
##     volume_mean
##
##                  Df Sum of Sq     RSS    AIC
## - Meds4Allergy    1       3.4  2670.0  965.0
## <none>                         2666.7  965.4
## - Med4Sleep       1       7.2  2673.8  966.8
## - Aspirin         1      11.1  2677.8  968.6
## - Meds4Asthma     1      18.9  2685.6  972.3
## - Med4Tension     1      25.7  2692.4  975.5
## - Ibuprofen       1      39.0  2705.6  981.6
## - Med4RheumArth   1      45.5  2712.2  984.7
## - volume_mean     1      89.5  2756.2 1004.9
## - adj_close_mean  1   19637.8 22304.5 3633.2
##
## Step:  AIC=964.98
## low_price_next_day ~ Med4RheumArth + Aspirin + Ibuprofen + Med4Tension +
##     Med4Sleep + Meds4Asthma + adj_close_mean + volume_mean
##
##                  Df Sum of Sq     RSS    AIC
## <none>                        2670.0  965.0
## - Med4Sleep       1       7.1  2677.1  966.3
## - Aspirin         1      11.1  2681.1  968.2
## - Meds4Asthma     1      19.4  2689.5  972.1
## - Med4Tension     1      26.5  2696.5  975.4
## - Ibuprofen       1      36.1  2706.1  979.8
## - Med4RheumArth   1      47.7  2717.7  985.2
## - volume_mean     1      90.4  2760.4 1004.8
## - adj_close_mean  1   19665.2 22335.3 3632.9
```

```
summary(stepB)
```

```
##
## Call:
## lm(formula = low_price_next_day ~ Med4RheumArth + Aspirin + Ibuprofen +
##     Med4Tension + Med4Sleep + Meds4Asthma + adj_close_mean +
##     volume_mean, data = data_n[, -11])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7744 -1.1178 -0.0119  1.2123  3.5585
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     56.2114     0.1981 283.737  < 2e-16 ***
## Med4RheumArth   -1.2701     0.2691  -4.720 2.62e-06 ***
## Aspirin          0.6686     0.2939   2.275 0.023065 *
## Ibuprofen       -2.0623     0.5022  -4.106 4.28e-05 ***
## Med4Tension      1.4132     0.4018   3.517 0.000451 ***
## Med4Sleep       -0.6049     0.3319  -1.822 0.068625 .
## Meds4Asthma     -0.8475     0.2812  -3.014 0.002631 **
## adj_close_mean  19.3454     0.2018  95.873  < 2e-16 ***
## volume_mean     -4.0254     0.6194  -6.499 1.17e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.463 on 1248 degrees of freedom
## Multiple R-squared:  0.9045, Adjusted R-squared:  0.9039
## F-statistic:  1477 on 8 and 1248 DF,  p-value: < 2.2e-16
```

Significant IVs are: Med4RheumArth, Aspirin, Ibuprofen, Med4Tension, Meds4Asthma, Meds4Allergy, adj_close_mean, and volume_mean. Hence will drop Med4OstArth, Med4Sleep, Meds4Allergy, and weekday_name

##Dropping non-significant IVs (but still keeping date)

```
data_n<-data_n[,-c(2,6,8,12)]
colnames(data_n)
```

```
## [1] "Med4RheumArth"      "Aspirin"            "Ibuprofen"
## [4] "Med4Tension"        "Meds4Asthma"        "adj_close_mean"
## [7] "volume_mean"        "date"               "low_price_next_day"
```

#Split Training-Test sets We will split the dataset into training (2014-2017) and test(2018) sets and finally drop the variable date from all sets

```
cut<-which(data$date=="2018-01-02")

train<-data_n[1:(cut-1),-8]
test<-data_n[cut:nrow(data_n),-8]
```

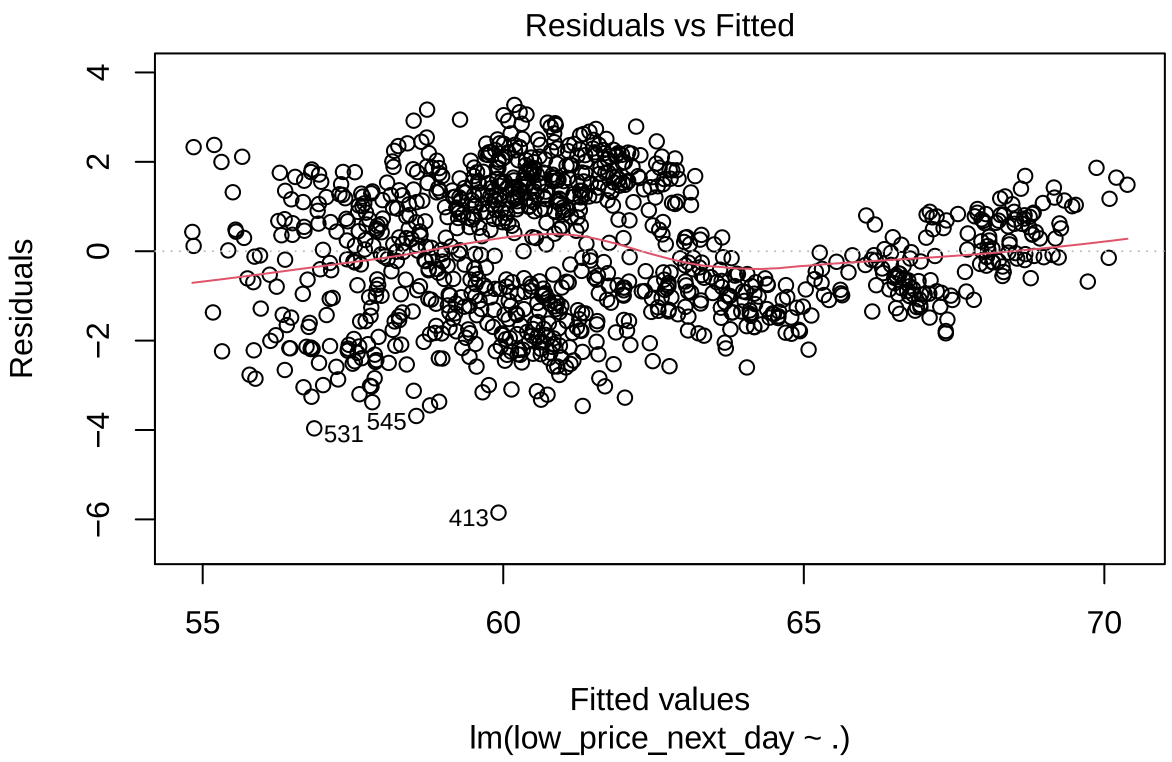
```
test_labels <- test[,8]      ##DV in the test set
test <- test[,-8]            ##Only keep IV in the test (set)
```

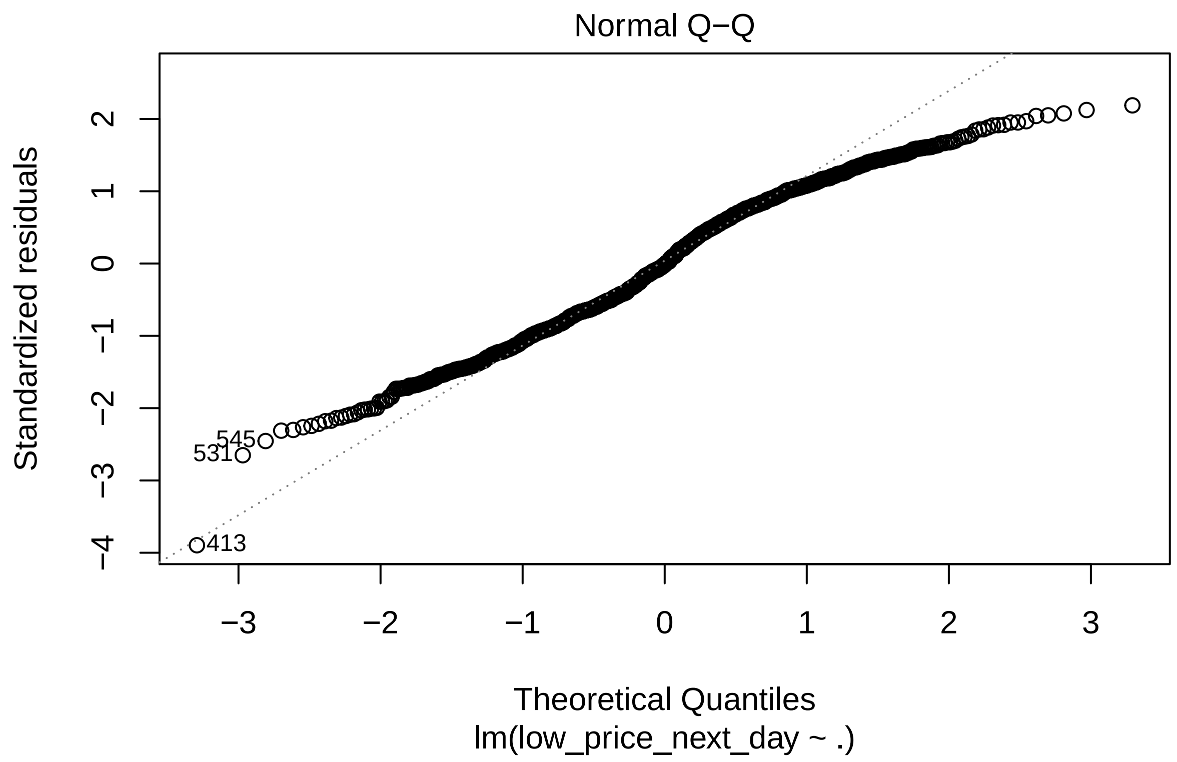
#Linear Regression Model ## Model Assumptions

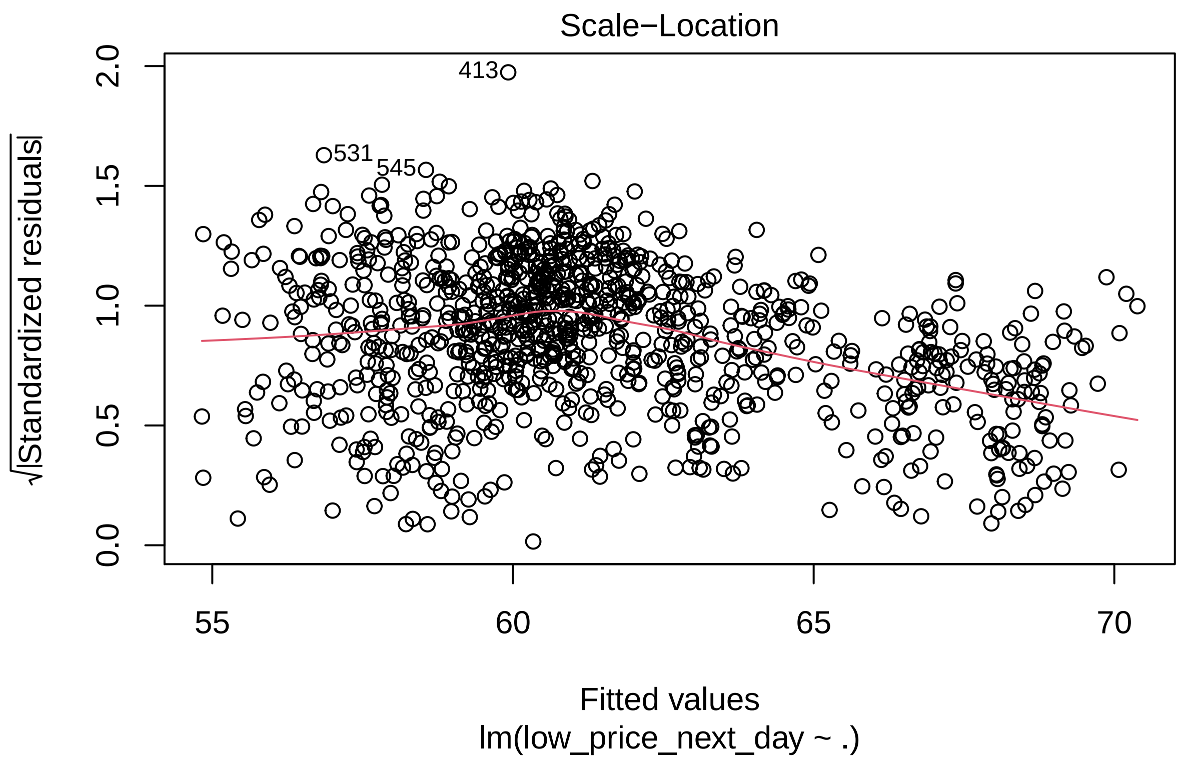
```
LRmodel1 <- lm(low_price_next_day ~., data = train)
summary(LRmodel1)
```

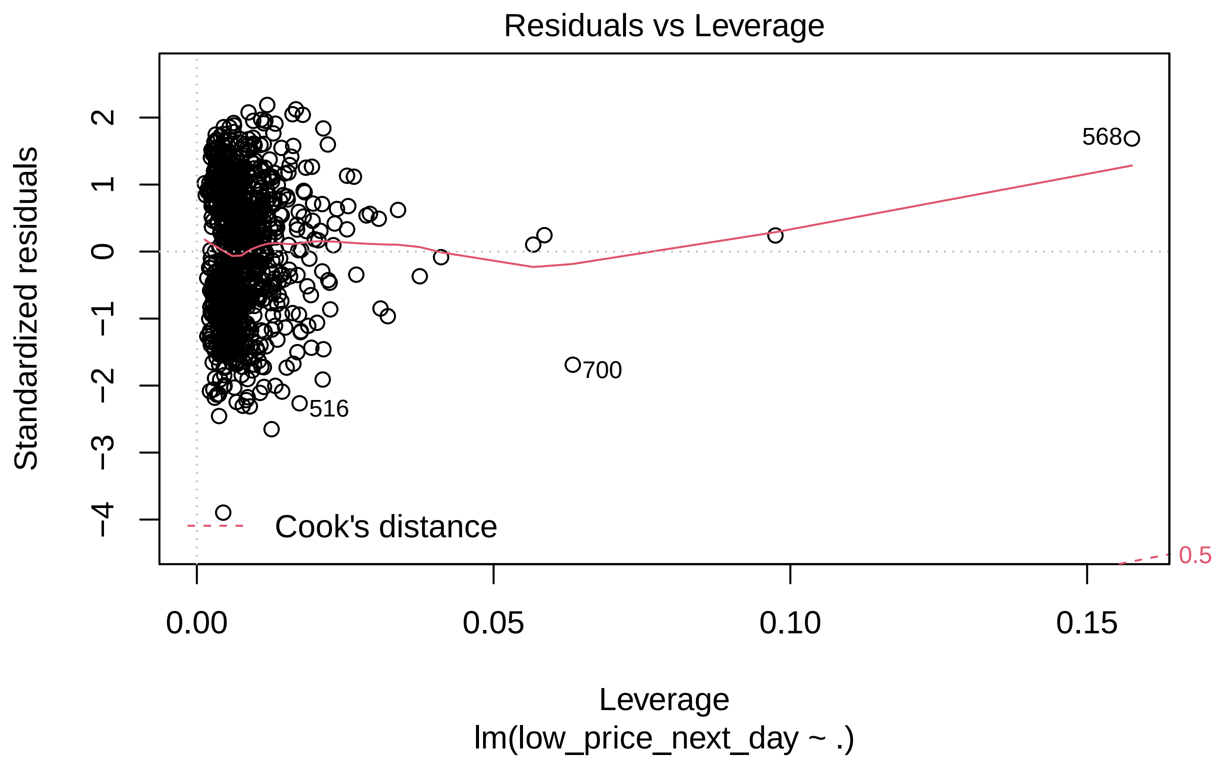
```
##
## Call:
## lm(formula = low_price_next_day ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.8488 -1.1264 -0.0219  1.2484  3.2724
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       56.4242     0.2492 226.462  < 2e-16 ***
## Med4RheumArth    -1.4841     0.3101  -4.786 1.96e-06 ***
## Aspirin           0.4539     0.3304   1.374 0.169777
## Ibuprofen        -3.1625     0.5727  -5.522 4.26e-08 ***
## Med4Tension       1.6541     0.4383   3.774 0.000170 ***
## Meds4Asthma      -1.2065     0.3468  -3.479 0.000524 ***
## adj_close_mean   19.7365     0.3386  58.295  < 2e-16 ***
## volume_mean      -3.9802     0.7122  -5.589 2.95e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.504 on 999 degrees of freedom
## Multiple R-squared:  0.8335, Adjusted R-squared:  0.8323
## F-statistic: 714.4 on 7 and 999 DF,  p-value: < 2.2e-16
```

```
plot(LRmodel1)
```


Residuals vs Fitted
Residuals
Fitted values
lm(low_price_next_day ~ .)
4
2
0
−2
−4
−6
55
60
65
70
531
545
413


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(low_price_next_day ~ .)
545
531
413
−4
−3
−2
−1
0
1
2
−3
−2
−1
0
1
2
3


Scale−Location
413
531
545
2.0
1.5
1.0
0.5
0.0
√|Standardized residuals|
55
60
65
70
Fitted values
lm(low_price_next_day ~ .)



All four assumptions of parametric linear regression were violated here. However, we decided to go against a transformation for interpretability and comparison (with other models) reasons.

##cross validation parameters

```
ctrl <- trainControl(method="repeatedcv", number =5, repeats=3)
```

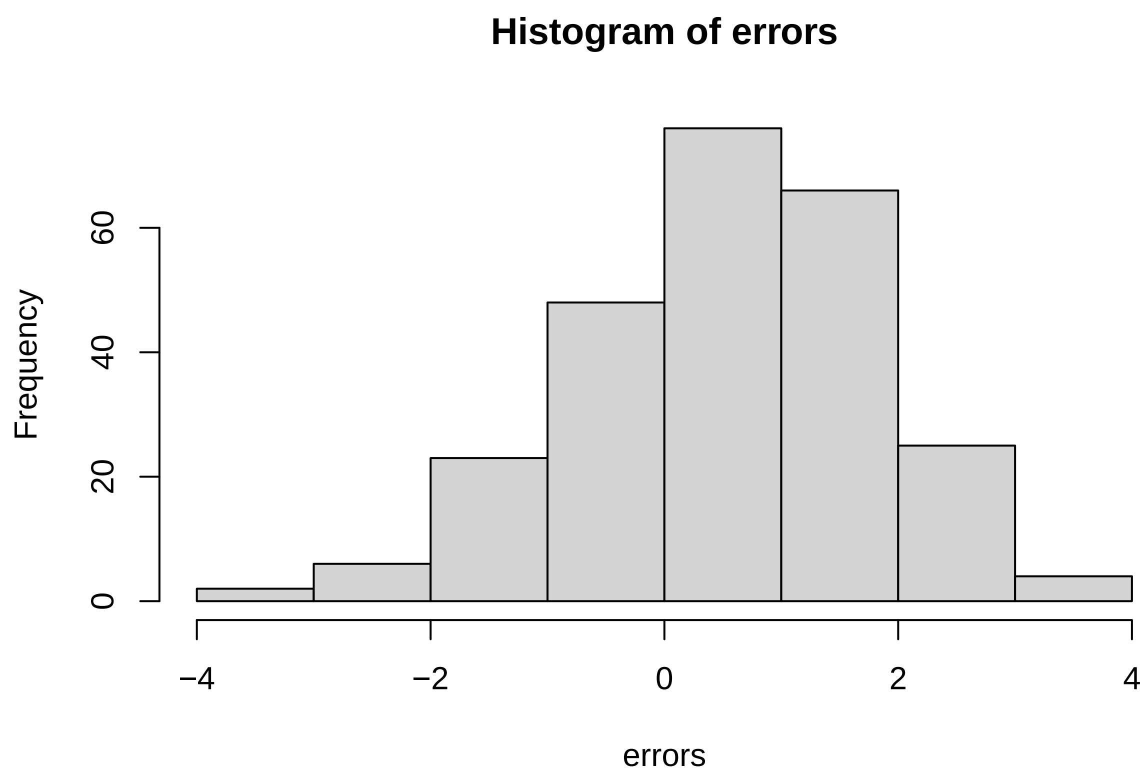
##Model and distribution of errors

```
set.seed(123)
LRmodel <- train(low_price_next_day ~ ., data= train, method="lm", trControl = ctrl)


test_predLR <- predict(LRmodel, test)
errors <- test_predLR - test_labels
summary(LRmodel)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.8488 -1.1264 -0.0219  1.2484  3.2724
##
## Coefficients:
```

```
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      56.4242     0.2492 226.462  < 2e-16 ***
## Med4RheumArth    -1.4841     0.3101  -4.786 1.96e-06 ***
## Aspirin           0.4539     0.3304   1.374 0.169777
## Ibuprofen        -3.1625     0.5727  -5.522 4.26e-08 ***
## Med4Tension       1.6541     0.4383   3.774 0.000170 ***
## Meds4Asthma      -1.2065     0.3468  -3.479 0.000524 ***
## adj_close_mean   19.7365     0.3386  58.295  < 2e-16 ***
## volume_mean      -3.9802     0.7122  -5.589 2.95e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.504 on 999 degrees of freedom
## Multiple R-squared:  0.8335, Adjusted R-squared:  0.8323
## F-statistic: 714.4 on 7 and 999 DF,  p-value: < 2.2e-16
```

```
hist(errors)
```



##RMSE and cases with less than 25%, 10%, 5%, and 1% error

```
rmse <- sqrt(mean((test_labels - test_predLR)^2))

rel_change <- abs(errors) / test_labels
pred25 <- sum((rel_change<0.25)=="TRUE")/nrow(test)    ## gives the count of those who are true on the c
##OR pred25 <- table(rel_change<0.25)["TRUE"] / nrow(test)
```

```
pred10 <- sum((rel_change<0.10)=="TRUE")/nrow(test)  ## gives the count of those who are true on the co

pred5 <- sum((rel_change<0.05)=="TRUE")/nrow(test)  ## gives the count of those who are true on the con

pred1 <- sum((rel_change<0.01)=="TRUE")/nrow(test)  ## gives the count of those who are true on the con

paste("RMSE:", rmse)
```

```
## [1] "RMSE: 1.36209276087444"
```

```
paste("Pred(25%):", pred25)
```

```
## [1] "Pred(25%): 1"
```

```
paste("Pred(10%):", pred10)
```

```
## [1] "Pred(10%): 1"
```

```
paste("Pred(5%):", pred5)
```

```
## [1] "Pred(5%): 0.996"
```

```
paste("Pred(1%):", pred1)
```

```
## [1] "Pred(1%): 0.348"
```

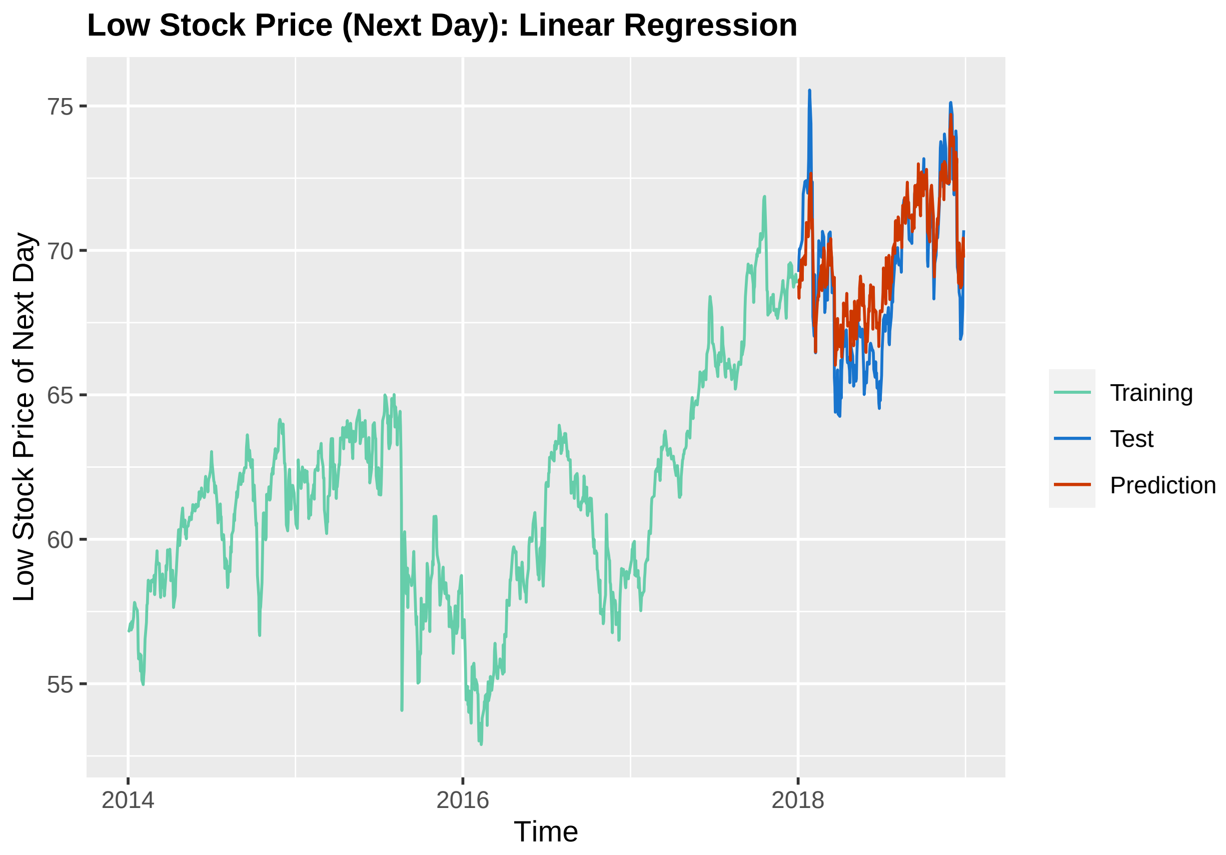
##Creating the time series dataset for low stock price (of next day)

```
data_ts<-data_n[,c(8,9)]
data_ts$date <- as.Date(data_ts$date)
```

```
train_ts<-data_ts[1:(cut-1),]
test_ts<-data_ts[cut:nrow(data_n),]
```

##Plot

```
test_ts$preds<- test_predLR
ggplot(data = train_ts, aes(x = as.Date(date), y = low_price_next_day,color = "aquamarine3"))+
  geom_line()+
  geom_line(data = test_ts, aes(x = as.Date(date), y = low_price_next_day ,color="dodgerblue3"))+
  geom_line(data = test_ts, aes(x = as.Date(date), y = preds,color="orangered3"))+
  scale_color_identity(name = " ",
                       breaks = c("aquamarine3", "dodgerblue3", "orangered3"),
                       labels = c("Training", "Test", "Prediction"),
                       guide = "legend")+
  xlab("Time")+
  ylab("Low Stock Price of Next Day")+
  ggtitle("Low Stock Price (Next Day): Linear Regression") +
  theme(plot.title = element_text(size=12, face = "bold"))
```

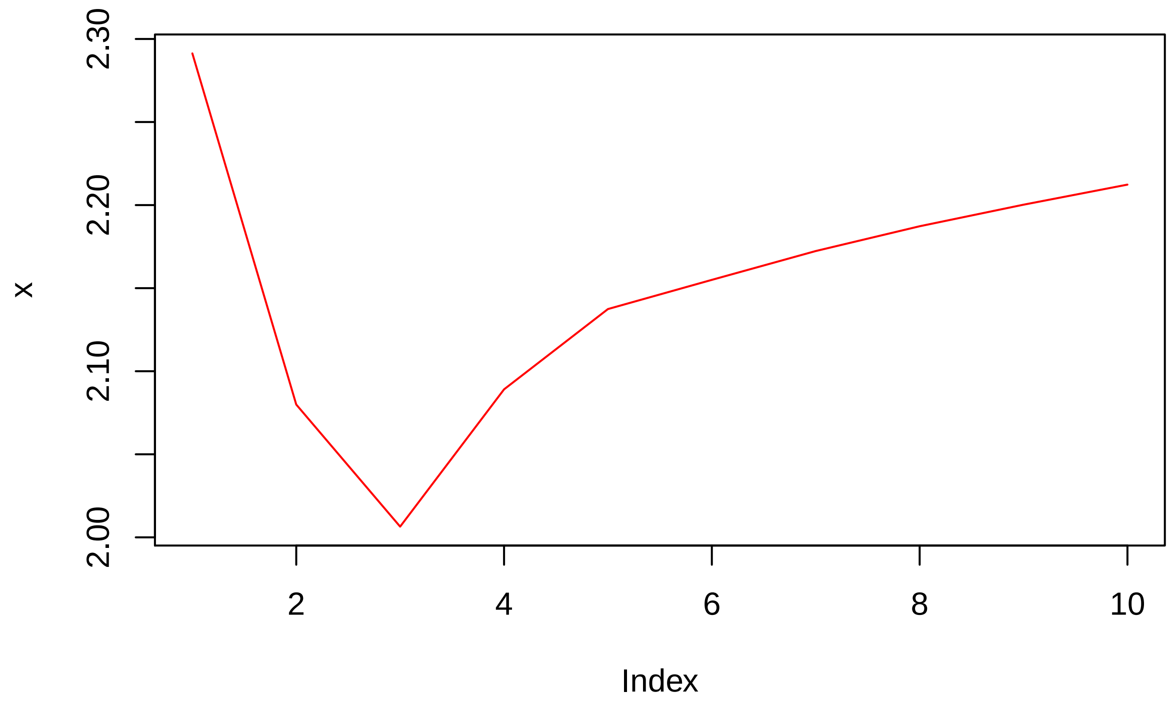


#KNN Model

```
train_labels <- train[,8]       ##DV in the training set
test_labels <- test_labels      ##DV in the test set
train_set <- train[,-8]         ##Only keep IV in the training set
test_set <- test                ##Only keep IV in the test set
```

##Best value of K

```
x <- 0
for (i in 1:10)
{
KNNmodel <- knn.reg(train =train_set , test = test_set, y = train_labels , k = i)
test_predKNN<- KNNmodel$pred
rmse <- sqrt(mean((test_labels-test_predKNN)^2))
x[i] <- rmse
}

plot(x, type="l", col="red")
```

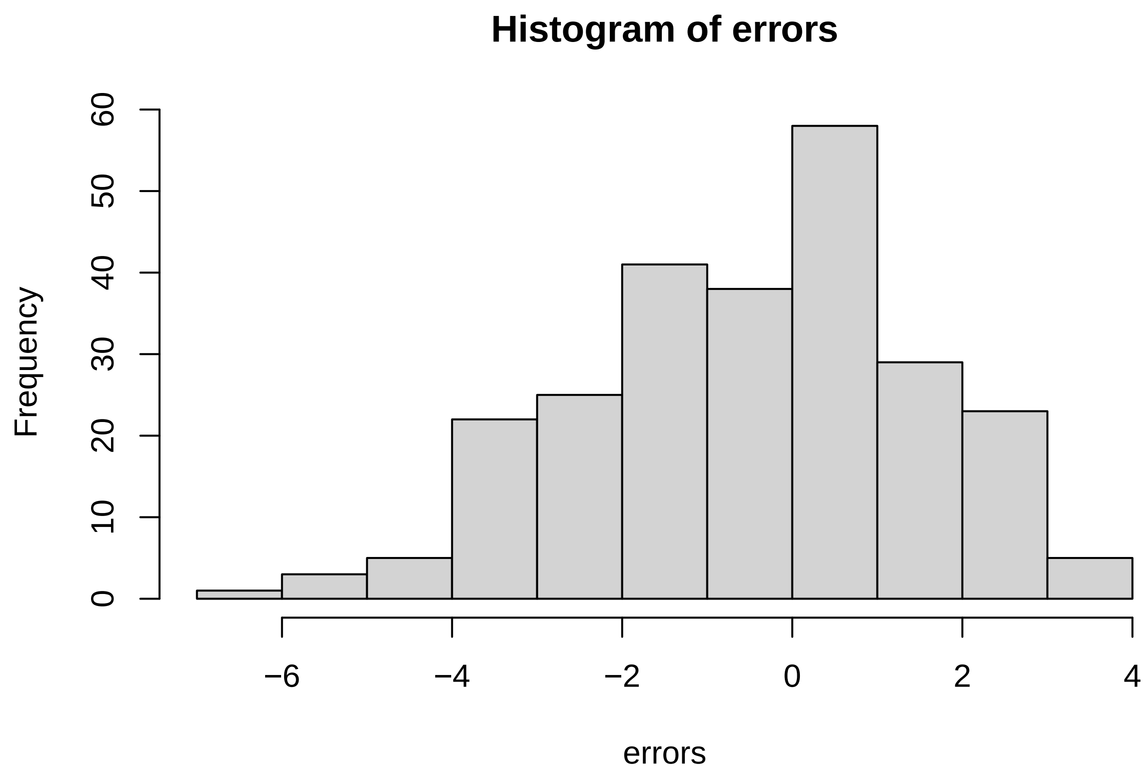


```r
bk<-which.min(x)
paste('Best K(by RMSE):', bk)
```

```
## [1] "Best K(by RMSE): 3"
```

##Model and distribution of errors

```r
set.seed(123)
KNNmodel <- knn.reg(train =train_set , test = test_set, y = train_labels , k =bk)
test_predKNN <- KNNmodel$pred
errors <- test_predKNN - test_labels
hist(errors)
```



##RMSE and cases with less than 25%, 10%, 5%, and 1% error

```
rmse <- sqrt(mean((test_labels-test_predKNN)^2))

rel_change <- abs(errors) / test_labels
pred25 <- sum((rel_change<0.25)=="TRUE")/nrow(test_set)   ## gives the count of those who are true on t
##OR pred25 <- table(rel_change<0.25)["TRUE"] / nrow(test_test)

pred10 <- sum((rel_change<0.10)=="TRUE")/nrow(test_set)  ## gives the count of those who are true on th

pred5 <- sum((rel_change<0.05)=="TRUE")/nrow(test_set)  ## gives the count of those who are true on the

pred1 <- sum((rel_change<0.01)=="TRUE")/nrow(test_set)  ## gives the count of those who are true on the

paste("RMSE:", rmse)
```

```
## [1] "RMSE: 2.00646946897918"
```

```
paste("Pred(25%):", pred25)
```

```
## [1] "Pred(25%): 1"
```

```
paste("Pred(10%):", pred10)
```

```
## [1] "Pred(10%): 1"
```

```
paste("Pred(5%):", pred5)
```

```
## [1] "Pred(5%): 0.944"
```

```
paste("Pred(1%):", pred1)
```

```
## [1] "Pred(1%): 0.244"
```

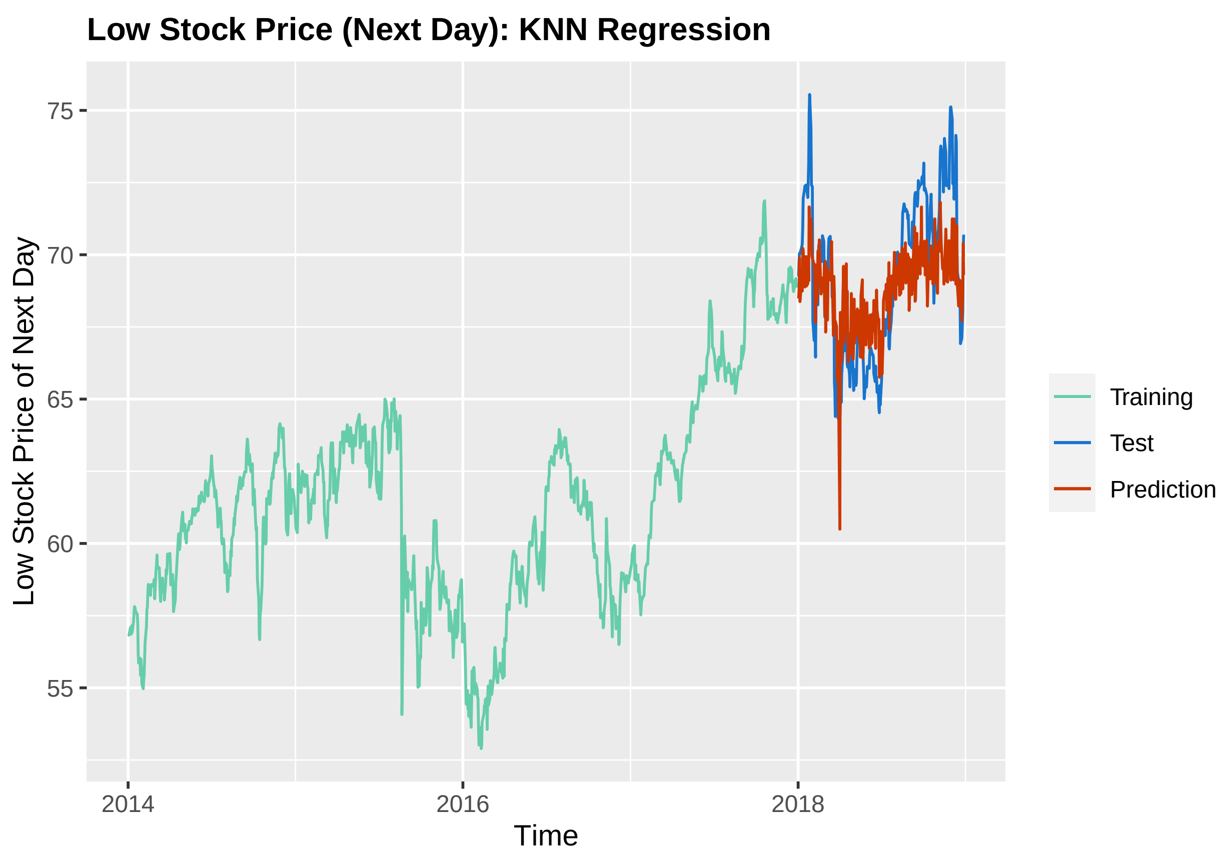
##Re-creating the time series dataset for low stock price (of next day)

```
data_ts<-data_n[,c(8,9)]
data_ts$date <- as.Date(data_ts$date)
```

```
train_ts<-data_ts[1:(cut-1),]
test_ts<-data_ts[cut:nrow(data_n),]
```

##Plot

```
test_ts$preds<- test_predKNN
ggplot(data = train_ts, aes(x = as.Date(date), y = low_price_next_day,color = "aquamarine3"))+
  geom_line()+
  geom_line(data = test_ts, aes(x = as.Date(date), y = low_price_next_day ,color="dodgerblue3"))+
  geom_line(data = test_ts, aes(x = as.Date(date), y = preds,color="orangered3"))+
  scale_color_identity(name = " ",
                       breaks = c("aquamarine3", "dodgerblue3", "orangered3"),
                       labels = c("Training", "Test", "Prediction"),
                       guide = "legend")+
  xlab("Time")+
  ylab("Low Stock Price of Next Day")+
  ggtitle("Low Stock Price (Next Day): KNN Regression") +
  theme(plot.title = element_text(size=12, face = "bold"))
```



#Random Forest Model ## Best number of trees

```
set.seed(123)
library(randomForest)
```

```
## randomForest 4.6-14
```

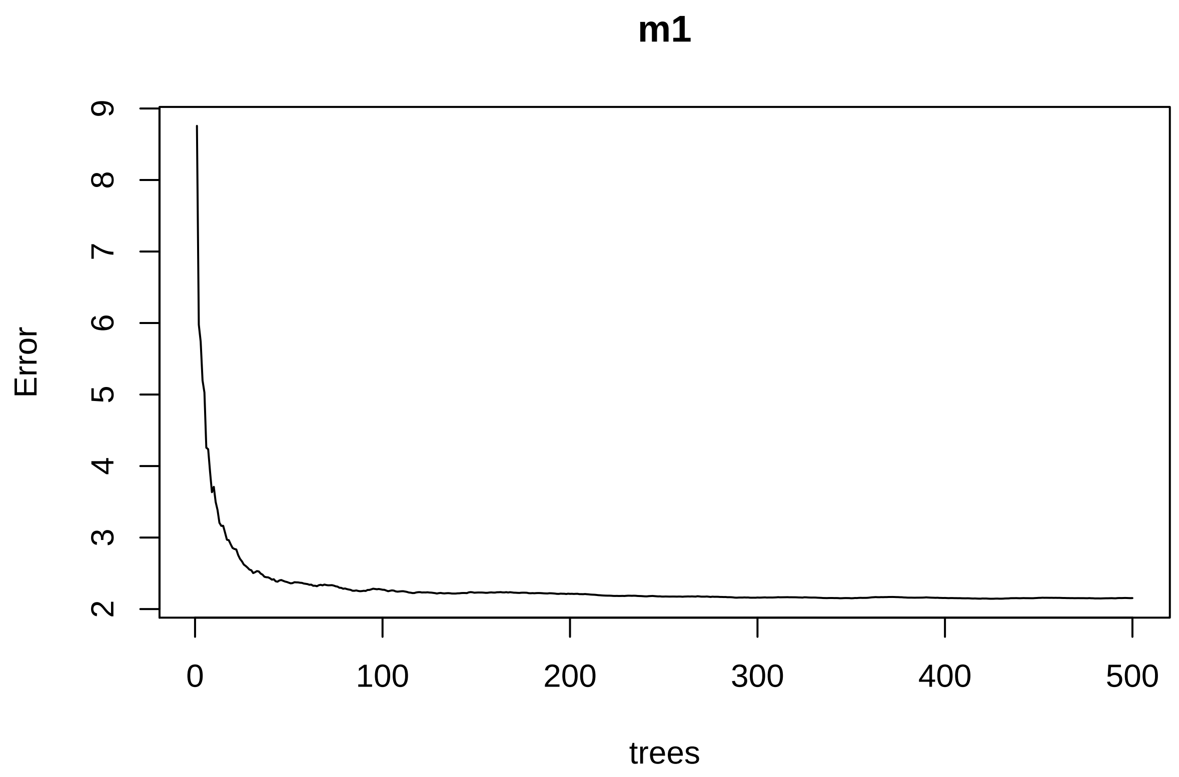
```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
m1 <- randomForest(formula = low_price_next_day ~ ., data = train)

plot(m1)
```

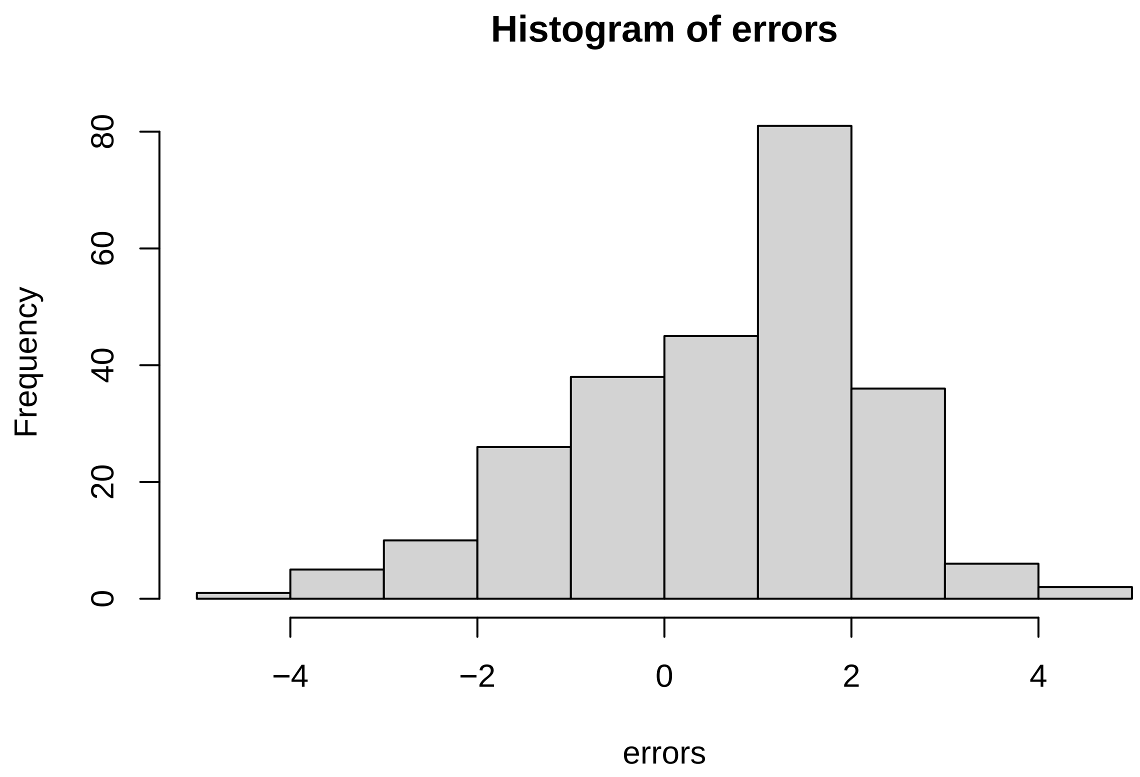


```
bntrees<-which.min(m1$mse)
paste("Best number of tree:", bntrees)
```

```
## [1] "Best number of tree: 425"
```

##Model and distribution of errors

```
set.seed(123)
RFmodel <- train(low_price_next_day ~ ., data= train, method="rf", ntree=bntrees, trControl = ctrl)

test_predRF <- predict(RFmodel, test)
errors <- test_predRF - test_labels
hist(errors)
```



##RMSE and cases with less than 25%, 10%, 5%, and 1% error

```
rmse <- sqrt(mean((test_labels-test_predRF)^2))

rel_change <- abs(errors) / test_labels
pred25 <- sum((rel_change<0.25)=="TRUE")/nrow(test)   ## gives the count of those who are true on the c
##OR pred25 <- table(rel_change<0.25)["TRUE"] / nrow(test)

pred10 <- sum((rel_change<0.10)=="TRUE")/nrow(test)  ## gives the count of those who are true on the co

pred5 <- sum((rel_change<0.05)=="TRUE")/nrow(test)  ## gives the count of those who are true on the con

pred1 <- sum((rel_change<0.01)=="TRUE")/nrow(test)  ## gives the count of those who are true on the con

paste("RMSE:", rmse)
```

```
## [1] "RMSE: 1.67819962929625"
```

```
paste("Pred(25%):", pred25)
```

```
## [1] "Pred(25%): 1"
```

```
paste("Pred(10%):", pred10)
```

```
## [1] "Pred(10%): 1"
```

```
paste("Pred(5%):", pred5)
```

```
## [1] "Pred(5%): 0.972"
```

```
paste("Pred(1%):", pred1)
```

```
## [1] "Pred(1%): 0.224"
```

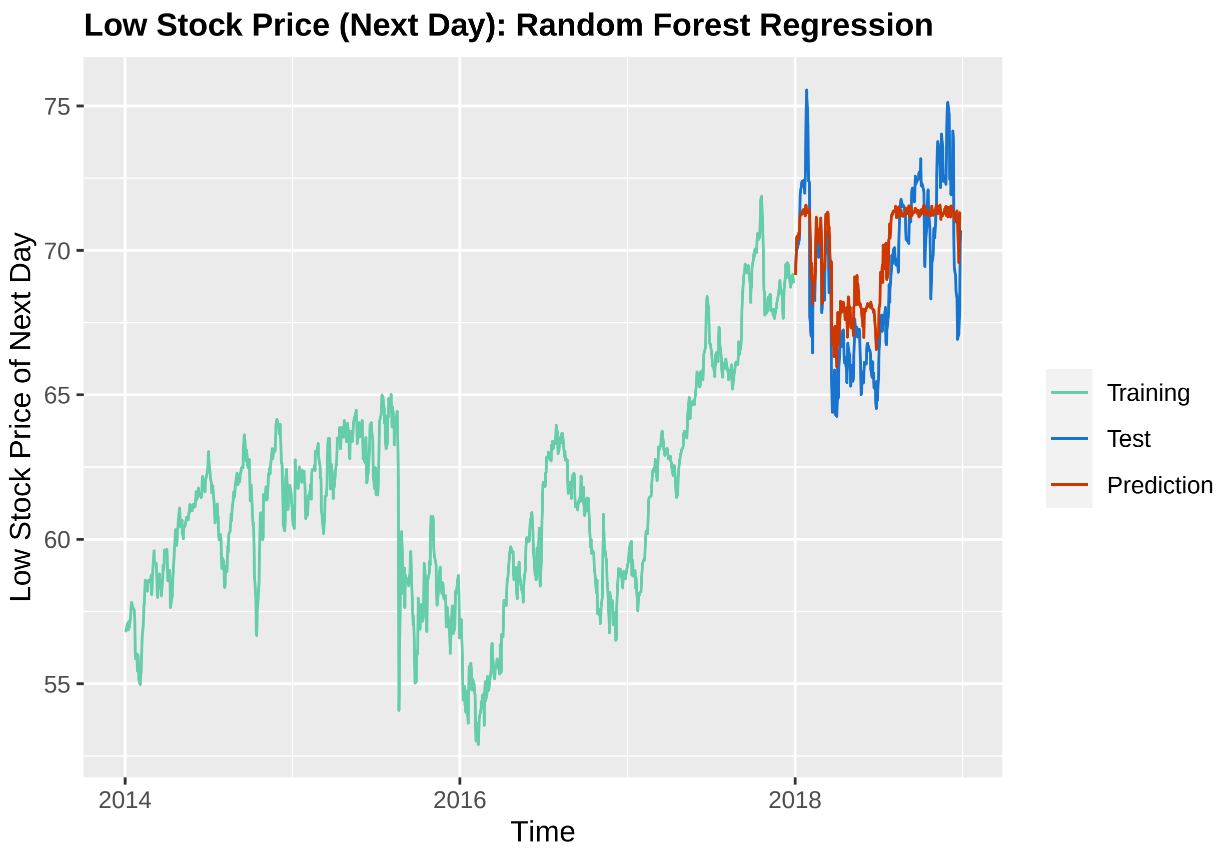
##Re-creating the time series dataset for low stock price (of next day)

```
data_ts<-data_n[,c(8,9)]
data_ts$date <- as.Date(data_ts$date)
```

```
train_ts<-data_ts[1:(cut-1),]
test_ts<-data_ts[cut:nrow(data_ts),]
```

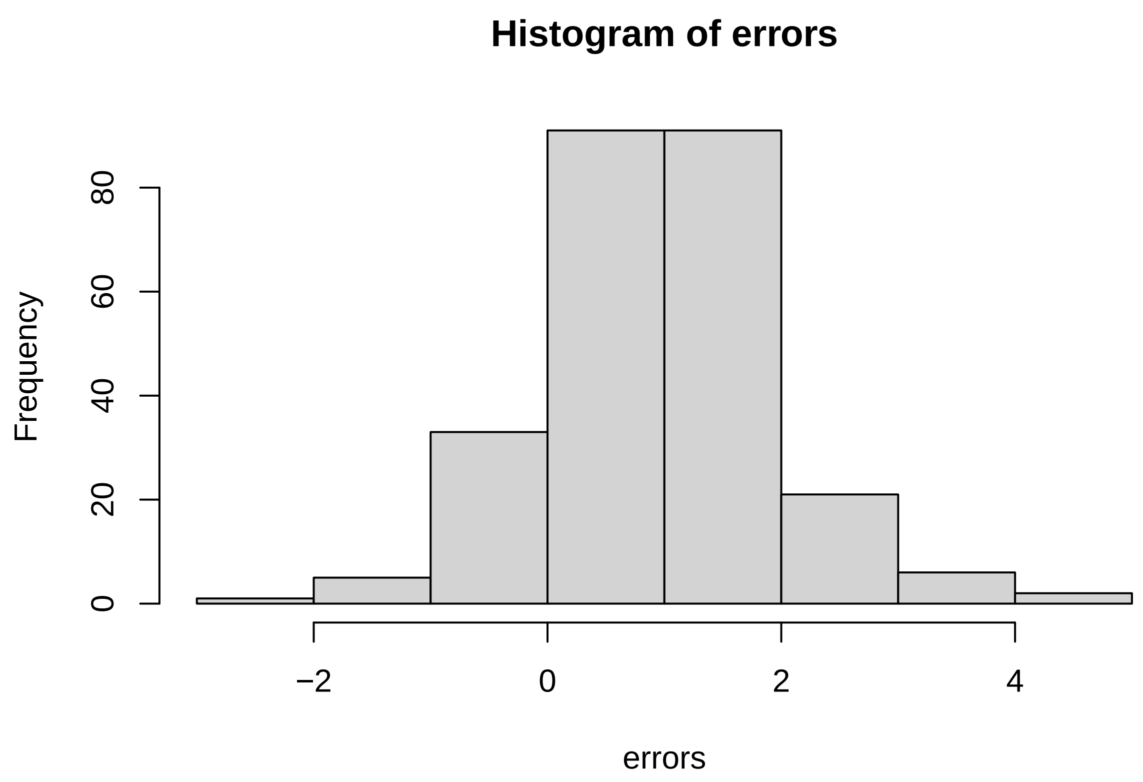
##Plot

```
test_ts$preds<- test_predRF
ggplot(data = train_ts, aes(x = as.Date(date), y = low_price_next_day,color = "aquamarine3"))+
  geom_line()+
  geom_line(data = test_ts, aes(x = as.Date(date), y = low_price_next_day ,color="dodgerblue3"))+
  geom_line(data = test_ts, aes(x = as.Date(date), y = preds,color="orangered3"))+
  scale_color_identity(name = " ",
                       breaks = c("aquamarine3", "dodgerblue3", "orangered3"),
                       labels = c("Training", "Test", "Prediction"),
                       guide = "legend")+
  xlab("Time")+
  ylab("Low Stock Price of Next Day")+
  ggtitle("Low Stock Price (Next Day): Random Forest Regression") +
  theme(plot.title = element_text(size=12, face = "bold"))
```



#SVM Regression ##Model and distribution of errors

```
set.seed(123)
SVMmodel <- train(low_price_next_day ~ ., data= train, method="svmPoly", trControl = ctrl)

test_predSVM <- predict(SVMmodel, test)
errors <- test_predSVM - test_labels
hist(errors)
```



##RMSE and cases with less than 25%, 10%, 5%, and 1% error

```r
rmse <- sqrt(mean((test_labels-test_predSVM)^2))

rel_change <- abs(errors) / test_labels
pred25 <- sum((rel_change<0.25)=="TRUE")/nrow(test)    ## gives the count of those who are true on the c
##OR pred25 <- table(rel_change<0.25)["TRUE"] / nrow(test)

pred10 <- sum((rel_change<0.10)=="TRUE")/nrow(test)  ## gives the count of those who are true on the co

pred5 <- sum((rel_change<0.05)=="TRUE")/nrow(test)  ## gives the count of those who are true on the con

pred1 <- sum((rel_change<0.01)=="TRUE")/nrow(test)  ## gives the count of those who are true on the con

paste("RMSE:", rmse)
```

```
## [1] "RMSE: 1.38080394004463"
```

```r
paste("Pred(25%):", pred25)
```

```
## [1] "Pred(25%): 1"
```

```r
paste("Pred(10%):", pred10)
```

```
## [1] "Pred(10%): 1"
```

```
paste("Pred(5%):", pred5)
```

```
## [1] "Pred(5%): 0.988"
```

```
paste("Pred(1%):", pred1)
```

```
## [1] "Pred(1%): 0.336"
```

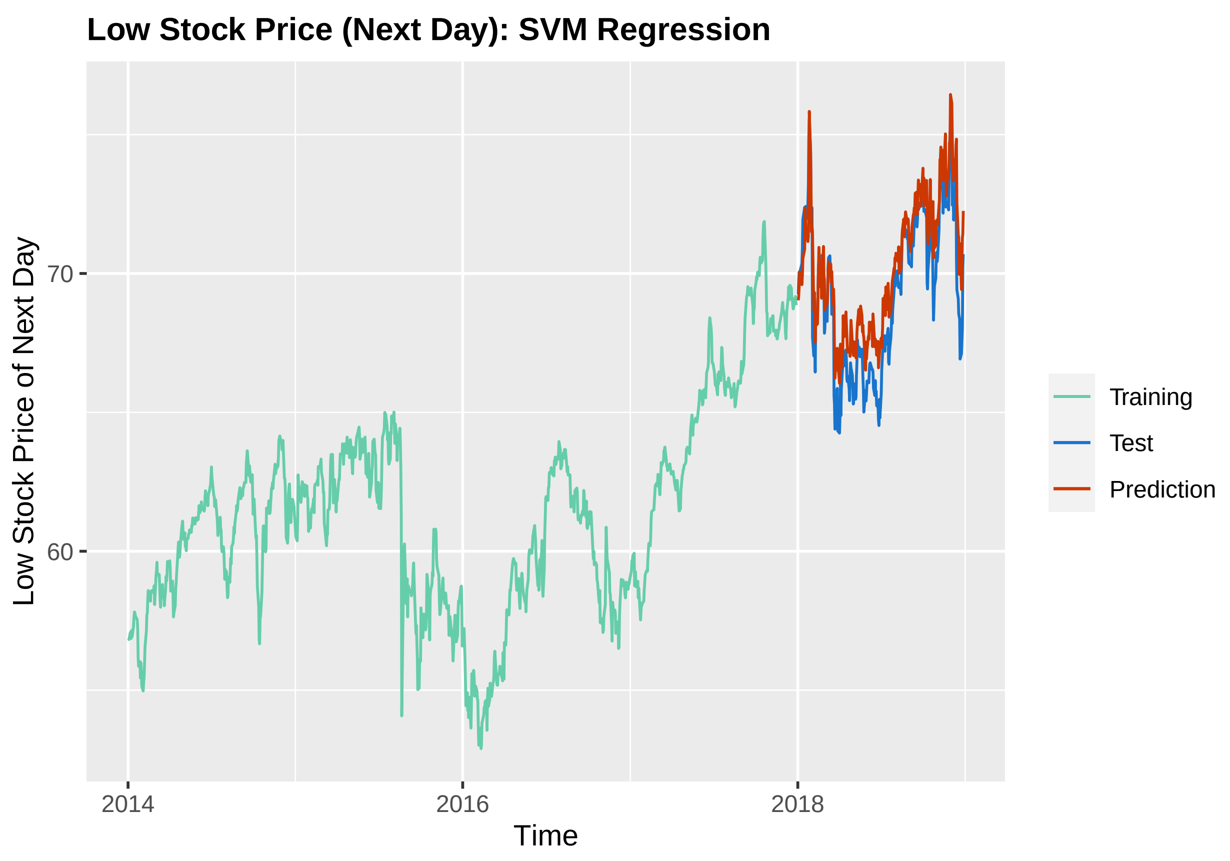
##Re-creating the time series dataset for low stock price (of next day)

```
data_ts<-data_n[,c(8,9)]
data_ts$date <- as.Date(data_ts$date)
```

```
train_ts<-data_ts[1:(cut-1),]
test_ts<-data_ts[cut:nrow(data_ts),]
```

##Plot

```
test_ts$preds<- test_predSVM
ggplot(data = train_ts, aes(x = as.Date(date), y = low_price_next_day,color = "aquamarine3"))+
  geom_line()+
  geom_line(data = test_ts, aes(x = as.Date(date), y = low_price_next_day ,color="dodgerblue3"))+
  geom_line(data = test_ts, aes(x = as.Date(date), y = preds,color="orangered3"))+
  scale_color_identity(name = " ",
                       breaks = c("aquamarine3", "dodgerblue3", "orangered3"),
                       labels = c("Training", "Test", "Prediction"),
                       guide = "legend")+
  xlab("Time")+
  ylab("Low Stock Price of Next Day")+
  ggtitle("Low Stock Price (Next Day): SVM Regression") +
  theme(plot.title = element_text(size=12, face = "bold"))
```



#Ensemble Model

```
library(caretEnsemble)
```

```
##
## Attaching package: 'caretEnsemble'

## The following object is masked from 'package:ggplot2':
##
##     autoplot
```

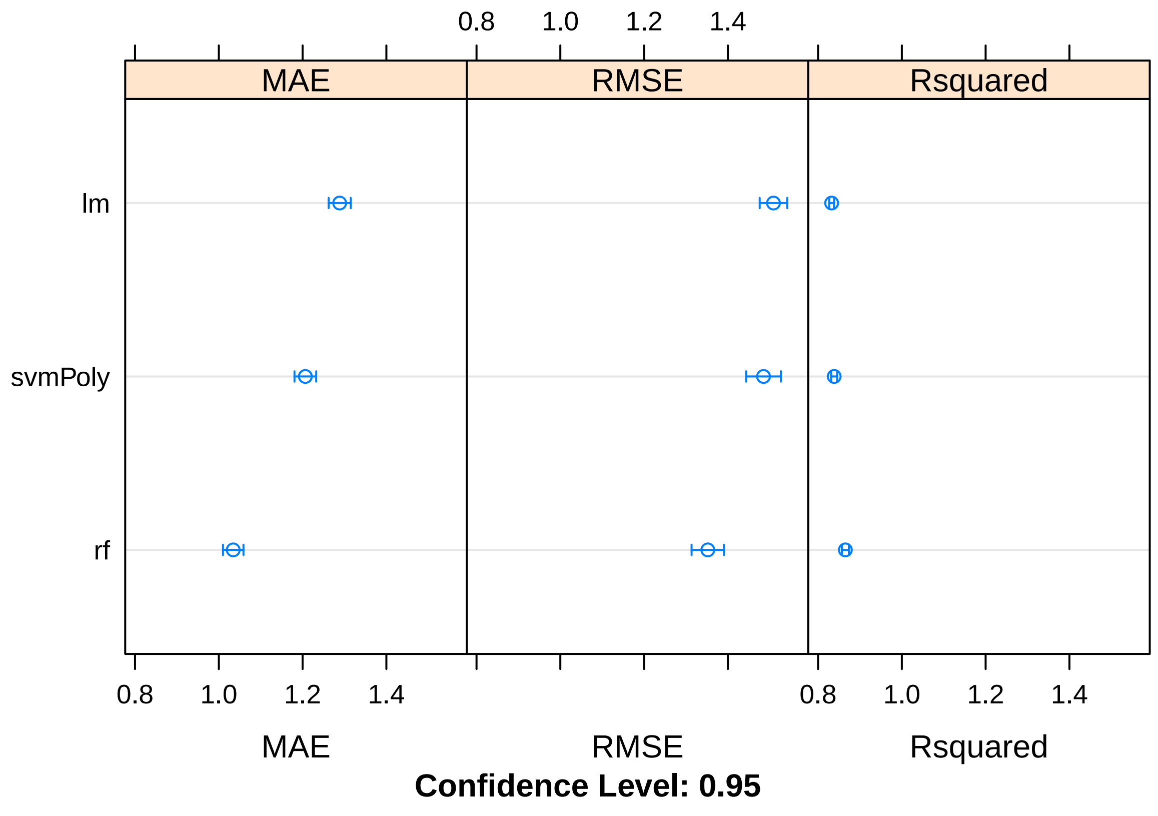
```
control <- trainControl(method="repeatedcv", number = 5, repeats=3, savePredictions="final")
algorithmList <- c('rf', 'svmPoly', 'lm')
set.seed(123)
models_ens <- caretList(low_price_next_day~., data= train, trControl=control, methodList=algorithmList)
```

```
## Warning in trControlCheck(x = trControl, y = target): indexes not defined in
## trControl. Attempting to set them ourselves, so each model in the ensemble will
## have the same resampling indexes.
```

```
set.seed(123)
results <- resamples(models_ens)
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: rf, svmPoly, lm
## Number of resamples: 15
##
## MAE
##              Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## rf      0.9403841 1.025694 1.051277 1.034433 1.055541 1.111234    0
## svmPoly 1.1378538 1.175047 1.191188 1.206576 1.246234 1.273159    0
## lm      1.2081997 1.269029 1.279424 1.288428 1.313243 1.365092    0
##
## RMSE
##             Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## rf      1.224972 1.317547 1.372771 1.352007 1.394982 1.466288    0
## svmPoly 1.361826 1.440426 1.462782 1.485002 1.545836 1.619354    0
## lm      1.389114 1.472643 1.514677 1.508817 1.559238 1.591784    0
##
## Rsquared
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## rf      0.8328952 0.8583957 0.8633991 0.8651661 0.8720154 0.8892301    0
## svmPoly 0.8164145 0.8265547 0.8409535 0.8384510 0.8488101 0.8570054    0
## lm      0.8127614 0.8270026 0.8328227 0.8323375 0.8411912 0.8466904    0
```

```r
dotplot(results)
```



```r
modelCor(results)
```

```
##                rf   svmPoly        lm
## rf      1.0000000 0.6908257 0.4090872
## svmPoly 0.6908257 1.0000000 0.8408880
## lm      0.4090872 0.8408880 1.0000000
```

SVM is highly correlated with random forest(RF) and linear regression(LR/LM). Hence, we will keep only random forest and linear regression.

##only keep RF and LR/LM

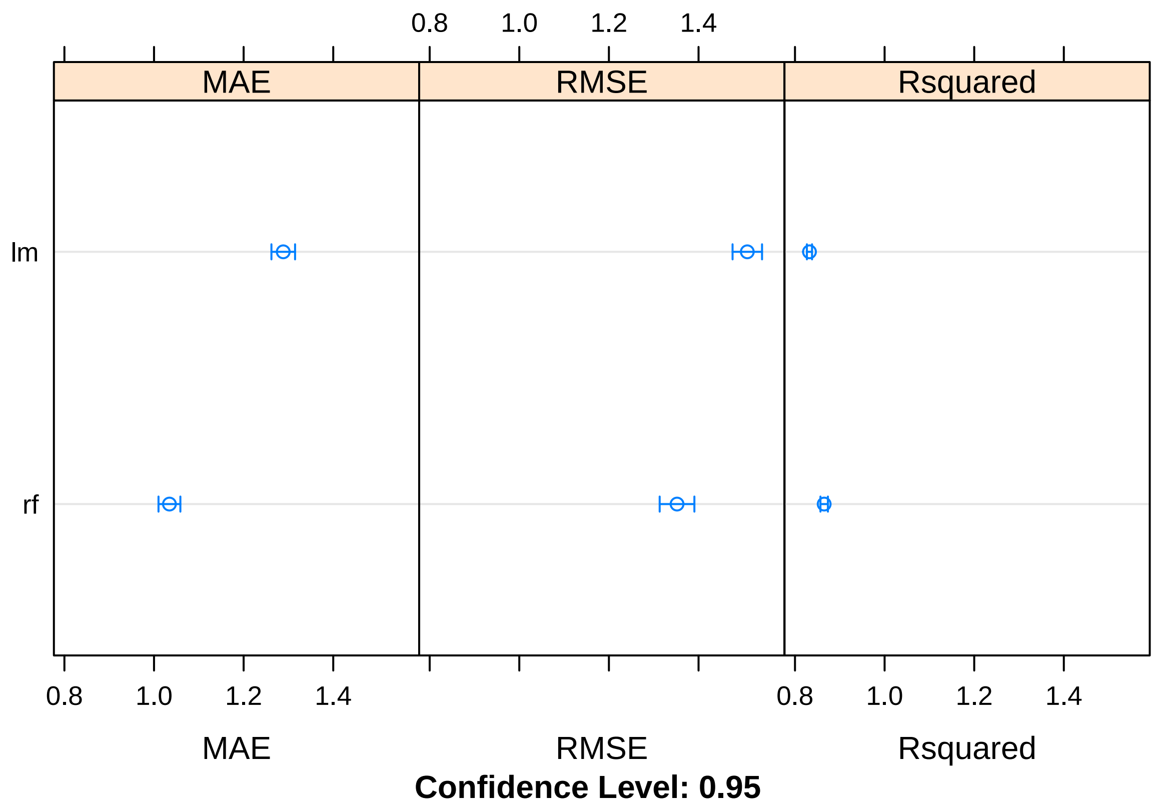
```r
library(caretEnsemble)
control <- trainControl(method="repeatedcv", number = 5, repeats=3, savePredictions="final")
algorithmList <- c('rf', 'lm')
set.seed(123)
models_ens2 <- caretList(low_price_next_day~., data= train, trControl=control, methodList=algorithmList
```

```
## Warning in trControlCheck(x = trControl, y = target): indexes not defined in
## trControl. Attempting to set them ourselves, so each model in the ensemble will
## have the same resampling indexes.
```

```r
set.seed(123)
results2 <- resamples(models_ens2)
summary(results2)
```

```
##
## Call:
## summary.resamples(object = results2)
##
## Models: rf, lm
## Number of resamples: 15
##
## MAE
##         Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## rf 0.9403841 1.025694 1.051277 1.034433 1.055541 1.111234    0
## lm 1.2081997 1.269029 1.279424 1.288428 1.313243 1.365092    0
##
## RMSE
##        Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## rf 1.224972 1.317547 1.372771 1.352007 1.394982 1.466288    0
## lm 1.389114 1.472643 1.514677 1.508817 1.559238 1.591784    0
##
## Rsquared
##         Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## rf 0.8328952 0.8583957 0.8633991 0.8651661 0.8720154 0.8892301    0
## lm 0.8127614 0.8270026 0.8328227 0.8323375 0.8411912 0.8466904    0
```

```r
dotplot(results2)
```



```
modelCor(results2)
```

```
##           rf        lm
## rf 1.0000000 0.4090872
## lm 0.4090872 1.0000000
```

##Combine predictions of the final models (RF and LR/LM) used using stack with RF

```
stackControl <- trainControl(method="repeatedcv", number=5, repeats=3, savePredictions=TRUE)
set.seed(123)
stack.rf <- caretStack(models_ens2, method="rf", metric="RMSE", trControl=stackControl)
```

```
## note: only 1 unique complexity parameters in default grid. Truncating the grid to 1 .
```

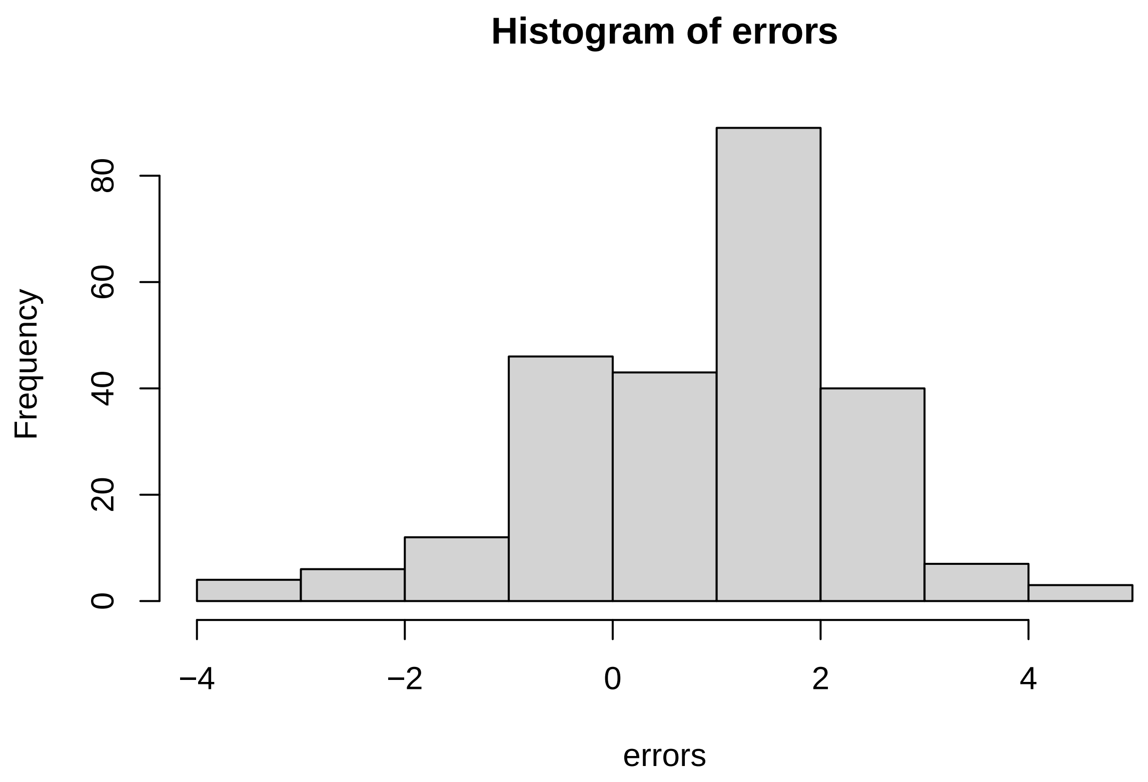
```
print(stack.rf)  ##ensemble model
```

```
## A rf ensemble of 2 base models: rf, lm
##
## Ensemble results:
## Random Forest
##
## 3021 samples
##    2 predictor
##
```

```
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 2416, 2417, 2416, 2417, 2418, 2417, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   1.385207  0.8580494  1.025302
##
## Tuning parameter 'mtry' was held constant at a value of 2
```

##predict stack.rf on the test dataset and plot its errors

```
test_predEM <- predict(stack.rf , test)
errors <- test_predEM - test_labels
hist(errors)
```



##RMSE and cases with less than 25%, 10%, 5%, and 1% error

```
rmse <- sqrt(mean((test_labels-test_predEM)^2))

rel_change <- abs(errors) / test_labels
pred25 <- sum((rel_change<0.25)=="TRUE")/nrow(test)    ## gives the count of those who are true on the c
##OR pred25 <- table(rel_change<0.25)["TRUE"] / nrow(test)

pred10 <- sum((rel_change<0.10)=="TRUE")/nrow(test)   ## gives the count of those who are true on the co
```

```
pred5 <- sum((rel_change<0.05)=="TRUE")/nrow(test)  ## gives the count of those who are true on the con

pred1 <- sum((rel_change<0.01)=="TRUE")/nrow(test)  ## gives the count of those who are true on the con

paste("RMSE:", rmse)
```

```
## [1] "RMSE: 1.6988030303098"
```

```
paste("Pred(25%):", pred25)
```

```
## [1] "Pred(25%): 1"
```

```
paste("Pred(10%):", pred10)
```

```
## [1] "Pred(10%): 1"
```

```
paste("Pred(5%):", pred5)
```

```
## [1] "Pred(5%): 0.976"
```

```
paste("Pred(1%):", pred1)
```

```
## [1] "Pred(1%): 0.284"
```

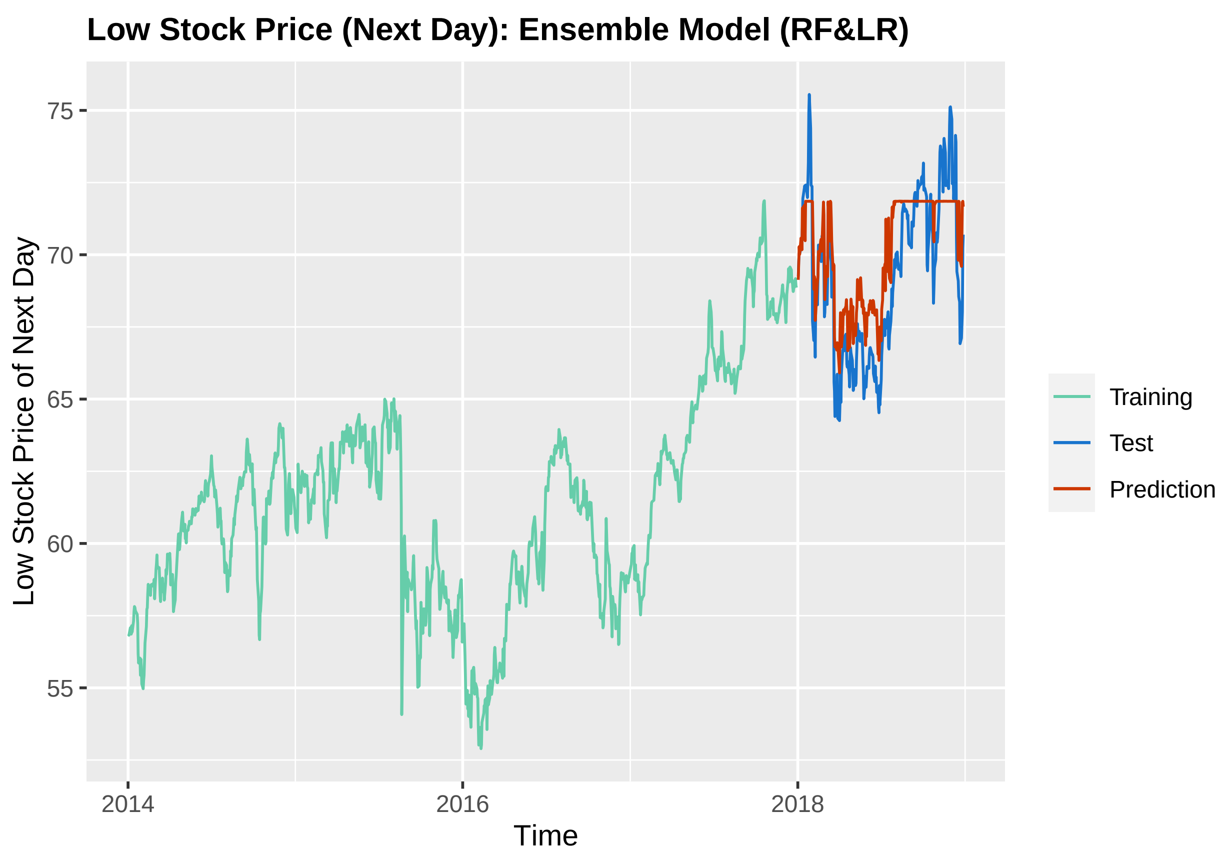
##Re-creating the time series dataset for low stock price (of next day)

```
data_ts<-data_n[,c(8,9)]
data_ts$date <- as.Date(data_ts$date)
```

```
train_ts<-data_ts[1:(cut-1),]
test_ts<-data_ts[cut:nrow(data_ts),]
```

##Plot

```
test_ts$preds<- test_predEM
ggplot(data = train_ts, aes(x = as.Date(date), y = low_price_next_day,color = "aquamarine3"))+
  geom_line()+
  geom_line(data = test_ts, aes(x = as.Date(date), y = low_price_next_day ,color="dodgerblue3"))+
  geom_line(data = test_ts, aes(x = as.Date(date), y = preds,color="orangered3"))+
  scale_color_identity(name = " ",
                       breaks = c("aquamarine3", "dodgerblue3", "orangered3"),
                       labels = c("Training", "Test", "Prediction"),
                       guide = "legend")+
  xlab("Time")+
  ylab("Low Stock Price of Next Day")+
  ggtitle("Low Stock Price (Next Day): Ensemble Model (RF&LR)") +
  theme(plot.title = element_text(size=12, face = "bold"))
```



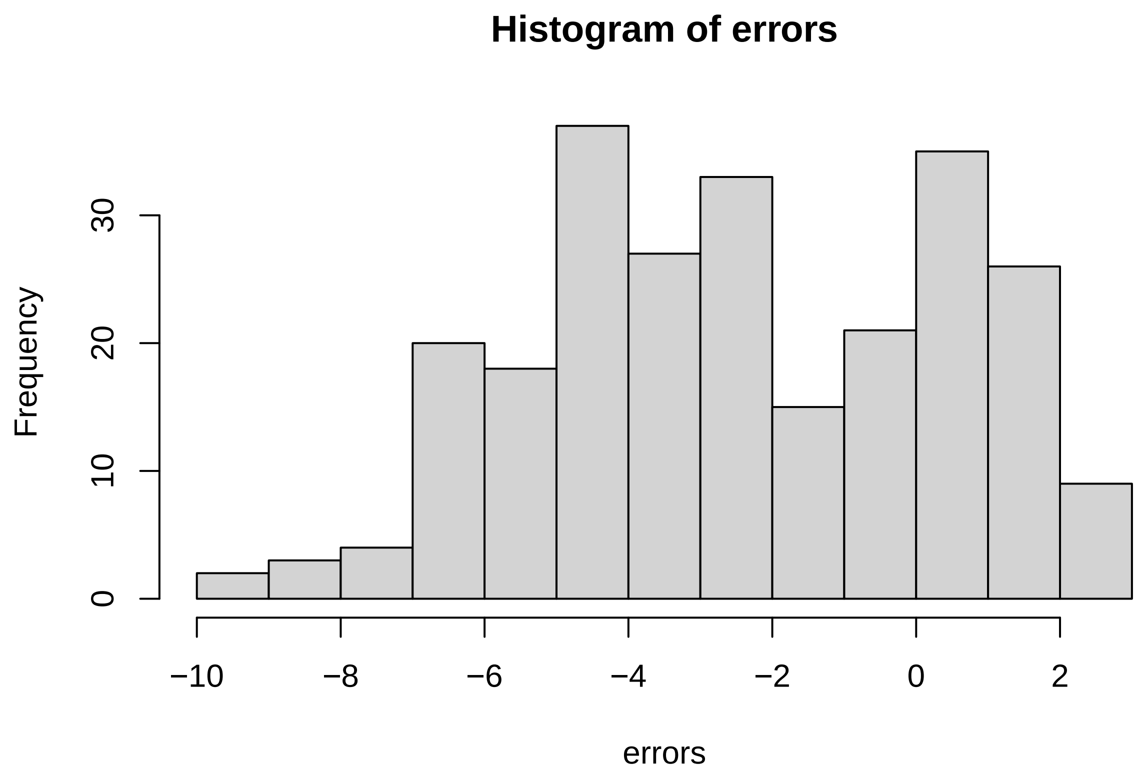
#Simple Moving Average (of last 250 low stock price) ##Re-creating the time series dataset for low stock price

```r
data_ts<-data[,c(1,13)]
data_ts$date = as.Date(data_ts$date)
```

```r
train_ts<-data_ts[1:(cut-1),]
test_ts<-data_ts[cut:nrow(data_ts),]
```

##Model and distribution of errors

```r
index<-nrow(test_ts)
preds = 0
for (i in 1:index)
  {
  a = sum(train_ts$low_mean[(nrow(train_ts)-index+i):(nrow(train_ts))]) + sum(preds)
  b = a/index
  preds[i] <- b
}
errors <- preds - test_ts$low_mean
hist(errors)
```



##RMSE and cases with less than 25%, 10%, 5%, and 1% error

```
rmse <- sqrt(mean((test_ts$low_mean - preds)^2))

rel_change <- abs(errors) / test_ts$low_mean
pred25 <- sum((rel_change<0.25)=="TRUE")/nrow(test_ts)    ## gives the count of those who are true on th
##OR pred25 <- table(rel_change<0.25)["TRUE"] / nrow(test_ts)

pred10 <- sum((rel_change<0.10)=="TRUE")/nrow(test_ts)  ## gives the count of those who are true on the

pred5 <- sum((rel_change<0.05)=="TRUE")/nrow(test_ts)  ## gives the count of those who are true on the

pred1 <- sum((rel_change<0.01)=="TRUE")/nrow(test_ts)  ## gives the count of those who are true on the

paste("RMSE:", rmse)
```

```
## [1] "RMSE: 3.69441010263325"
```

```
paste("Pred(25%):", pred25)
```

```
## [1] "Pred(25%): 1"
```

```
paste("Pred(10%):", pred10)
```

```
## [1] "Pred(10%): 0.972"
```

```
paste("Pred(5%):", pred5)
```

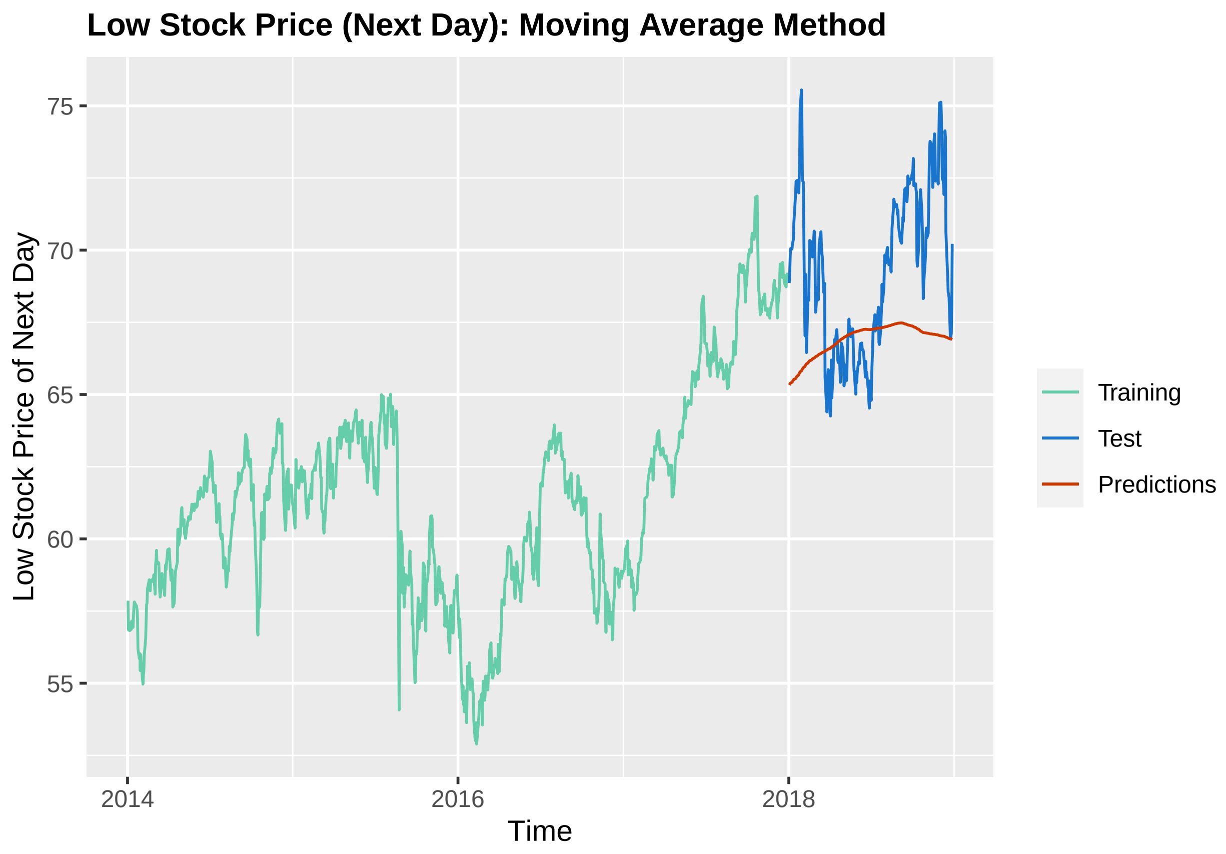
```
## [1] "Pred(5%): 0.604"
```

```
paste("Pred(1%):", pred1)
```

```
## [1] "Pred(1%): 0.164"
```

##Plot

```
test_ts$preds <- preds
ggplot(data = train_ts, aes(x = as.Date(date), y = low_mean,color = "aquamarine3"))+
  geom_line()+
  geom_line(data = test_ts, aes(x = as.Date(date), y = low_mean ,color="dodgerblue3"))+
  geom_line(data = test_ts, aes(x = as.Date(date), y = preds ,color="orangered3"))+
  scale_color_identity(name = " ",
                       breaks = c("aquamarine3", "dodgerblue3", "orangered3"),
                       labels = c("Training", "Test", "Predictions"),
                       guide = "legend")+
  xlab("Time")+
  ylab("Low Stock Price of Next Day")+
  ggtitle("Low Stock Price (Next Day): Moving Average Method") +
  theme(plot.title = element_text(size=12, face = "bold"))
```



#Auto-ARIMA ##Re-creating the time series dataset for low stock price

```r
data_ts<-data[,c(1,13)]
data_ts$date = as.Date(data_ts$date)
```

```r
train_ts<-data_ts[1:(cut-1),]
test_ts<-data_ts[cut:nrow(data_ts),]
```

##Model and distribution of errors

```r
#install.packages("forecast")
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

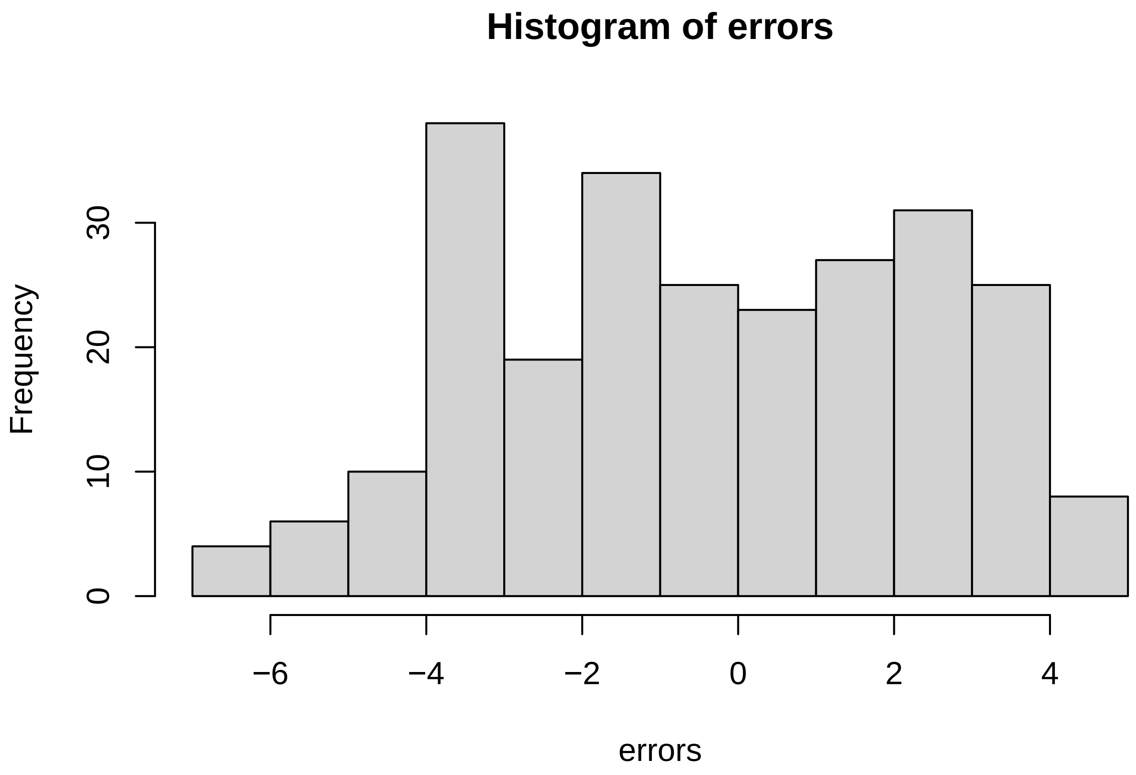
```
##
## Attaching package: 'forecast'
```

```
## The following object is masked from 'package:caretEnsemble':
##
##     autoplot
```

```
## The following object is masked from 'package:rcompanion':
##
##     accuracy
```

```r
ARIMAmodel <- auto.arima(train_ts$low_mean,
                  max.p = 5,
                  max.q = 5,
                  max.P = 2,
                  max.Q = 2,
                  max.order = 5,
                  max.d = 2,
                  max.D = 1,
                  start.p = 2,
                  start.q = 2,
                  start.P = 1,
                  start.Q = 1,
                  stationary = FALSE,
                  seasonal = TRUE,
                  ic = c("aicc", "aic", "bic"),
                  stepwise = TRUE,
                  nmodels = 100,
                  trace = FALSE,
                  method = NULL,
                  truncate = NULL,
                  test = c("kpss", "adf", "pp"),
                  seasonal.test = c("seas", "ocsb", "hegy", "ch"),
                  allowdrift = TRUE,
                  lambda = NULL,
                  biasadj = FALSE,
                   parallel = FALSE)

test_predARIMA<-predict(ARIMAmodel,n.ahead = nrow(test_ts))
```

```
test_predARIMA<- test_predARIMA$pred
errors <- test_predARIMA - test_ts$low_mean
hist(errors)
```



##RMSE and cases with less than 25%, 10%, 5%, and 1% error

```
rmse <- sqrt(mean((test_ts$low_mean - test_predARIMA)^2))

rel_change <- abs(errors) / test_ts$low_mean
pred25 <- sum((rel_change<0.25)=="TRUE")/nrow(test_ts)    ## gives the count of those who are true on th
##OR pred25 <- table(rel_change<0.25)["TRUE"] / nrow(test_ts)

pred10 <- sum((rel_change<0.10)=="TRUE")/nrow(test_ts)   ## gives the count of those who are true on the

pred5 <- sum((rel_change<0.05)=="TRUE")/nrow(test_ts)   ## gives the count of those who are true on the

pred1 <- sum((rel_change<0.01)=="TRUE")/nrow(test_ts)   ## gives the count of those who are true on the

paste("RMSE:", rmse)
```

```
## [1] "RMSE: 2.76106116666636"
```

```r
paste("Pred(25%):", pred25)
```

```
## [1] "Pred(25%): 1"
```

```r
paste("Pred(10%):", pred10)
```

```
## [1] "Pred(10%): 1"
```

```r
paste("Pred(5%):", pred5)
```

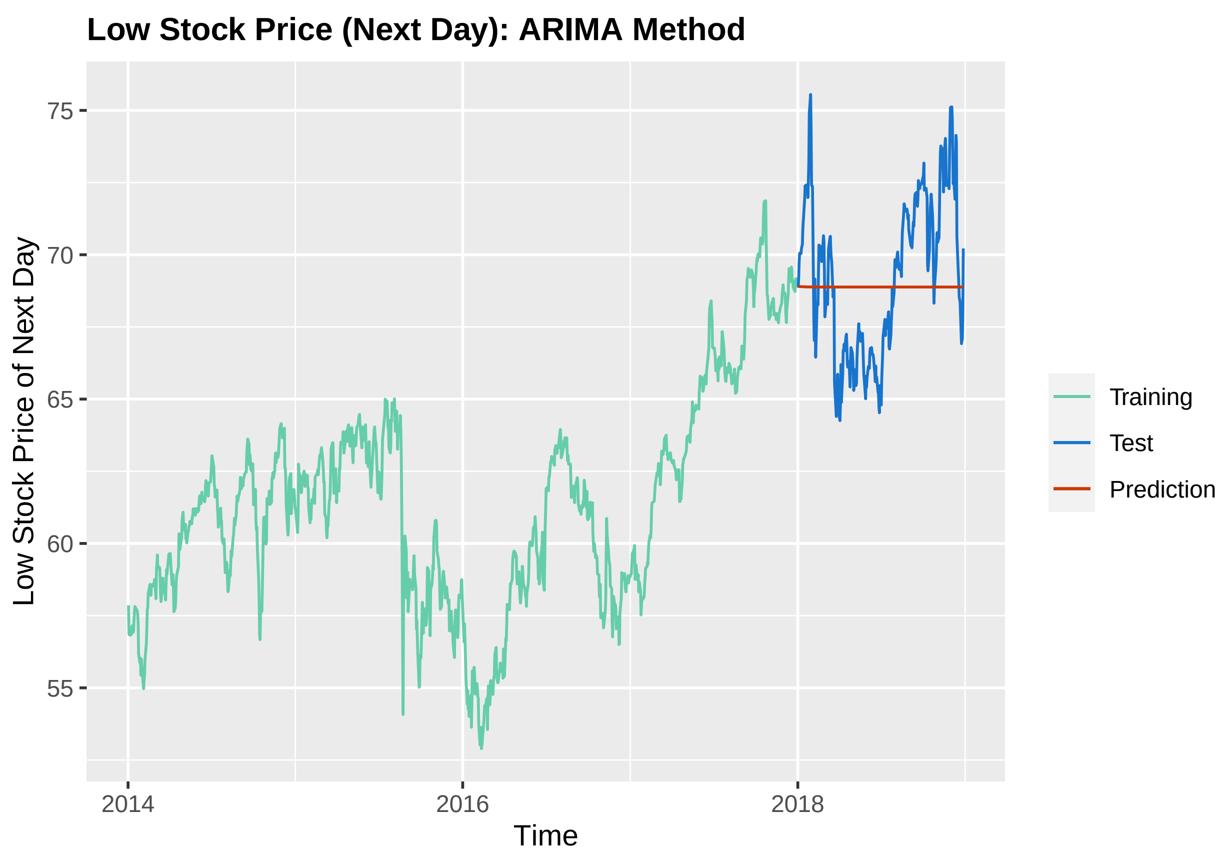
```
## [1] "Pred(5%): 0.796"
```

```r
paste("Pred(1%):", pred1)
```

```
## [1] "Pred(1%): 0.136"
```

##Plot

```r
test_ts$preds <- test_predARIMA
ggplot(data = train_ts, aes(x = as.Date(date), y = low_mean,color = "aquamarine3"))+
  geom_line()+
  geom_line(data = test_ts, aes(x = as.Date(date), y = low_mean ,color="dodgerblue3"))+
  geom_line(data = test_ts, aes(x = as.Date(date), y = preds,color="orangered3"))+
  scale_color_identity(name = " ",
                       breaks = c("aquamarine3", "dodgerblue3", "orangered3"),
                       labels = c("Training", "Test", "Prediction"),
                       guide = "legend")+
  xlab("Time")+
  ylab("Low Stock Price of Next Day")+
  ggtitle("Low Stock Price (Next Day): ARIMA Method") +
  theme(plot.title = element_text(size=12, face = "bold"))
```



One might ask why we cannot just average out the low stock price of the previous 2, 3, 5 days and go with it rather than engage in a "complex ML process". Averaging out the low stock values of the past 5 previous days seems reasonable and fair since the stock market operates only on weekdays (except holidays). Let's try and see.

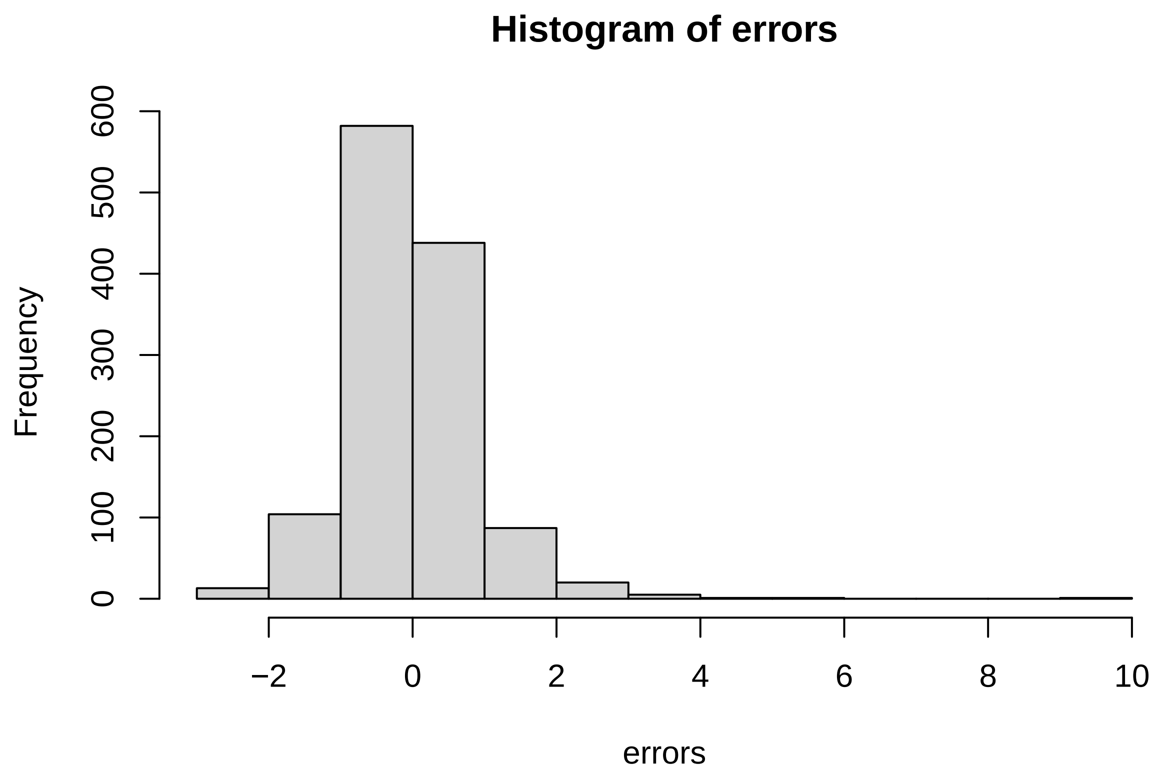
#Average of Previous '5 Days' (of low stock price to predict next low stock price) Method ##Re-creating the time series dataset for low stock price

```
data_ts<-data[,c(1,13)]
data_ts$date = as.Date(data_ts$date)
```

```
train_ts<-data_ts
test_ts<-data_ts[6:nrow(data_ts),]
```

##Model and distribution of errors

```
index<-nrow(test_ts)
preds = 0
for (i in 1:index)
  {
  a = sum(train_ts$low_mean[i:(i+4)])
  b = a/5
  preds[i] <- b
}
errors <- preds - test_ts$low_mean
hist(errors)
```



##RMSE and cases with less than 25%, 10%, 5%, and 1% error

```
rmse <- sqrt(mean((test_ts$low_mean - preds)^2))

rel_change <- abs(errors) / test_ts$low_mean
pred25 <- sum((rel_change<0.25)=="TRUE")/nrow(test_ts)    ## gives the count of those who are true on th
##OR pred25 <- table(rel_change<0.25)["TRUE"] / nrow(test_ts)

pred10 <- sum((rel_change<0.10)=="TRUE")/nrow(test_ts)   ## gives the count of those who are true on the

pred5 <- sum((rel_change<0.05)=="TRUE")/nrow(test_ts)   ## gives the count of those who are true on the

pred1 <- sum((rel_change<0.01)=="TRUE")/nrow(test_ts)   ## gives the count of those who are true on the

paste("RMSE:", rmse)
```

```
## [1] "RMSE: 0.894067825440456"
```

```
paste("Pred(25%):", pred25)
```

```
## [1] "Pred(25%): 1"
```

```
paste("Pred(10%):", pred10)
```

```
## [1] "Pred(10%): 0.999201277955272"
```

```r
paste("Pred(5%):", pred5)
```

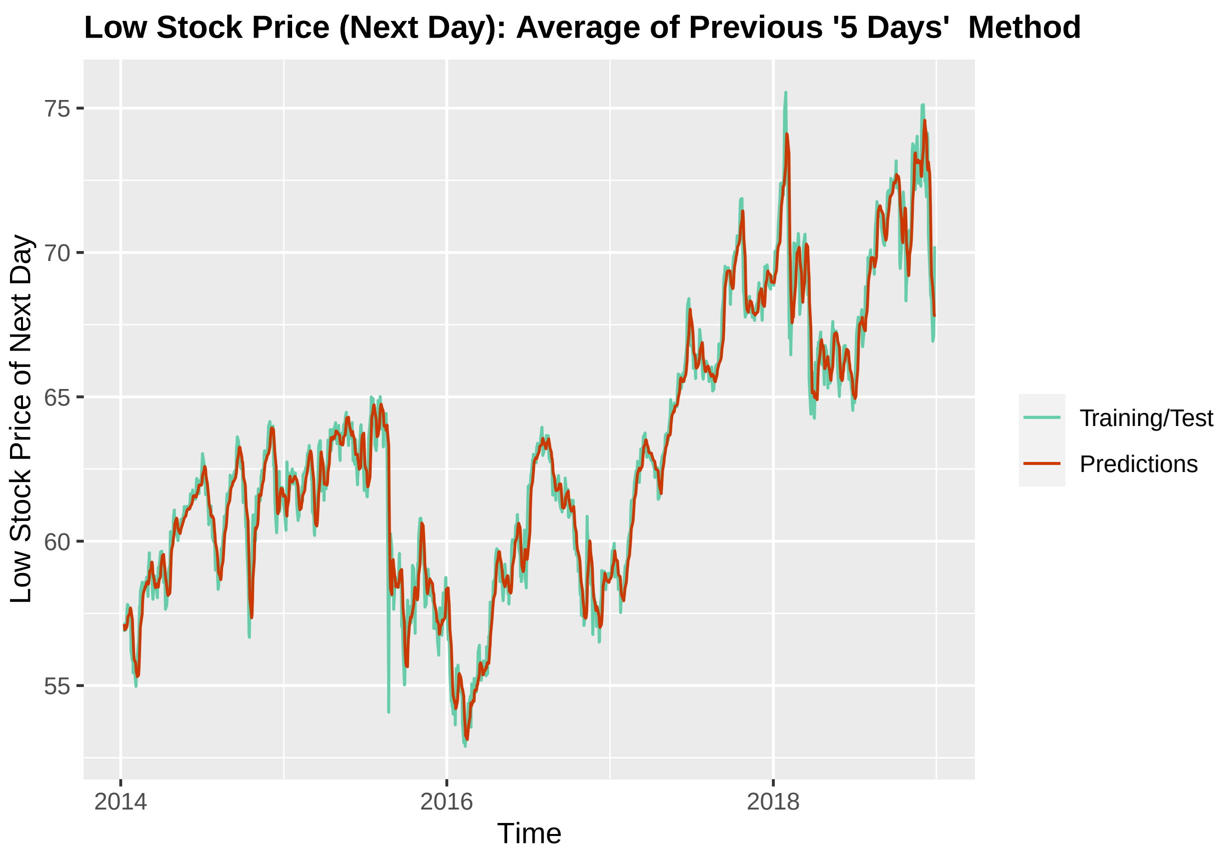
```
## [1] "Pred(5%): 0.996006389776358"
```

```r
paste("Pred(1%):", pred1)
```

```
## [1] "Pred(1%): 0.60223642172524"
```

##Plot

```r
test_ts$preds <- preds
ggplot(data = test_ts, aes(x = as.Date(date), y = low_mean,color = "aquamarine3"))+
  geom_line()+
  geom_line(data = test_ts, aes(x = as.Date(date), y = preds ,color="orangered3"))+
  scale_color_identity(name = " ",
                       breaks = c("aquamarine3", "orangered3"),
                       labels = c("Training/Test", "Predictions"),
                       guide = "legend")+
  xlab("Time")+
  ylab("Low Stock Price of Next Day")+
  ggtitle("Low Stock Price (Next Day): Average of Previous '5 Days'  Method") +
  theme(plot.title = element_text(size=12, face = "bold"))
```



Averaging out the low stock price registered on the five previous days seems to perform poorly when there is a sharp change.

#Long Short Term Memory (LSTM)???

#Version 2: Another approch to question 1

Using the 200+ financial indicators data to predict if whether or not a company's stock will go up or down the following year. From a trading perspective, we want to predict if whether or not an hypothetical trader should buy or sell at the end of the year (or at the start of the following year) for a profit.

Our dependent variable "Class" is binary with 0 indicating a decrease in stock value for the following year (do not buy/sell) and 1 indicating an increase in value for the following year (buy/do not sell).

From a trading perspective, an hypothetical trader should buy (or not sell) at the end of the year (or at the start of the following year) for a profit if "Class" is 1. On the other hand, an hypothetical trader should sell (or not but) at the end of the year (or at the start of the following year) if "Class" is 0 because the stock will decrease, meaning a loss of capital.

"From a trading perspective, the 1 identifies those stocks that an hypothetical trader should BUY at the start of the year and sell at the end of the year for a profit. From a trading perspective, the 0 identifies those stocks that an hypothetical trader should NOT BUY, since their value will decrease, meaning a loss of capital."

We will use: 1. Logistic Regression 2. KNN 3. Random Forest 4. SVM 5. Naive Bayes 6. Ensemble Model

#Libraries needed

```r
#install.packages('RCurl')
#install.packages('MASS')
#install.packages('leaps')
#install.packages('corrplot')
#install.packages('caret')
#install.packages("FNN")
#install.packages("mlbench)
library(tidyverse)
library(RCurl) # getURL
library(MASS) # stepwise regression
library(leaps) # all subsets regression
library(corrplot)
library(caret)
library(FNN)
library(mlbench)
library(rcompanion)
library(e1071)
```

#Import Datasets

```r
data2014<-read.csv("2014_Financial_Data.csv")
data2015<-read.csv("2015_Financial_Data.csv")
data2016<-read.csv("2016_Financial_Data.csv")
data2017<-read.csv("2017_Financial_Data.csv")
data2018<-read.csv("2018_Financial_Data.csv")
```

#Merge Datasets

```r
names(data2014)[names(data2014) == "X2015.PRICE.VAR...."] <- "PRICE_VAR_NY"

names(data2015)[names(data2015) == "X2016.PRICE.VAR...."] <- "PRICE_VAR_NY"
names(data2016)[names(data2016) == "X2017.PRICE.VAR...."] <- "PRICE_VAR_NY"

names(data2017)[names(data2017) == "X2018.PRICE.VAR...."] <- "PRICE_VAR_NY"

names(data2018)[names(data2018) == "X2019.PRICE.VAR...."] <- "PRICE_VAR_NY"

data<-rbind(data2014,data2015, data2016, data2017, data2018)
view(data)
```

#Explore Dependent (Target) Varibale: Class ##Table

```r
table(data$Class)
```

```
##
##     0     1
##  9918 12159
```

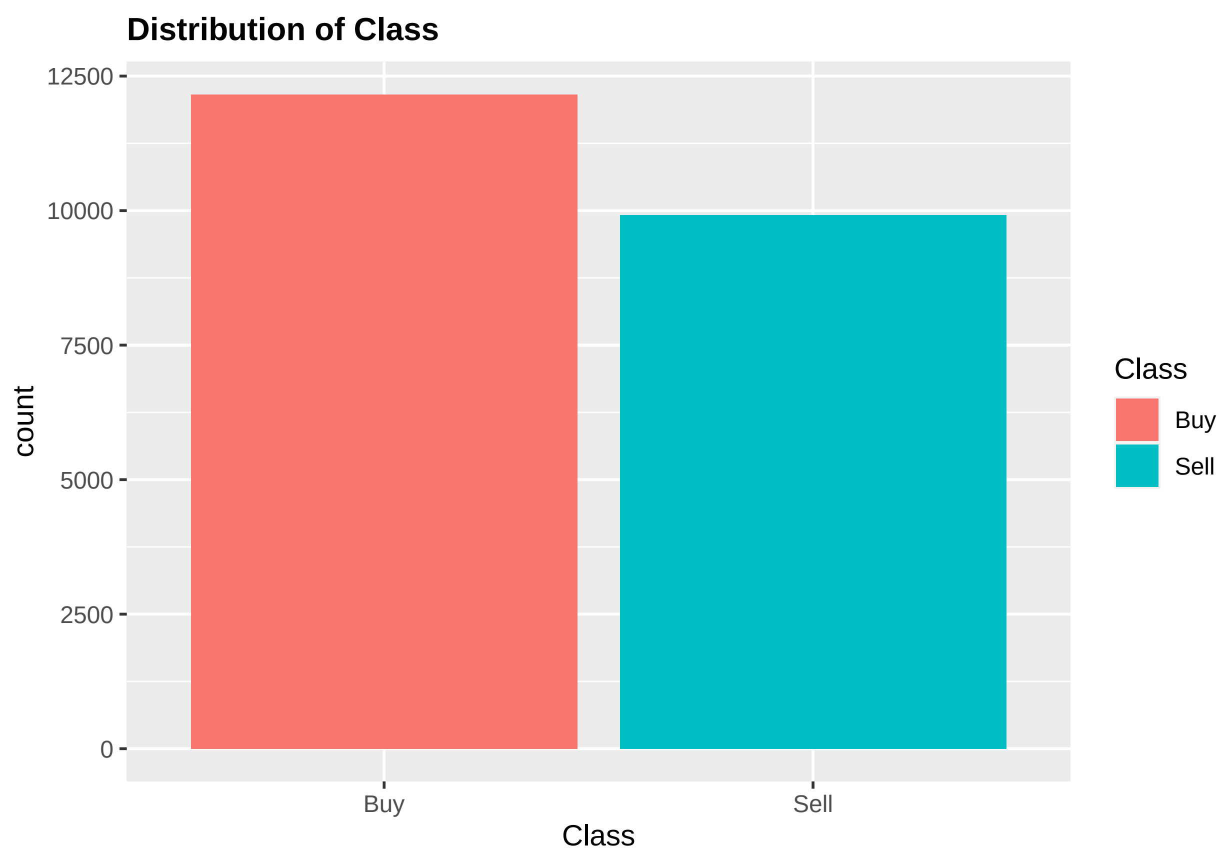
Change labels of Class

1:Buy 2:Sell

```r
data$Class  <- ifelse (data$Class ==1 , "Buy", "Sell")
```

##Plot

```r
ggplot(data)+
  geom_bar(mapping = aes(Class, fill=Class)) +
  ggtitle("Distribution of Class") +
  theme(plot.title = element_text(size=12, face = "bold"))
```



The target variable is fairly is balanced.

#Data Preprocessing (of IVs) ##Dropping the columns X and PRICE_VAR_NY X is just the name of the companies and PRICE_VAR_NY or the the percent price variation of each stock for the year and is the numeric counterpart of our dependent variable.

```
data<- data[ , -which(names(data) %in% c("X","PRICE_VAR_NY"))]
```

##Dealing with Missing Values ###Dropping columns wih too many zero or NA values

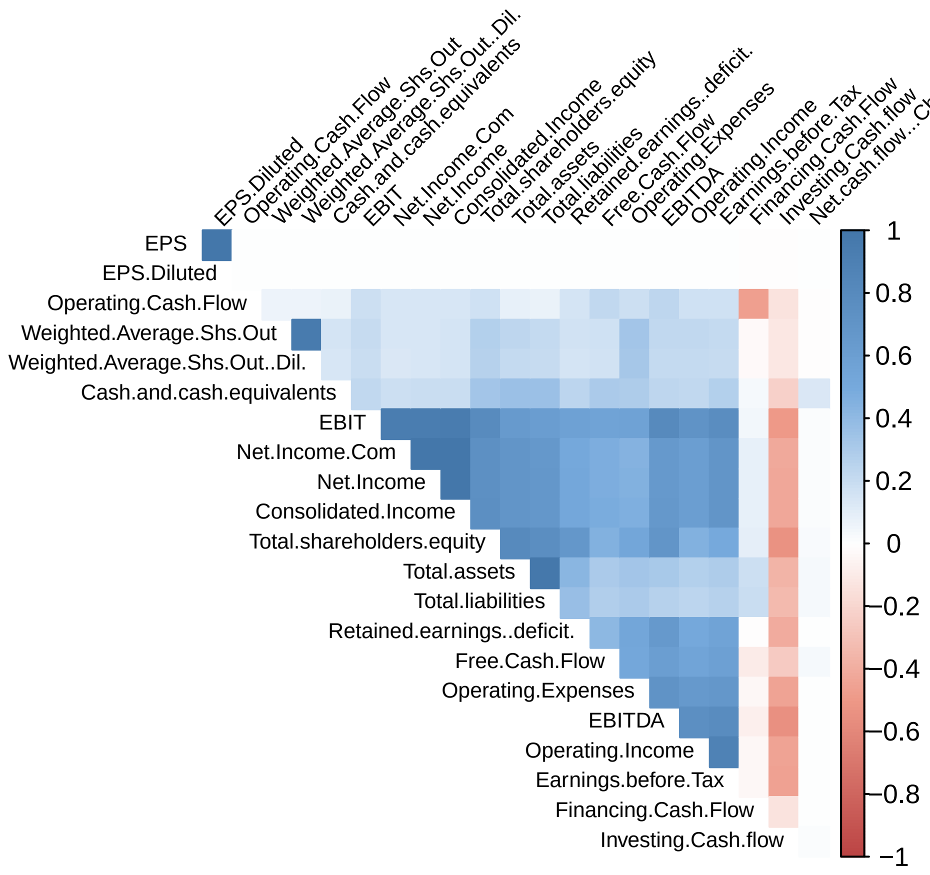
```
#Dropping columns with more than 2000 na values
a<-which(colSums(is.na(data))>2000)
data_bis<-data[,-a]

#Dropping columns with more than 2000zeros
a<-which(lapply(data_bis, function(x){ length(which(x==0)) })>2000)
data_bis<-data_bis[,-a]


#Dropping columns with more than 2000 na values or zeros
a<-which(lapply(data_bis, function(x){
  length(which(x==0)) + sum(is.na(x))
  })>2000)
data_bis<-data_bis[,-a]
```

###Replacing NA values with variable/column mean

Correlation Analysis

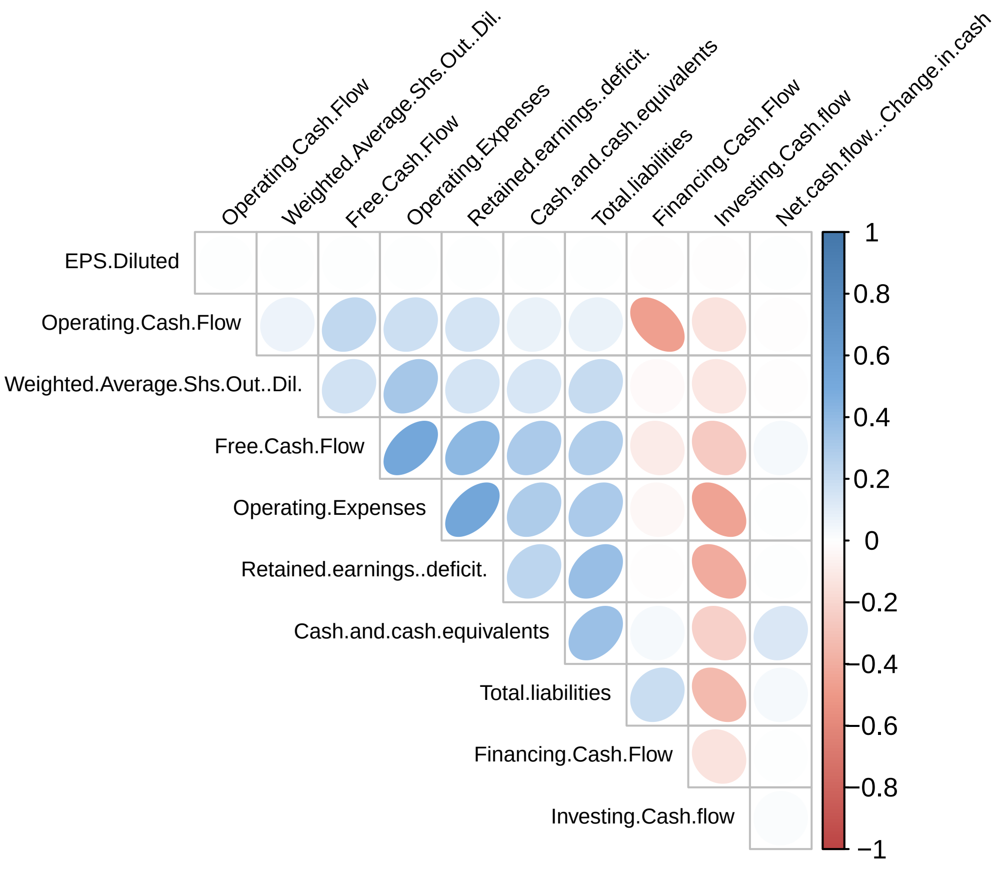
###Identify highly correlated “numeric” variables

```
###draw correlation matrix of the numeric independent variables only
num_data <- data_ter[ , -which(names(data_ter) %in% c("Class","Sector"))] ### remove Class because it i

correlationMatrix <- cor(num_data, method = "pearson")
col <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD", "#4477AA"))
corrplot(correlationMatrix, method="color", col=col(200),
         type="upper", order="hclust",
         tl.col="black", tl.srt=45, tl.cex= 0.7, #Text label color and rotation
         # Combine with significance
         sig.level = 0.01,
         # hide correlation coefficient on the principal diagonal
         diag=FALSE
)
```



###Remove highly correlated numeric IVs (0.6)

```
highlyCorrelated <- findCorrelation(correlationMatrix, cutoff=0.6)
noncor <- num_data[,-highlyCorrelated]  #keep only those not highly
correlationMatrix2 <- cor(noncor, method = "pearson")  ### only numeric vars
col <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD", "#4477AA"))
corrplot(correlationMatrix2, method="ellipse", col=col(200),
         type="upper", order="hclust",
         tl.col="black", tl.srt=45, tl.cex= 0.7, #Text label color and rotation
         # Combine with significance
         sig.level = 0.01,
         # hide correlation coefficient on the principal diagonal
         diag=FALSE
)
```



##Normalizing numeric IVs

```
normalize <- function(x) {
              return ((x - min(x)) / (max(x) - min(x))) }

noncor_n <- as.data.frame(lapply(noncor, normalize))

data_n<-cbind(noncor_n,data_ter[,c(23,24)])
data_n$Class<-as.factor(data_n$Class)
data_n$Sector<-as.factor(data_n$Sector)
```

Feature Selection using Random Forest

```
#install.packages("randomForest")
library(randomForest)
set.seed((123))
full=randomForest(Class~., data=data_n, ntree=500)
```

```r
importance    <- importance(full)
varImportance <- data.frame(Variables = row.names(importance),
                            Importance = importance[ ,'MeanDecreaseGini'])
x <- filter(varImportance, Importance>10)   #keep only variables with importance >10
Class <- data_n$Class
features <- as.character(x$Variables)
cleandata <- cbind(Class, data_n[,features])
```

#Train and test sets and labels (70/30 split)

```r
set.seed(123)
rn_train <- sample(nrow(cleandata), floor(nrow(cleandata)*0.70))
train_set <- cleandata[rn_train,]
test_set <- cleandata[-rn_train,]

train_labels <- cleandata[rn_train, 1]
test_labels <- cleandata[-rn_train, 1]
```

#Logistic regression classifier

```r
set.seed(123)
ctrl <- trainControl(method="repeatedcv", number =10, repeats=3)
LRmodel <- train(Class ~ ., data= train_set, method="glm", trControl = ctrl)
test_predLR <- predict(LRmodel, test_set)
cf_LR <- confusionMatrix(as.factor(test_predLR), as.factor(test_labels), positive="Buy" , mode = "every
print(cf_LR)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  Buy Sell
##       Buy  3041 2120
##       Sell  646  817
##
##                Accuracy : 0.5824
##                  95% CI : (0.5704, 0.5943)
##     No Information Rate : 0.5566
##     P-Value [Acc > NIR] : 1.191e-05
##
##                   Kappa : 0.1085
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8248
##             Specificity : 0.2782
##          Pos Pred Value : 0.5892
##          Neg Pred Value : 0.5584
##               Precision : 0.5892
##                  Recall : 0.8248
##                      F1 : 0.6874
##              Prevalence : 0.5566
##          Detection Rate : 0.4591
##    Detection Prevalence : 0.7791
##       Balanced Accuracy : 0.5515
##
##        'Positive' Class : Buy
##
```

#KNN classifier #Train and test sets and labels without target variables (Class)

```
train_set_s <- cleandata[rn_train,-1]
test_set_s <- cleandata[-rn_train,-1]

train_labels <- cleandata[rn_train, 1]
test_labels <- cleandata[-rn_train, 1]
```

```
oh_train <- model.matrix(~0+train_set_s[,'Sector'])
oh_test <- model.matrix(~0+test_set_s[,'Sector'])

#Renaming the columns to be more concise
attr(oh_train, "dimnames")[[2]] <- levels(train_set_s$Sector)
attr(oh_test, "dimnames")[[2]] <- levels(test_set_s$Sector)
```

We will only use odd numbers for K in order to avoid ties between classes.