# Homework 4 Solution:

The dataset was created using data from Yahoo Finance and consisted of information such as the end price of stock, 10-year bond interest and crude oil prices. The percentage change in price for the past 5 days is been calculated and stored as different variables for Stock, interest, and crude oil. In the process of cleaning the dataset, the mean is used to impute the values of the columns which had missing values.

Then, the variation in Price change of Stock Price is categorized into 5 labels based on comparing the value with product of certain integers and standard deviation. The categorical variable consisted of 5 levels and hence, the models chosen are Naïve Bayes, Recursive Partitioning and Generalized Boosting Model.

With respect to Naïve Bayes, the accuracy achieved was 25% and when cross validation is performed, the highest accuracy which is achieved was 33%. As we know that the Naïve Bayes calculates the probability based on the likelihood of two events occurring together which is not a frequent scenario in Stock Market and hence the accuracy is less for this model.

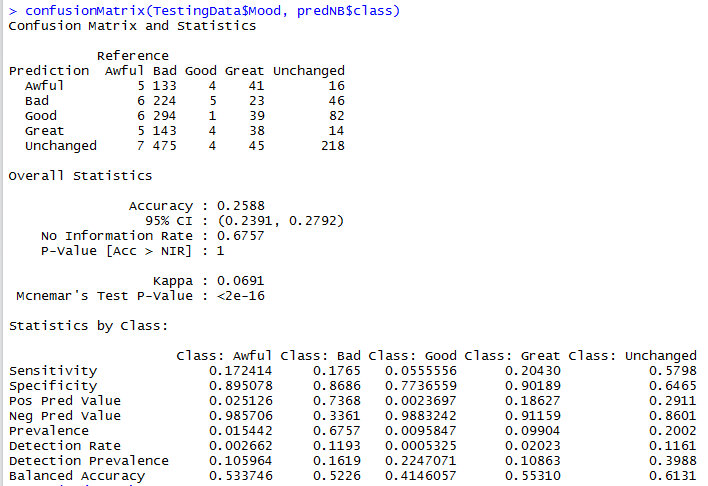
In Recursive Partitioning, the accuracy that’s achieved is 31%. As we know that the Recursive Partitioning works by splitting the independent variables based on the values of the dependent variable and eventually forming a tree. The overfitting of the model can be prevented by choosing an optimum value for Control Parameter.

The Gradient boosting model has the highest accuracy of 41% when compared to the prior models. It is an ensemble model and creates random forest of tree structure thereby leading to increase in accuracy.

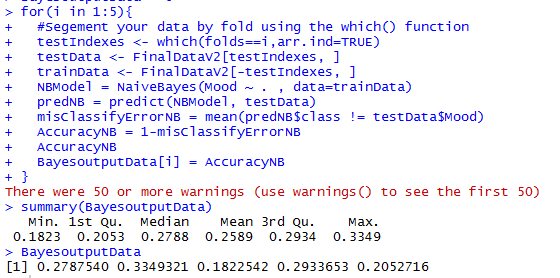
An accuracy of 41% in Stock Market prediction can be considered as an average model and high accuracy wasn’t able to be achieved since the Stock Market price changes based on different attribute and we have used only few attributes to predict the change. Hence, by including more independent variables, the accuracy can be improved.

## Result:

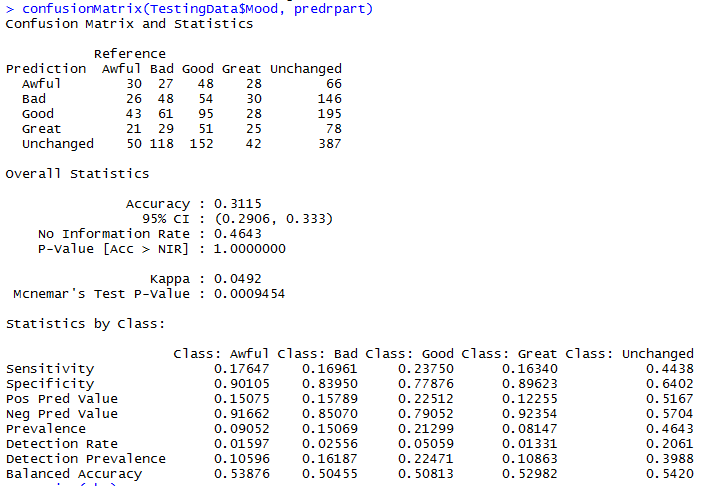
### Naïve Bayes:



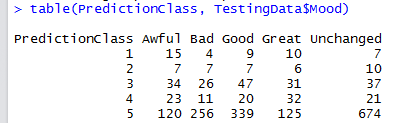
Using Cross validation:



### Recursive Partitioning:



### Generalized Boosting Model:



Accuracy = (15+7+47+32+674)/1878 = 41.26%

## R – Code:

Stocks <- read.csv(file.choose())

str(Stocks)

Stocks$Date <- as.Date(Stocks$Date)

Stocksv1 <- Stocks[, -c(2:6)]

colnames(Stocksv1) <- c('Date', 'StocksAdjClose')

Interest <- read.csv(file.choose())

str(Interest)

Interest$Date <- as.Date(Interest$Date)

Interestv1 <- Interest[, -c(2:6)]

colnames(Interestv1) <- c('Date', 'InterestAdjClose')

require(readxl)

Crude <- read\_excel(file.choose(), sheet = "Data 1", skip = 2)

str(Crude)

Crude$Date <- as.Date(Crude$Date)

colnames(Crude) <- c('Date', 'CrudeAdjClose','AdjClose\_2')

Crudev1 <- Crude[, -c(3)]

StockInterest <- merge(Stocksv1, Interestv1, by.x = 'Date', by.y = 'Date')

FinalData <- merge(StockInterest, Crudev1, by.x = 'Date', by.y = 'Date')

summary(FinalData)

FinalData = transform(FinalData, CrudeAdjClose = ifelse(is.na(CrudeAdjClose), mean(CrudeAdjClose, na.rm=TRUE), CrudeAdjClose))

is.na(FinalData$CrudeAdjClose)

summary(FinalData)

library(dplyr)

FinalData$Stocks\_PC1 <- ((lag(FinalData$StocksAdjClose, 1)-lag(FinalData$StocksAdjClose,2))/lag(FinalData$StocksAdjClose, 2))\*100

FinalData$Stocks\_PC2 <- ((lag(FinalData$StocksAdjClose, 2)-lag(FinalData$StocksAdjClose,3))/lag(FinalData$StocksAdjClose, 3))\*100

FinalData$Stocks\_PC3 <- ((lag(FinalData$StocksAdjClose, 3)-lag(FinalData$StocksAdjClose,4))/lag(FinalData$StocksAdjClose, 4))\*100

FinalData$Stocks\_PC4 <- ((lag(FinalData$StocksAdjClose, 4)-lag(FinalData$StocksAdjClose,5))/lag(FinalData$StocksAdjClose, 5))\*100

FinalData$Stocks\_PC5 <- ((lag(FinalData$StocksAdjClose, 5)-lag(FinalData$StocksAdjClose,6))/lag(FinalData$StocksAdjClose, 6))\*100

FinalData$Interest\_PC1 <- ((lag(FinalData$InterestAdjClose, 1)-lag(FinalData$InterestAdjClose,2))/lag(FinalData$InterestAdjClose, 2))\*100

FinalData$Interest\_PC2 <- ((lag(FinalData$InterestAdjClose, 2)-lag(FinalData$InterestAdjClose,3))/lag(FinalData$InterestAdjClose, 3))\*100

FinalData$Interest\_PC3 <- ((lag(FinalData$InterestAdjClose, 3)-lag(FinalData$InterestAdjClose,4))/lag(FinalData$InterestAdjClose, 4))\*100

FinalData$Interest\_PC4 <- ((lag(FinalData$InterestAdjClose, 4)-lag(FinalData$InterestAdjClose,5))/lag(FinalData$InterestAdjClose, 5))\*100

FinalData$Interest\_PC5 <- ((lag(FinalData$InterestAdjClose, 5)-lag(FinalData$InterestAdjClose,6))/lag(FinalData$InterestAdjClose, 6))\*100

FinalData$Crude\_PC1 <- ((lag(FinalData$CrudeAdjClose, 1)-lag(FinalData$CrudeAdjClose,2))/lag(FinalData$CrudeAdjClose, 2))\*100

FinalData$Crude\_PC2 <- ((lag(FinalData$CrudeAdjClose, 2)-lag(FinalData$CrudeAdjClose,3))/lag(FinalData$CrudeAdjClose, 3))\*100

FinalData$Crude\_PC3 <- ((lag(FinalData$CrudeAdjClose, 3)-lag(FinalData$CrudeAdjClose,4))/lag(FinalData$CrudeAdjClose, 4))\*100

FinalData$Crude\_PC4 <- ((lag(FinalData$CrudeAdjClose, 4)-lag(FinalData$CrudeAdjClose,5))/lag(FinalData$CrudeAdjClose, 5))\*100

FinalData$Crude\_PC5 <- ((lag(FinalData$CrudeAdjClose, 5)-lag(FinalData$CrudeAdjClose,6))/lag(FinalData$CrudeAdjClose, 6))\*100

FinalData$PriceChange <- FinalData$StocksAdjClose -lag(FinalData$StocksAdjClose, 1)

FinalData$PriceChange[is.na(FinalData$PriceChange)] <- round(mean(FinalData$PriceChange, na.rm = TRUE))

FinalDataV1 <- FinalData[-c(1:6),]

PriceChangeSTD <- sd(FinalDataV1$PriceChange)

FinalDataV1$Mood <- ifelse((FinalDataV1$PriceChange < -1\*PriceChangeSTD), "Awful",

ifelse((FinalDataV1$PriceChange >= -1\*PriceChangeSTD & FinalDataV1$PriceChange < -0.3\*PriceChangeSTD), "Bad",

ifelse((FinalDataV1$PriceChange >= -0.3\*PriceChangeSTD & FinalDataV1$PriceChange < 0.3\*PriceChangeSTD), "Unchanged",

ifelse((FinalDataV1$PriceChange >= 0.3\*PriceChangeSTD & FinalDataV1$PriceChange < 1\*PriceChangeSTD), "Good",

"Great"))))

#Naive Bayes

library ("klaR")

library ("caret")

library ("e1071")

FinalDataV2 <- subset(FinalDataV1, select = -c(1,2,3,4,20))

str(FinalDataV2)

FinalDataV2$Mood <- as.factor(FinalDataV2$Mood)

set.seed(3976)

FinalDataV2 <- FinalDataV2[sample(nrow(FinalDataV2)),]

folds <- cut(seq(1,nrow(FinalDataV2)),breaks=5,labels=FALSE)

head(folds)

tail(folds)

BayesoutputData = 0

#Perform 10 fold cross validation

for(i in 1:5){

#Segement your data by fold using the which() function

testIndexes <- which(folds==i,arr.ind=TRUE)

testData <- FinalDataV2[testIndexes, ]

trainData <- FinalDataV2[-testIndexes, ]

NBModel = NaiveBayes(Mood ~ . , data=trainData)

predNB = predict(NBModel, testData)

misClassifyErrorNB = mean(predNB$class != testData$Mood)

AccuracyNB = 1-misClassifyErrorNB

AccuracyNB

BayesoutputData[i] = AccuracyNB

}

summary(BayesoutputData)

BayesoutputData

#Rpart

RPoutputData = 0

require(rpart)

for(i in 1:5){

#Segement your data by fold using the which() function

testIndexes <- which(folds==i,arr.ind=TRUE)

testData <- FinalDataV2[testIndexes, ]

trainData <- FinalDataV2[-testIndexes, ]

RpartModel = rpart(Mood ~ . , data=trainData, cp=0)

predRP = predict(RpartModel, testData)

predRP

misClassifyErrorRP = mean(predRP != testData$Mood)

AccuracyRP = 1-misClassifyErrorRP

AccuracyRP

RPoutputData[i] = AccuracyRP

}

summary(RPoutputData)

RPoutputData

#Naive Bayes

FinalDataV3 <- subset(FinalDataV1, select = -c(1,2,3,4,20))

str(FinalDataV3)

FinalDataV3$Mood <- as.factor(FinalDataV3$Mood)

selectDS<-sample(1:6257, .7\*6257)

TrainingData <-FinalDataV3[selectDS,]

TestingData <-FinalDataV3[-selectDS,]

NBModelV1 = NaiveBayes(Mood ~ . , data=TrainingData)

predNB = predict(NBModelV1, TestingData)

predNB

confusionMatrix(TestingData$Mood, predNB$class)

#Rpart

require(rpart)

RpartModel = rpart(Mood ~ . , data=TrainingData, cp=0)

predrpart = predict(RpartModel, TestingData, type = 'class')

predrpart

confusionMatrix(TestingData$Mood, predrpart)

#GBM

require(gbm)

library(gbm)

GbmModel<- gbm(Mood ~ . , data=TrainingData, n.trees=200, cv.folds = 5,interaction.depth=6, shrinkage=0.01)

predGbm = predict(GbmModel, newdata=TestingData, n.trees=200, type="response")

PredictionClass <- apply(predGbm,1,which.max)

#PredictionClass = ifelse(predGbm[,1,]>0.18 ,0,0)

#PredictionClass = ifelse(predGbm[,2,]>0.18 ,1,PredictionClass)

#PredictionClass = ifelse(predGbm[,3,]>0.18 ,2,PredictionClass)

#PredictionClass = ifelse(predGbm[,4,]>0.18 ,3,PredictionClass)

#PredictionClass = ifelse(predGbm[,5,]>0.18 ,4,PredictionClass)

View(prediction3)

PredictionClass

table(PredictionClass, TestingData$Mood)

head(TestingData$Mood)