

Predicting Credit Card Defaults

Data Mining Project on Classification



Report by:

Pallavi taneja

Table of Contents

[Introduction 2](#_Toc532077415)

[Objectives 2](#_Toc532077416)

[Data Source 2](#_Toc532077417)

[Data Set 2](#_Toc532077418)

[Exploratory Data Analysis 3](#_Toc532077419)

[Stratified Sampling of Data 3](#_Toc532077420)

[Data Preprocessing 3](#_Toc532077421)

[Graphical Results 3](#_Toc532077422)

[Methodology 6](#_Toc532077423)

[Algorithm Performance Comparison 6](#_Toc532077424)

[Variable Importance 7](#_Toc532077425)

[Model Correlation 8](#_Toc532077426)

[Model Improvement with Stacked and Ensemble Models 8](#_Toc532077427)

[Cost Analysis 9](#_Toc532077428)

[Conclusion and Recommendation 9](#_Toc532077429)

[Appendix 10](#_Toc532077430)

[R Code with sample output 10](#_Toc532077431)

# Introduction

A Taiwanese Banking Institution has provided data related to the credit card payments of their customers. The data highlights customers’ credit card bill amount and their payment history over the last six months. In this project, I have built and compared classification models which predicts the clients who are more likely to default the next month’s credit card payment.

For this project, I have also conducted the cost analysis for the bank. The bank would also like to contact the customers who are likely to default through a phone call. If the bank does not make the call, the bank is assumed to lose a certain amount of money, this means there is a cost associated with bank not being able to identify the customers who are likely to default.

# Objectives

* Build a classification model to predict whether the client will default the credit card payment next month or not
* To identify a model that minimizes the cost to the bank which includes the cost of making a call as well as cost of not being able to identify the customers who are likely to default

# Data Source

The data is available on UCI Machine Learning repository (https://archive.ics.uci.edu/). The data set contains 30000 instances and 24 attributes. The outcome is whether the customer will or will not default the next month’s credit card payment. 22% of the 30000 customers defaulted the next month’s payment in the dataset.

# Data Set

This output or response variable is a binary variable that depicts default payment (Yes = 1, No = 0), as the response variable. There are 23 independent variables which are used to predict the default of payment and are explained as below:

|  |  |  |
| --- | --- | --- |
| **Variable** | **Variable Name** | **Description** |
| X1 | LIMIT\_BAL | Amount of the given credit (NT dollar): it includes both the individual consumer credit and his/her family (supplementary) credit |
| X2 | SEX | Gender (1 = male; 2 = female) |
| X3 | EDUCATION | Education (1 = graduate school; 2 = university; 3 = high school; 4 = others) |
| X4 | MARRIAGE | Marital status (1 = married; 2 = single; 3 = others) |
| X5 | AGE | Age (year) |
| X6 | PAY\_1 | The repayment status in September, 2005 |
| X7 | PAY\_2 | The repayment status in August, 2005 |
| X8 | PAY\_3 | The repayment status in July, 2005 |
| X9 | PAY\_4 | The repayment status in June, 2005 |
| X10 | PAY\_5 | The repayment status in May, 2005 |
| X11 | PAY\_6 | The repayment status in April, 2005 |
| X12 | BILL\_AMT1 | Amount of bill statement in September, 2005 |
| X13 | BILL\_AMT2 | Amount of bill statement in August, 2005 |
| X14 | BILL\_AMT3 | Amount of bill statement in July, 2005 |
| X15 | BILL\_AMT4 | Amount of bill statement in June, 2005 |
| X16 | BILL\_AMT5 | Amount of bill statement in May, 2005 |
| X17 | BILL\_AMT6 | Amount of bill statement in Apil, 2005 |
| X18 | PAY\_AMT1 | Amount paid in September, 2005 |
| X19 | PAY\_AMT2 | Amount paid in August, 2005 |
| X20 | PAY\_AMT3 | Amount paid in July, 2005 |
| X21 | PAY\_AMT4 | Amount paid in June, 2005 |
| X22 | PAY\_AMT5 | Amount paid in May, 2005 |
| X23 | PAY\_AMT6 | Amount paid in April, 2005 |
| Y | default payment next month | Payment default (Yes = 1, No = 0) |

Note: Measurement scale for the repayment status is: -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; 3 = payment delay for three months; 8 = payment delay for eight months; 9 = payment delay for nine months and above.

# Exploratory Data Analysis

## Stratified Sampling of Data

The dataset has 30000 instances; therefore, I performed a stratified sampling of the data to take a subset of data to perform classification. I took 15% of the data which resulted in a selection of 4500 instances.

This data was further divided into training and test data. 67% or 2/3 of the data was used to train the classifier and 33% or 1/3 of the data was used to test the performance of the classifiers used.

## Data Preprocessing

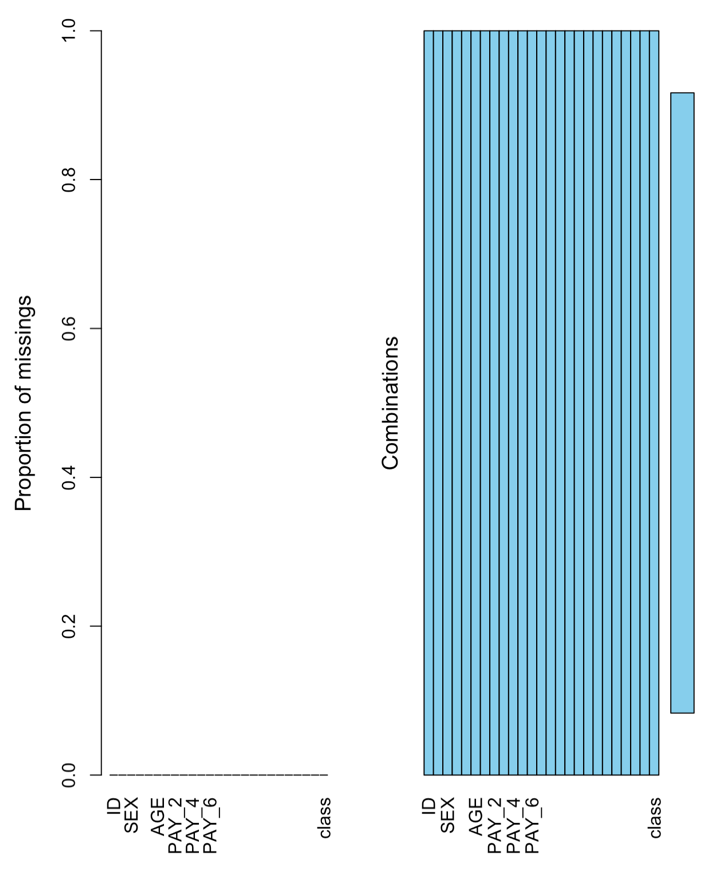
There was no missing data and hence no data cleaning was required. However, some preprocessing was conducted to make the data readable and ready for classification:

* The data type for all the variables was numeric, the nominal variables were identified and converted into factor variables.
* The output column was renamed as ‘class’ for ease of reference
* The class column was recoded as ‘yes’ for values ‘1’ and ‘no’ for values ‘0’
* The class column was releveled to make sure the positive class is always considered as ‘yes’
* The ID column was dropped since it had unique values

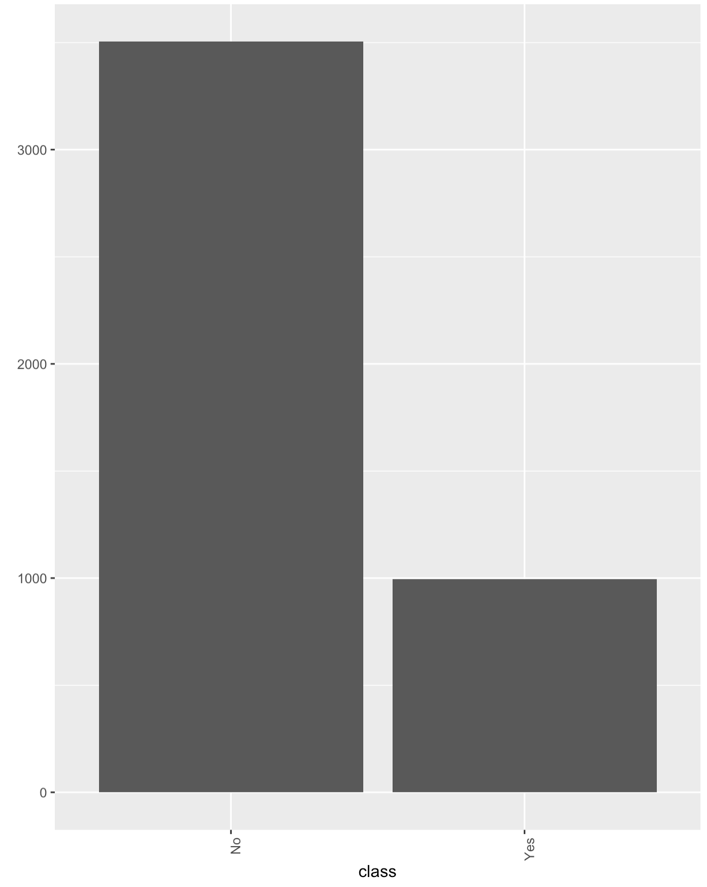
## Graphical Results

Some basic visualization was performed on the dataset like missing values, class and gender distribution, and amount of given credit.

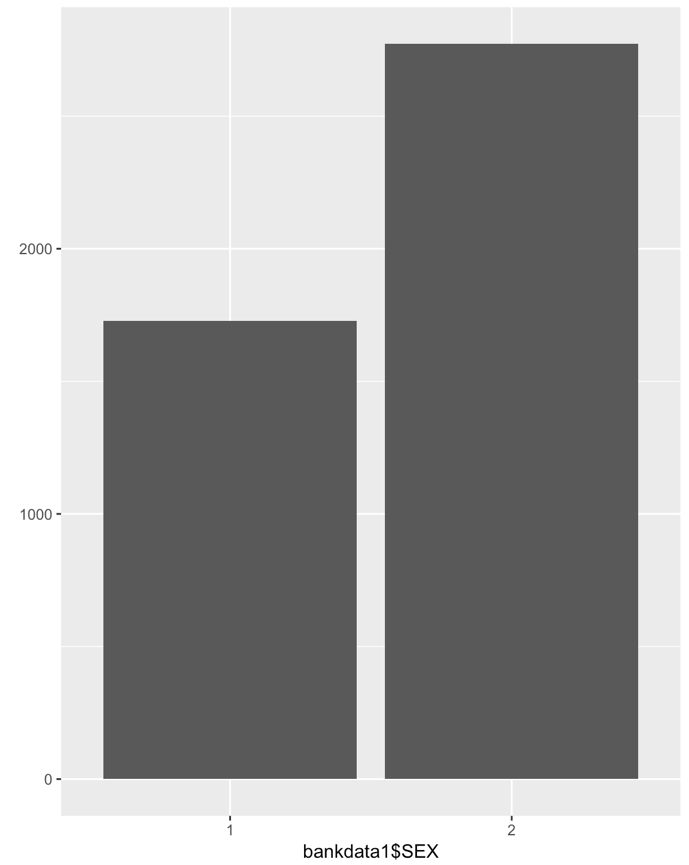
Missing Values



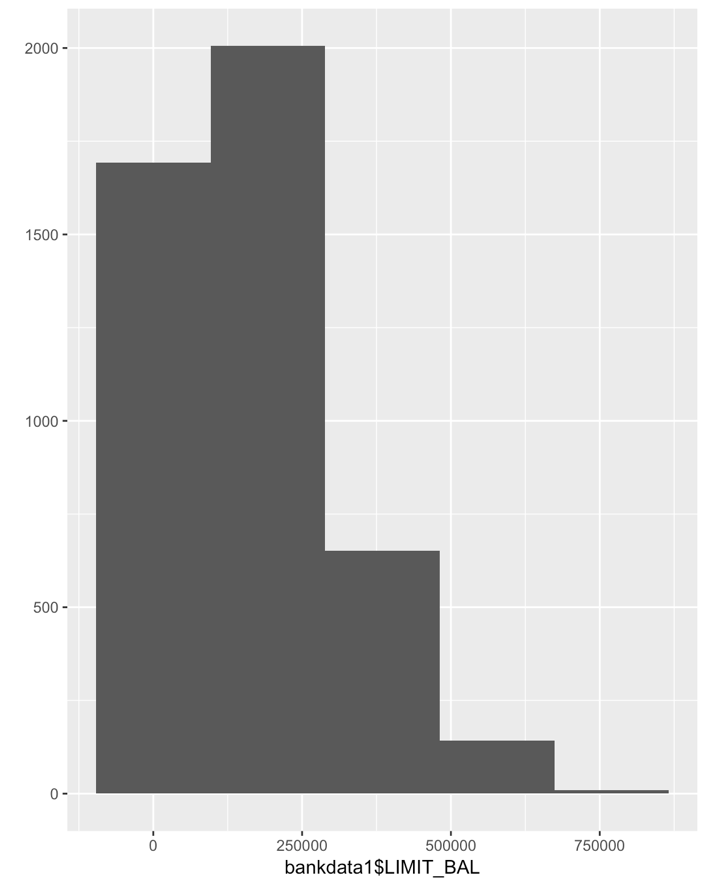
Class distribution



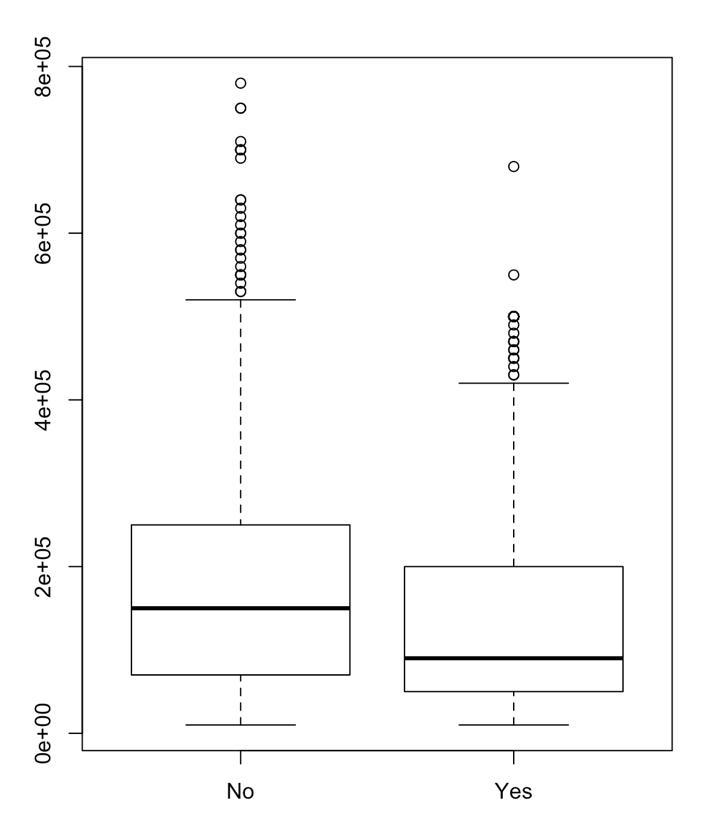
Gender distribution



Amount of given credit



Amount of given credit in the respective classes



# Methodology

The task at hand was a binary classification problem with two possible responses or classes – Yes and No. I used 4 individual classifiers for prediction at first and then built stacked and ensemble models using all 4 classifiers and two of the least correlated of them.

* **Nonlinear algorithms**: Decision Tree (C5.0) and Naïve Bayes (NB)
* **Bagging algorithms**: Random Forest (RF)
* **Boosting algorithms**: Gradient Boosting (GBM)

## Algorithm Performance Comparison

The 4x4 Matrix (Models vs Metrics) below lists the performance of each classifier:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **Sensitivity** | **Specificity** | **Precision** | **F1 Score** |
| **Decision Tree** | 0.8113 | 0.3293 | 0.9481 | 0.6429 | 0.4355 |
| **Naïve Bayes** | 0.6813 | 0.6768 | 0.6825 | 0.3768 | 0.4842 |
| **Random Forest** | 0.7999 | 0.4207 | 0.9074 | 0.5633 | 0.4817 |
| **GBM** | 0.7776 | 0.5762 | 0.8348 | 0.4974 | 0.5339 |

Since it is an imbalanced dataset, accuracy alone cannot be a good measure of classifier performance, we need precision and/or recall too to identify the best performing classifier. The imbalance of dataset also means that the models are better at predicting ‘no’ class (customers who do not default a payment) as compared to the ‘yes’ class. This is due to the fact that there are more customers who didn’t default in the dataset and we took a stratified sample for predicting to represent the dataset correctly.

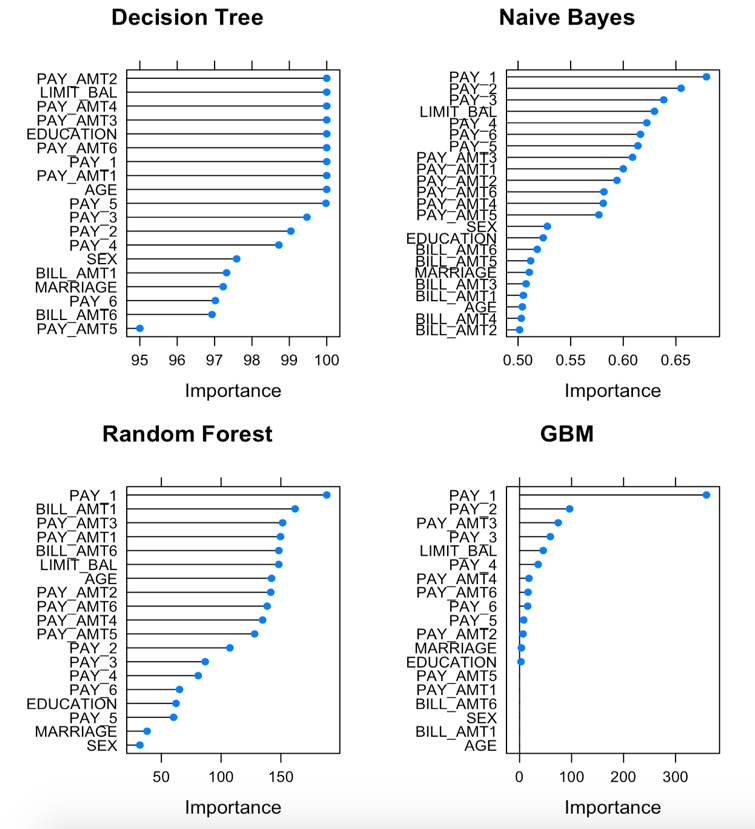
And since our goal here is to predict whether the customer will default or not, recall / sensitivity holds more importance for us. The bank would like to correctly identify the customers who are likely to default. Here, our positive class is ‘yes’ and since Recall is TP/P, we need high recall to be able to correctly identify all true positive class customers. Decision Tree gives us the best accuracy and precision but lowest recall. Naïve Bayes gives us least accuracy and precision but maximum recall. Random Forest and GBM give us a balanced mix of Accuracy, Sensitivity and Precision.

I performed a weighted F-measure comparison with Beta > 1 to further compare the performance of the classifiers. GBM has the maximum weighted F measure with Beta = 1.1 and hence is the recommended model.

|  |  |
| --- | --- |
| **Model** | **F score at Beta = 1.1** |
| **Decision Tree** | 0.4225 |
| **Naïve Bayes** | 0.4976 |
| **Random Forest** | 0.4751 |
| **GBM** | 0.5377 |

## Variable Importance

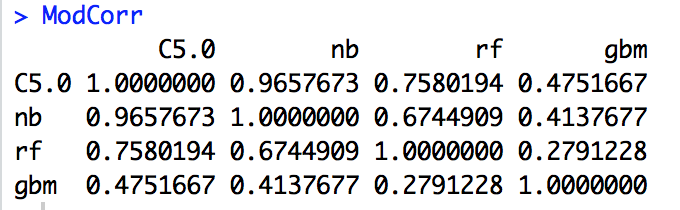
Variable importance for all 4 models developed was also studied and the variable PAY\_1 i.e. the repayment status in September 2005 was pegged as one of the most important among all the classifiers. This means that the customers who missed the last payment are most likely to default the next credit card payment. Therefore, it is suggested to the bank that more emphasis should be put on these customers and they should be identified and contacted to urge for meeting the payment deadline.

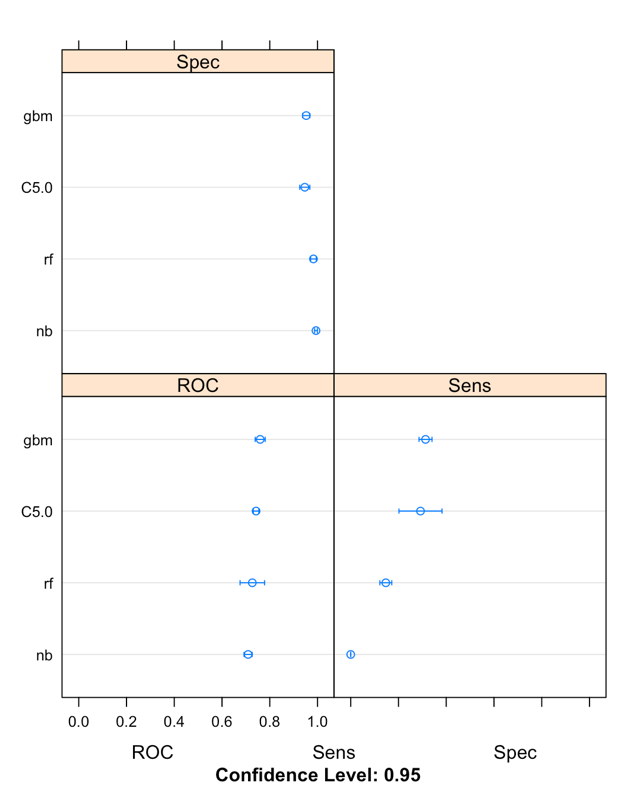
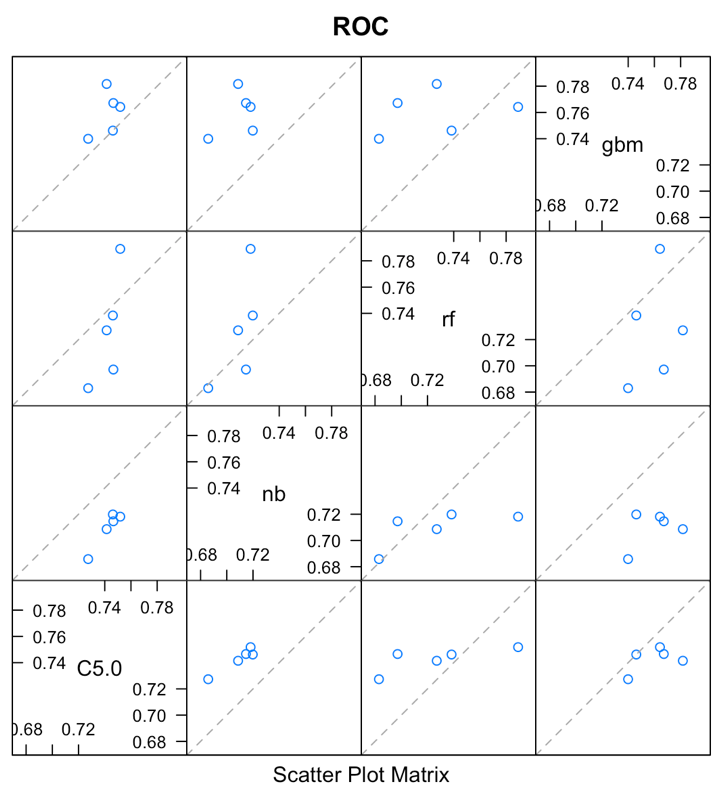


## Model Correlation

Then I studied the correlation for 4 models developed earlier to find out the least correlated models.

4x4 Model Prediction Correlation Matrix and plots:





## Model Improvement with Stacked and Ensemble Models

The models were further improved by building stacked and ensemble models using all the 4 models developed earlier and the two least correlated models identified above through model correlation. The two least correlated models were used so that if one model does not work the other would give the desired results.

6 models in total were built with sensitivity as the optimization metric and tested with the following results:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **Sensitivity** | **Precision** | **F1 Score** |
| **Stack – all 4 models** | 0.8261 | 0.4390 | 0.6606 | 0.5275 |
| **Stack – 2 least correlated models** | 0.8187 | 0.4543 | 0.6234 | 0.5256 |
| **Stack – all 4 models + up sampling** | 0.7983 | 0.5274 | 0.5340 | 0.5307 |
| **Ensemble – all 4 models** | 0.8255 | 0.3476 | 0.7169 | 0.4682 |
| **Ensemble – 2 least correlated models** | 0.8248 | 0.3415 | 0.7179 | 0.4628 |
| **Ensemble – all 4 models + up sampling** | 0.7803 | 0.5549 | 0.5028 | 0.5275 |

Ensemble model built with all 4 classifiers and up sampling resulted in the maximum recall but least accuracy and precision. The stacked model built using all 4 models and up sampling gave slightly less recall but performed better on accuracy and precision. It also has the highest F1 score.

Therefore, up sampling for both stacked and ensemble models has resulted in the best numbers, but it is still not as good as GBM in terms of Recall (0.5762) and F1 Score (0.5339).

## Cost Analysis

For this project, I have assumed that after predicting the customers who are likely to default the next month’s payment, the bank would like to take some action. Let’s say the bank calls every customer who is expected to default. When a person who is not likely to default but is contacted (FP / False Positive) the bank incurs a cost of $1. And for every person who is expected to default but is not contacted (FN / False Negative), the bank incurs a cost of $10.

Three models were built and tested to minimize the cost to the bank. The models were developed using Decision Tree, Naïve Bayes and Random Forest algorithms and the profit calculated for each is as follows. The Naïve Bayes method results in the minimum cost and maximum profits.

|  |  |  |
| --- | --- | --- |
| **Model** | **Cost in $** | **Profit in $** |
| **Decision Tree** | 1563 | 1527 |
| **Naïve Bayes** | 1260 | 1738 |
| **Random Forest** | 2227 | 942 |

# Conclusion and Recommendation

This project was focused on building a model to predict whether a customer will default the next month credit card payment and identify the model that minimizes the cost for the bank.

I performed required data preprocessing including stratified sampling and datatype conversion. Then I developed a mix of non-linear, bagging, and boosting algorithms and assessed them using accuracy, recall, precision and weighted F-measure. GBM was the recommended classifier with approx. 78% accuracy and 58% recall.

Model correlation was performed and identified two least correlated models for building a mix of stacked and ensemble models. I used a combination of 4 algorithms and 2 least correlated models with and without up sampling to improve the performance.

Ensemble model built with all 4 classifiers and up sampling gave the best scores for recall and F1 among all the 6 ensemble and stacked models. But, GBM fared out to be the best classifier for recall and F measure.

Considering cost while training the data, Naïve Bayes method resulted in the most cost efficient and profitable model in comparison to Decision Tree and Random Forest.

It was identified that customers who missed the last payment are most likely to default the next credit card payment, therefore the bank should put more emphasis on these customers and they should be contacted to urge for meeting the payment deadline.

# Appendix

## R Code with sample output

#--------------------------------------Begin--------------------------------------#

# load libraries

library(dplyr)

library(ggplot2)

library(caret)

library(plyr)

library(C50)

library(kernlab)

library(reshape)

library(MLmetrics)

library(caretEnsemble)

library(naivebayes)

library(neuralnet)

library(readxl)

library(VIM)

library(splitstackshape)

library(rpart)

library(gbm)

# load traning dataset

bankdata <- read\_excel("Desktop/620 - Data Mining/Project/default of credit card clients.xlsx")

# summary of dataset

summary(bankdata)

# view top rows of dataset

head(bankdata)

# check class of dataset

class(bankdata)

# variables in the dataset

names(bankdata)

#variable types of dataset

str(bankdata)

#rename last column as class

colnames(bankdata)[25] <- "class"

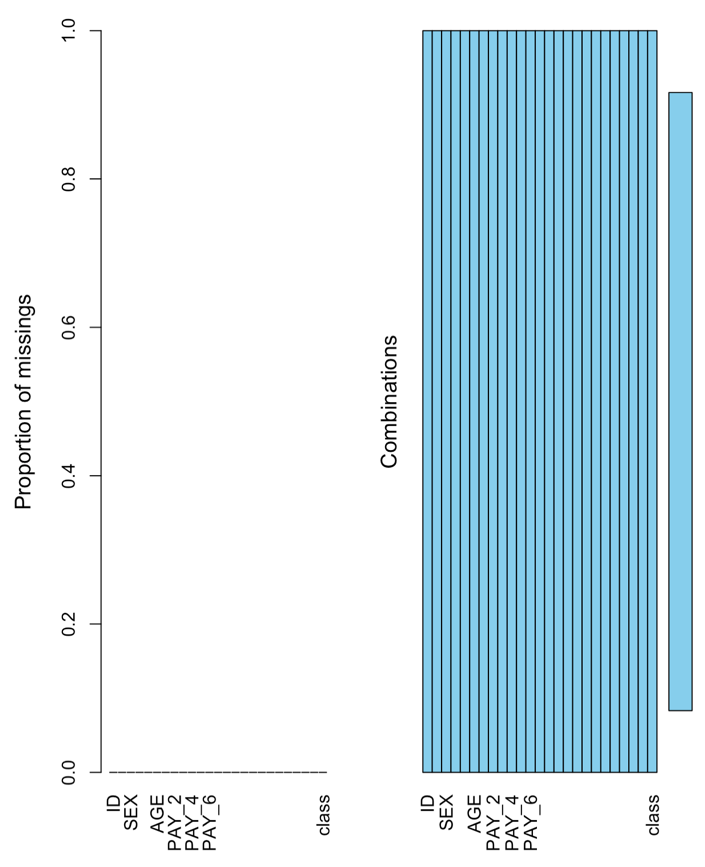
#convert into dataframe

bankdata <- as.data.frame(bankdata[])

View(bankdata)

# check for missing values

summary(aggr(bankdata, plot = TRUE))



#---------------Preparing Data for Classification----------------#

#do stratified sampling to take a subset of the data

bankdata <- stratified(bankdata, group = "class", size = 0.15,

select = NULL, replace = FALSE,

bothSets = FALSE)

View(bankdata)

# write the sampled data to a csv file

write.csv(bankdata,"Desktop/620 - Data Mining/Project/sampled data.csv")

# Convert nominal variables to factor but first copy in a new dataframe

bankdata1 <- bankdata

str(bankdata1)

bankdata1[,c(3:5,7:12,25)] <- lapply(bankdata1[,c(3:5,7:12,25)], as.factor)

str(bankdata1)

#convert positive class 1 as yes and negative class as no in the dataset

bankdata1$class <- revalue(bankdata1$class, c("1"="Yes"))

bankdata1$class <- revalue(bankdata1$class, c("0"="No"))

View(bankdata1)

# class summary for dataset

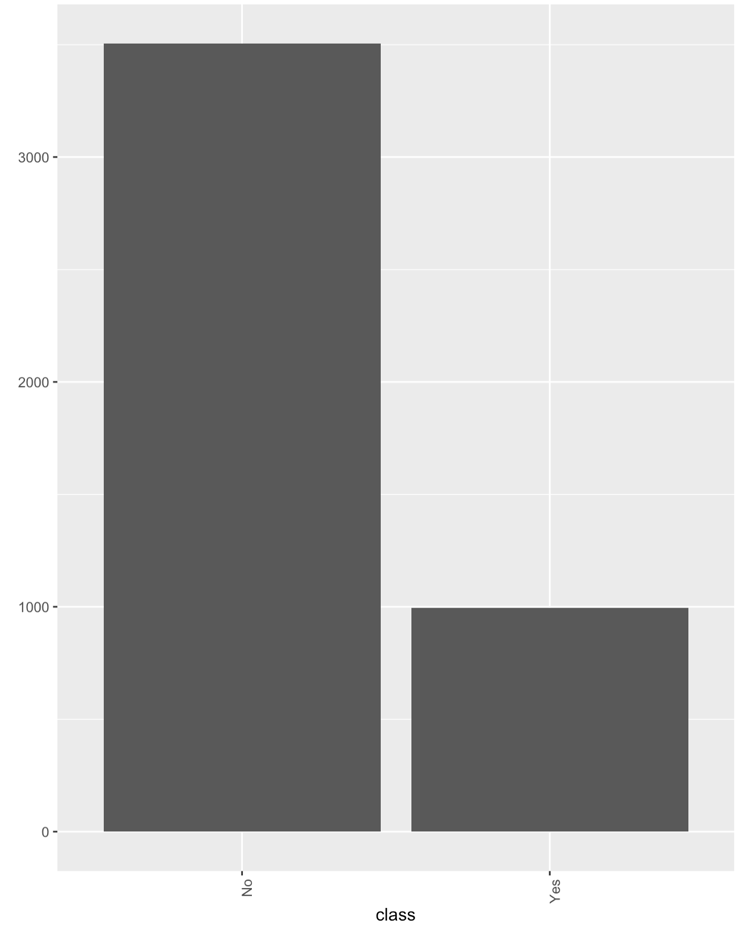
summary(bankdata1$class)

#drop the customerid column

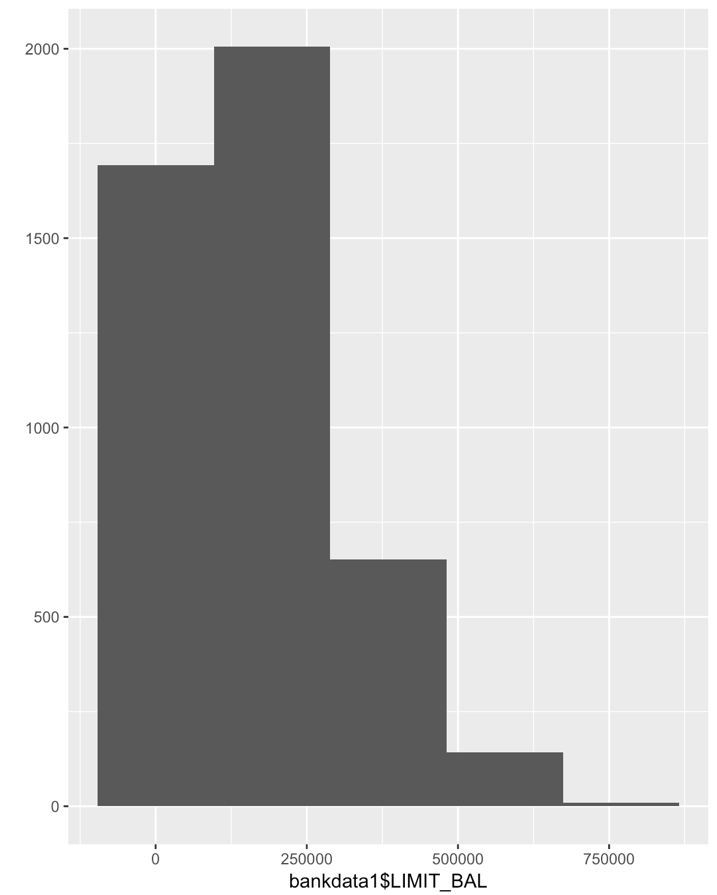
bankdata1$ID <- NULL

# plot the class distribution

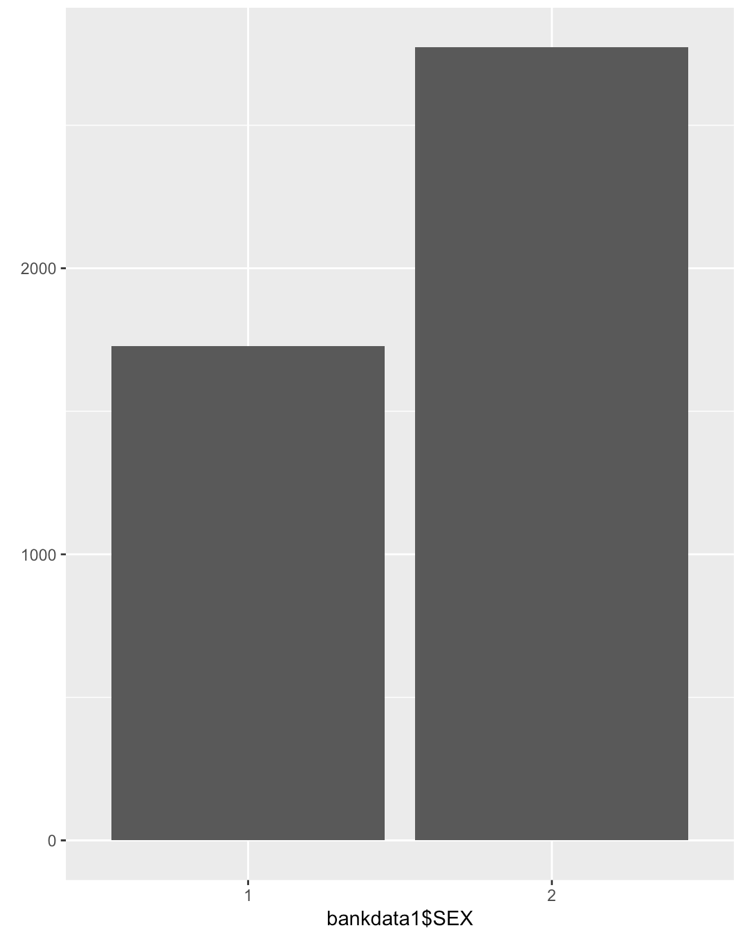
qplot(class, data=bankdata1, geom = "bar") + theme(axis.text.x = element\_text(angle = 90, hjust = 1))



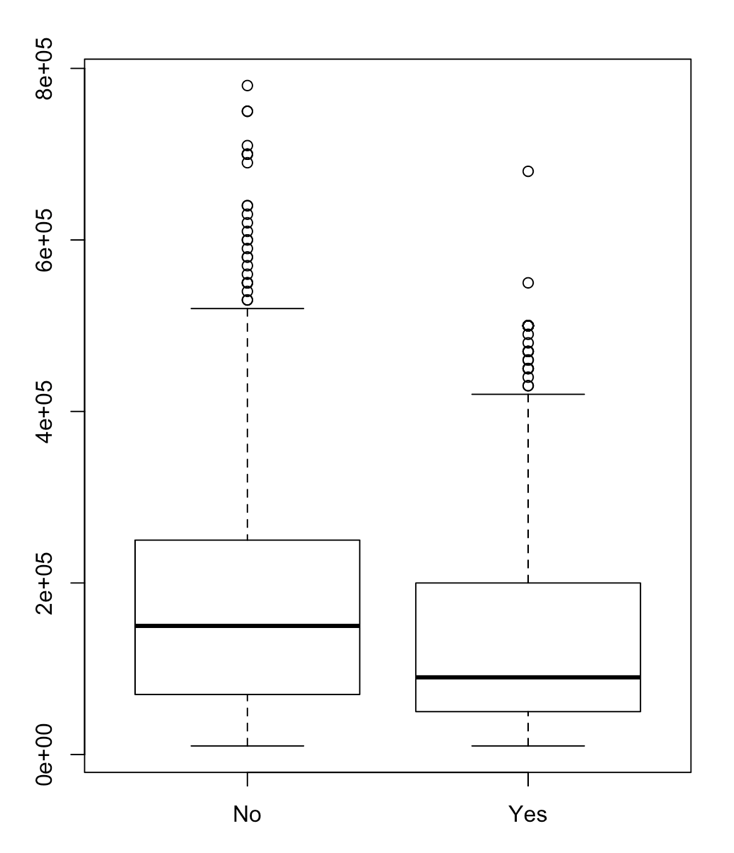
qplot(bankdata1$LIMIT\_BAL, bins = 5, margins = TRUE)



qplot(bankdata1$SEX)



boxplot(bankdata1$LIMIT\_BAL ~ bankdata1$class, bankdata1)



#create partition in data for training and testing

tdi <- createDataPartition(bankdata1$class, p=0.67, list = FALSE)

# Create Training Data as subset

trainingdata <- bankdata1[tdi,]

View(trainingdata)

# write the training data to a csv file

write.csv(trainingdata,"Desktop/620 - Data Mining/Project/training data.csv")

# Everything else not in training is test data. Note the - (minus)sign

testdata <- bankdata1[-tdi,]

View(testdata)

# write the test data to a csv file

write.csv(testdata,"Desktop/620 - Data Mining/Project/test data.csv")

# relevel training data to get "yes" as positive class

trainingdata$class <- relevel(trainingdata$class, "Yes")

summary(trainingdata$class)

# relevel test data to get "yes" as positive class

testdata$class <- relevel(testdata$class, "Yes")

summary(testdata$class)

#-------------------Building Classifiers---------------------#

# we will train and evaluate the model using 10-fold cross validation

trainingparameter <- trainControl(method = "repeatedcv", number = 10, sampling = "up")

#-------------C5.0 algorithm for Decision Tree Classifier-------------#

DecTree <- train(trainingdata[,-24], trainingdata$class,

method = "C5.0",

preProcess = c("nzv", "corr"),

trControl= trainingparameter,

na.action = na.omit)

DecTree

# make predictions on test set & see results

DecTreePred <- predict(DecTree, testdata, na.action = na.pass)

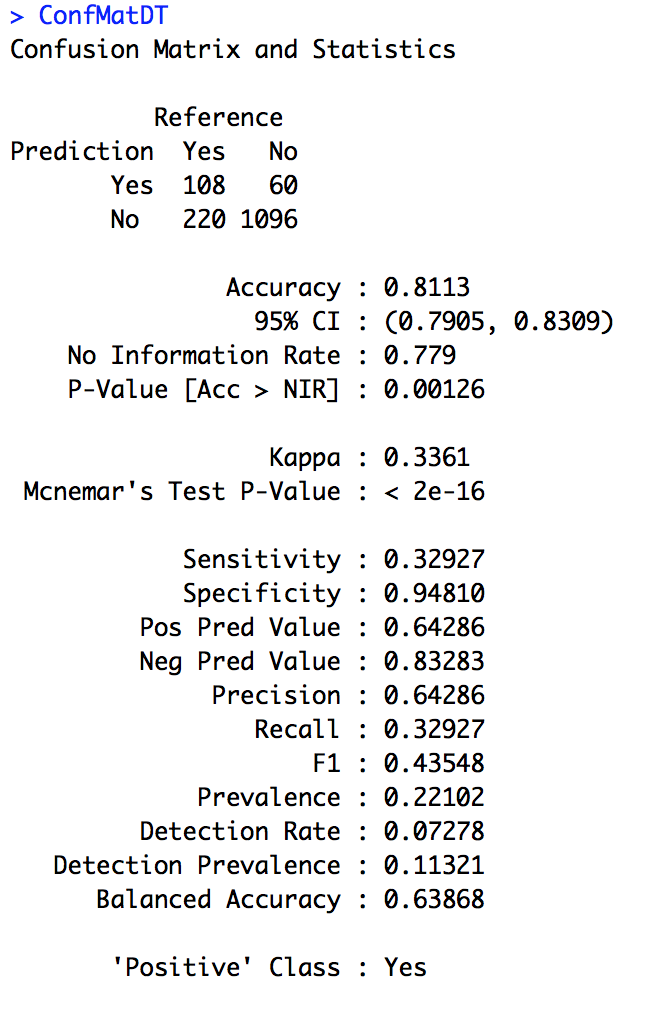
DecTreePred

# create confusion matrix & see results

ConfMatDT <-confusionMatrix(DecTreePred, testdata$class, mode = "everything")

ConfMatDT

t(ConfMatDT$table)



#-----------------------Naïve Bayes Classifer-----------------------#

NaiveBayes <- train(trainingdata[,-24], trainingdata$class,

method = "nb",

preProcess = c("nzv", "corr"),

trControl= trainingparameter,

na.action = na.omit)

NaiveBayes

# make predictions on test set & see results

NaiveBayesPred <-predict(NaiveBayes, testdata, na.action = na.pass)

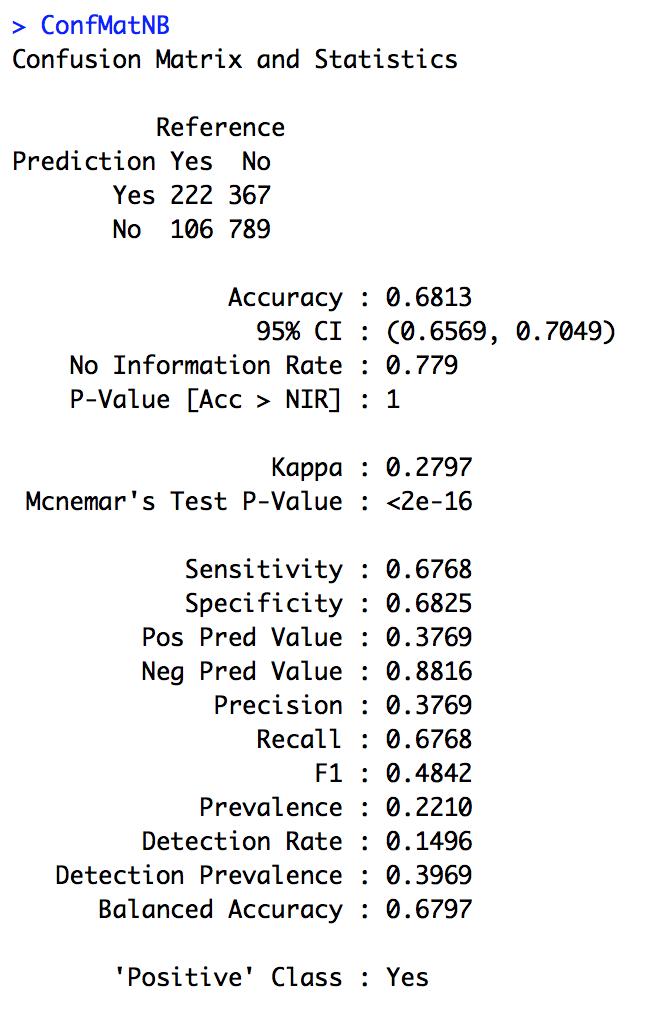
NaiveBayesPred

# create confusion matrix & see results

ConfMatNB <-confusionMatrix(NaiveBayesPred, testdata$class, mode = "everything")

ConfMatNB

t(ConfMatNB$table)



#---------------------Random Forest Classifier----------------------#

RF <- train(trainingdata[,-24], trainingdata$class,

method = "rf",

preProcess = c("nzv", "corr"),

trControl= trainingparameter)

RF

# make predictions on test set & see results

RFPred <-predict(RF, testdata)

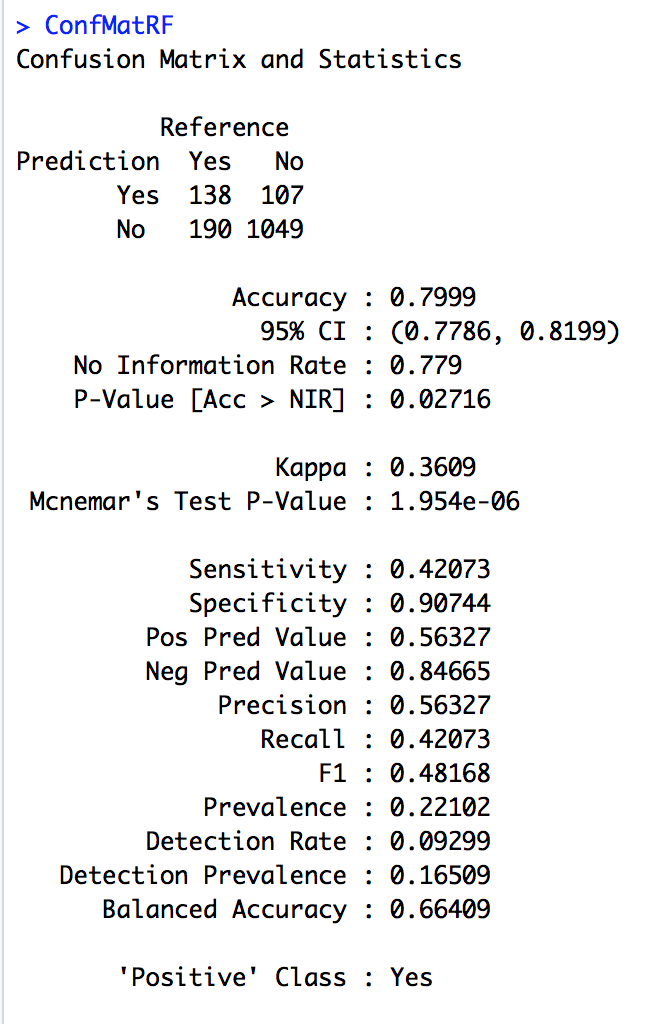
RFPred

# create confusion matrix & see results

ConfMatRF <-confusionMatrix(RFPred, testdata$class, mode = "everything")

ConfMatRF

t(ConfMatRF$table)



#---------------------------GBM Classifier---------------------------#

GBM <- train(trainingdata[,-24], trainingdata$class,

method = "gbm",

preProcess = c("nzv", "corr"),

trControl= trainingparameter)

GBM

# make predictions on test set & see results

GBMPred <-predict(GBM, testdata)

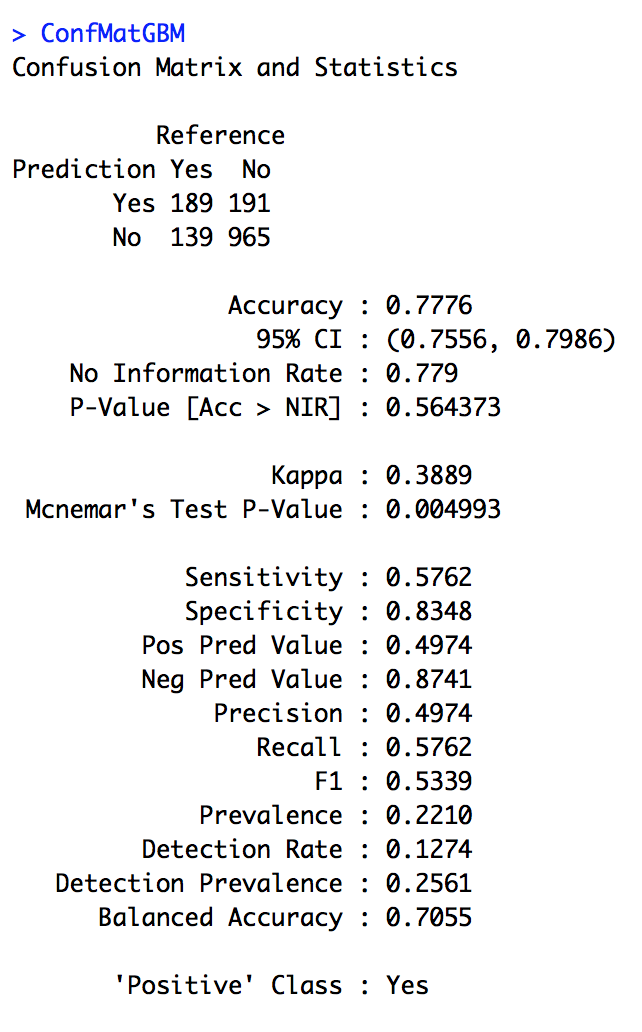
GBMPred

# create confusion matrix & see results

ConfMatGBM <-confusionMatrix(GBMPred, testdata$class, mode = "everything")

ConfMatGBM

t(ConfMatGBM$table)



*4x4 Matrix (Models vs Metrics)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **Sensitivity** | **Specificity** | **Precision** |
| **Decision Tree** | 0.8113 | 0.3293 | 0.9481 | 0.6429 |
| **Naïve Bayes** | 0.6813 | 0.6768 | 0.6825 | 0.3768 |
| **Random Forest** | 0.7999 | 0.4207 | 0.9074 | 0.5633 |
| **GBM** | 0.7776 | 0.5762 | 0.8348 | 0.4974 |

#-----------Comparing Models through Weighted F Measure---------------#

# Decision Tree model F measure #score get worse for recall

F1\_Score(testdata$class, DecTreePred)

FBeta\_Score(testdata$class, DecTreePred, beta = 0.5)

FBeta\_Score(testdata$class, DecTreePred, beta = 0.1)

FBeta\_Score(testdata$class, DecTreePred, beta = 1.1)

FBeta\_Score(testdata$class, DecTreePred, beta = 1.5)

# Naive Bayes model F measure #score gets better for recall

F1\_Score(testdata$class, NaiveBayesPred)

FBeta\_Score(testdata$class, NaiveBayesPred, beta = 0.5)

FBeta\_Score(testdata$class, NaiveBayesPred, beta = 0.1)

FBeta\_Score(testdata$class, NaiveBayesPred, beta = 1.1)

FBeta\_Score(testdata$class, NaiveBayesPred, beta = 1.5)

# Random Forest model F measure #score get worse for recall

F1\_Score(testdata$class, RFPred)

FBeta\_Score(testdata$class, RFPred, beta = 0.5)

FBeta\_Score(testdata$class, RFPred, beta = 0.1)

FBeta\_Score(testdata$class, RFPred, beta = 1.1)

FBeta\_Score(testdata$class, RFPred, beta = 1.5)

# GBM model F measure, #score get worse for recall

F1\_Score(testdata$class, GBMPred)

FBeta\_Score(testdata$class, GBMPred, beta = 0.5)

FBeta\_Score(testdata$class, GBMPred, beta = 0.1)

FBeta\_Score(testdata$class, GBMPred, beta = 1.1)

FBeta\_Score(testdata$class, GBMPred, beta = 1.5)

# header for classifier list

classifier <- c("C5.0","nb","rf", "gbm")

# since we want to give more weightage to recall the Beta should be greater than 1

# we consider the Beta = 1.1 since it gives higher value of FMeasure between 1.1 and 1.5

FMeasure <- c(FBeta\_Score(testdata$class, DecTreePred, beta = 1.1),

FBeta\_Score(testdata$class, NaiveBayesPred, beta = 1.1),

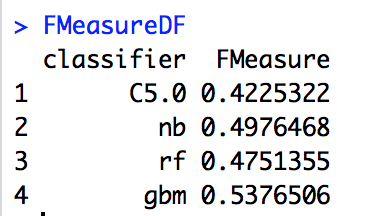
FBeta\_Score(testdata$class, RFPred, beta = 1.1),

FBeta\_Score(testdata$class, GBMPred, beta = 1.1))

# create dataframe to view classifiers with respective FMeasures for Beta = 1.1

FMeasureDF <- data.frame(classifier,FMeasure)

FMeasureDF



Gradient Boosting Classifier has the maximum weighted F measure; hence it is the recommended model.

#--------------Variable Importance----------------#

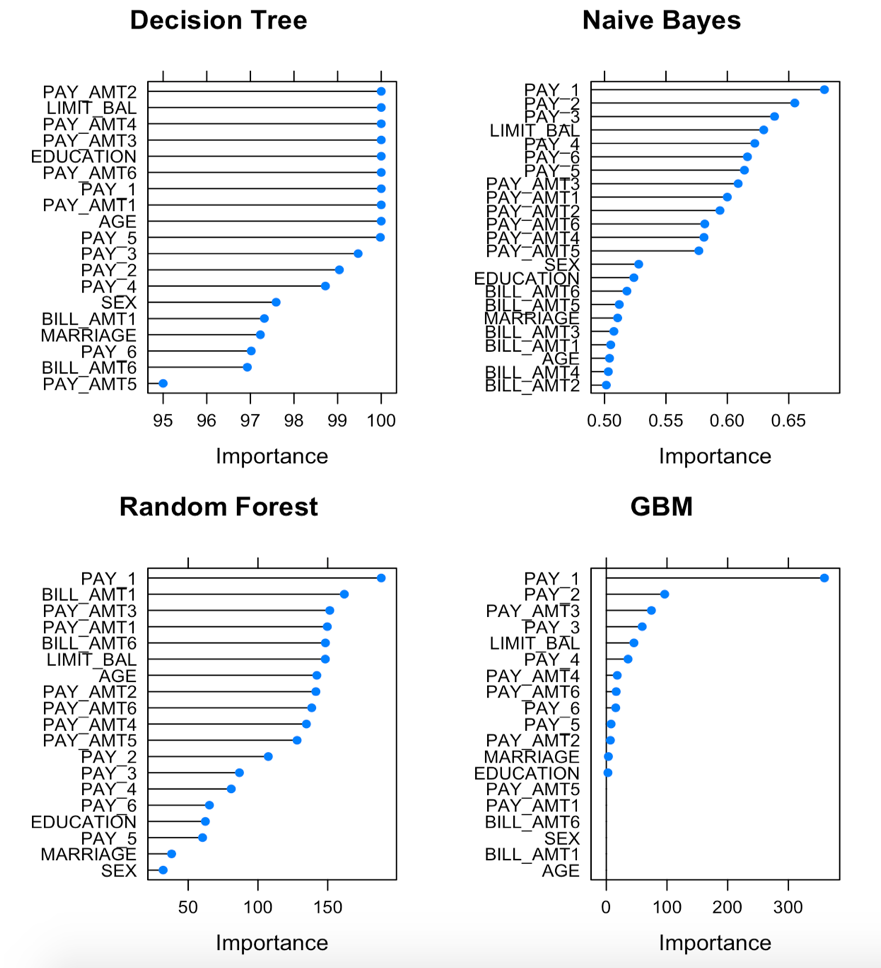
vimpdt <- plot(varImp(DecTree, scale = F), main = "Decision Tree")

vimpnb <- plot(varImp(NaiveBayes, scale = F), main = "Naive Bayes")

vimprf <- plot(varImp(RF, scale = F), main = "Random Forest")

vimpgbm <- plot(varImp(GBM, scale = F), main = "GBM")

grid.arrange(vimpdt, vimpnb, vimprf, vimpgbm)



#------------Finding Model Correlation and Building Ensemble Models--------------#

# create training control

EnsTrainParam <- trainControl(method="cv", number=5, summaryFunction = twoClassSummary,

savePredictions=TRUE, classProbs=TRUE)

# create train of models for finding correlation

AllModels <- caretList(class ~., data=trainingdata,

methodList=c("C5.0", "nb", "rf", "gbm"),

trControl = EnsTrainParam)

AllModels

ModResults <- resamples(AllModels)

ModResults$values

ModResults$metrics

summary(ModResults)

dotplot(ModResults)

# find model correlation

ModCorr <-modelCor(ModResults)

ModCorr

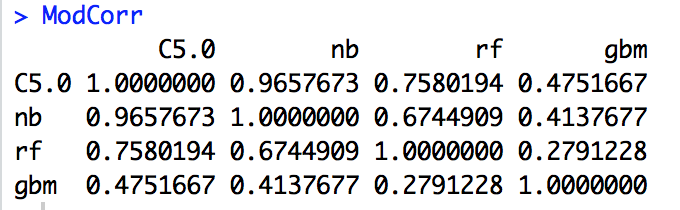
# Plot model correlation

splom(ModResults)

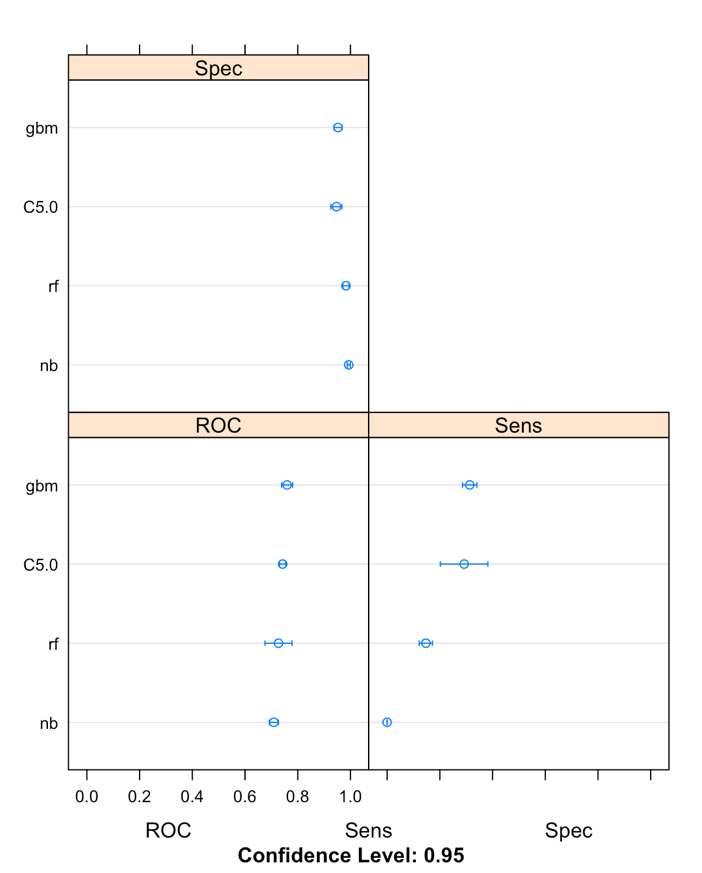
ModResults

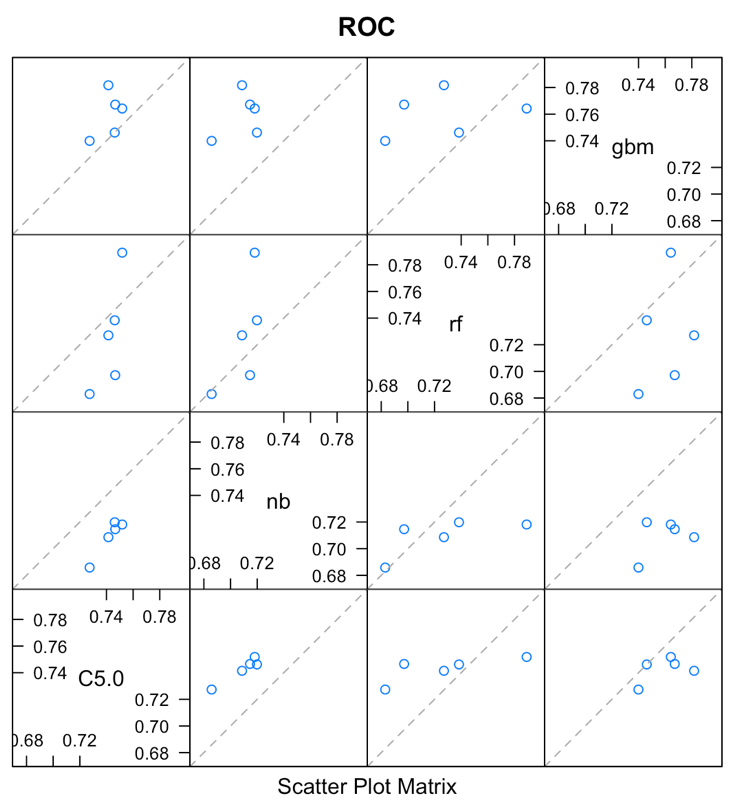
*Model Correlation Matrix*

Random Forest and Gradient Boosting are the least correlated models and Decision Tree and Naïve Bayes are the most correlated models.



*Model Results*





#----------------Stacked Models-------------------#

# Create stacked models for all models

EnsStackModel <- caretStack(AllModels, method = "C5.0", metric = "Sens",

trControl = trainControl(number = 5, summaryFunction = twoClassSummary, classProbs = TRUE))

print(EnsStackModel)

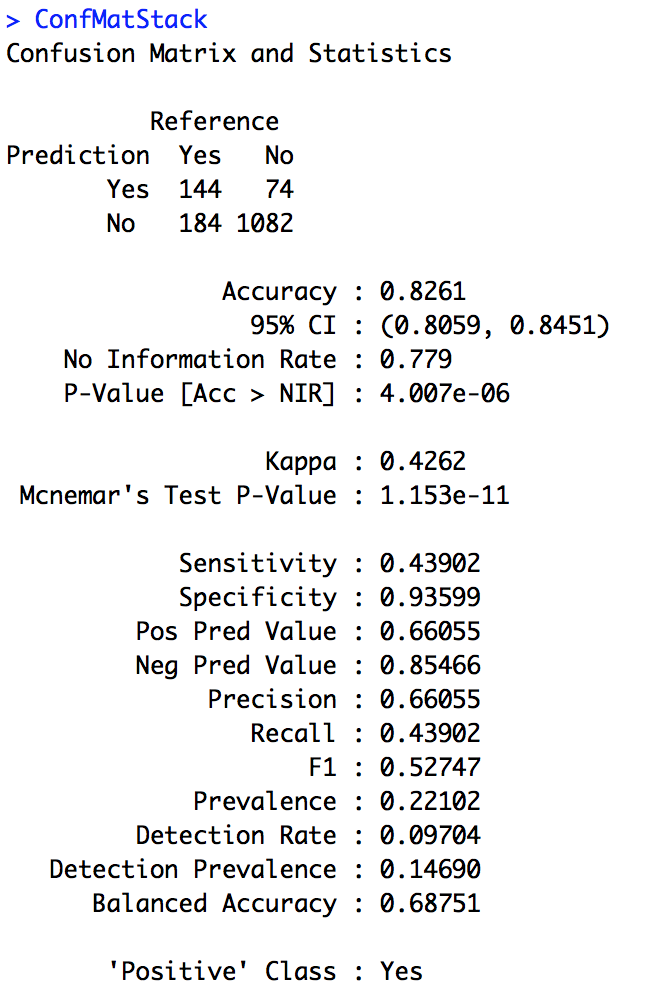
# predict & create confusion matrix

EnsStackPred <- predict(EnsStackModel, testdata)

ConfMatStack <-confusionMatrix(EnsStackPred, testdata$class, mode="everything")

ConfMatStack

t(ConfMatStack$table)



# create stacked models for Random Forest and Gradient boosting as they are the least correlated

LowCorr <- caretList(class ~., data=trainingdata,

methodList=c("rf", "gbm"),

trControl = EnsTrainParam)

EnsStackModel2 <- caretStack(LowCorr, method = "C5.0", metric="Sens",

trControl = trainControl(number = 5,

summaryFunction = twoClassSummary,

classProbs = TRUE))

print(EnsStackModel2)

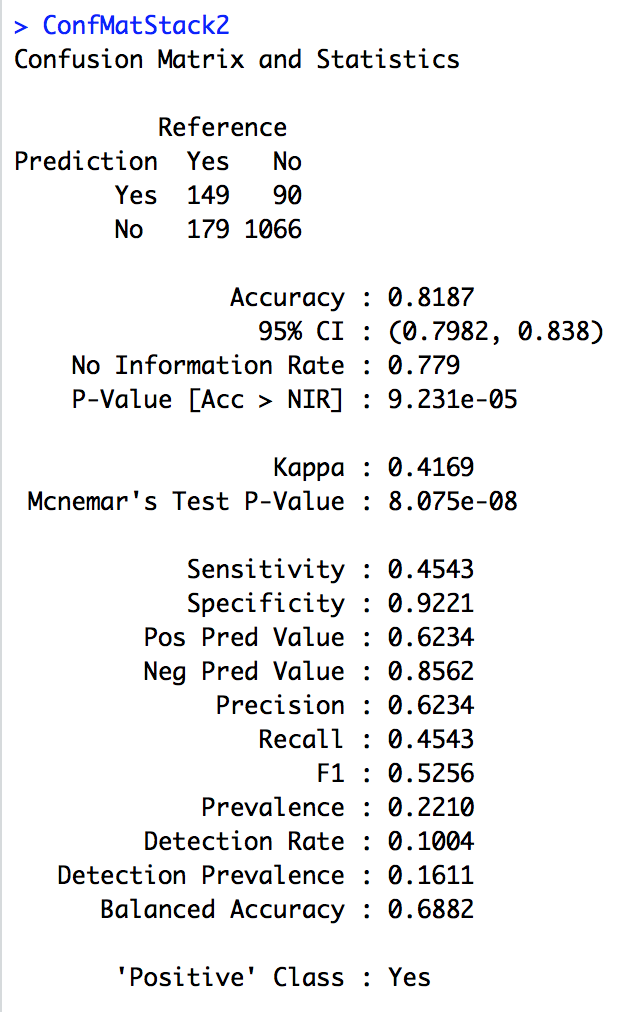
# predict & create confusion matrix

EnsStackPred2 <-predict(EnsStackModel2, testdata, na.action = na.omit)

ConfMatStack2 <-confusionMatrix(EnsStackPred2, testdata$class, mode="everything")

ConfMatStack2

t(ConfMatStack2$table)



#create stacked models for all models with up sampling

EnsStackModelUp <- caretStack(AllModels, method = "C5.0", metric="Sens",

trControl = trainControl(number = 5,

summaryFunction = twoClassSummary,

classProbs = TRUE, sampling = "up"))

print(EnsStackModelUp)

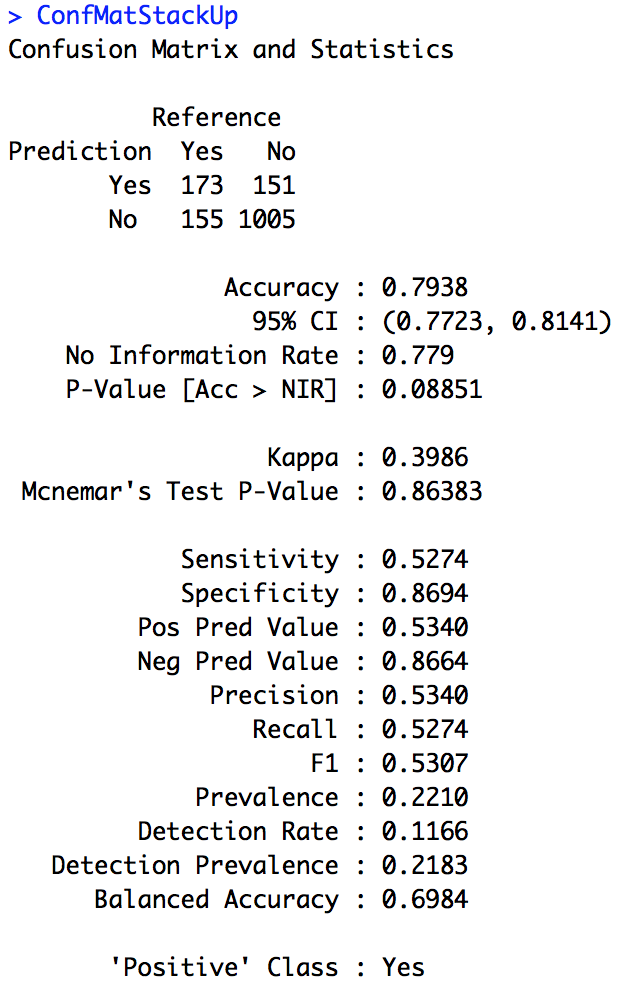
#predict & create confusion matrix

EnsStackPredUp <-predict(EnsStackModelUp, testdata, na.action = na.omit)

ConfMatStackUp <-confusionMatrix(EnsStackPredUp, testdata$class, mode="everything")

ConfMatStackUp

t(ConfMatStackUp$table)



#-------------------Ensemble Models------------------------#

# create ensemble model with sensitivity as optimization metric for all models

EnsModelSens <- caretEnsemble(AllModels, metric = "Sens",

trControl = trainControl(number = 5,

summaryFunction = twoClassSummary,

classProbs = TRUE))

summary(EnsModelSens)

# predict & create confusion matrix

EnsPredSens <-predict(EnsModelSens, testdata, na.action = na.omit)

ConfMatEMS <-confusionMatrix(EnsPredSens, testdata$class, mode="everything")

ConfMatEMS

t(ConfMatEMS$table)

# create ensemble model with sensitivity as optimization metric for least correlated models

EnsModelSens2 <- caretEnsemble(LowCorr, metric = "Sens",

trControl = trainControl(number = 5,

summaryFunction = twoClassSummary,

classProbs = TRUE))

summary(EnsModelSens2)

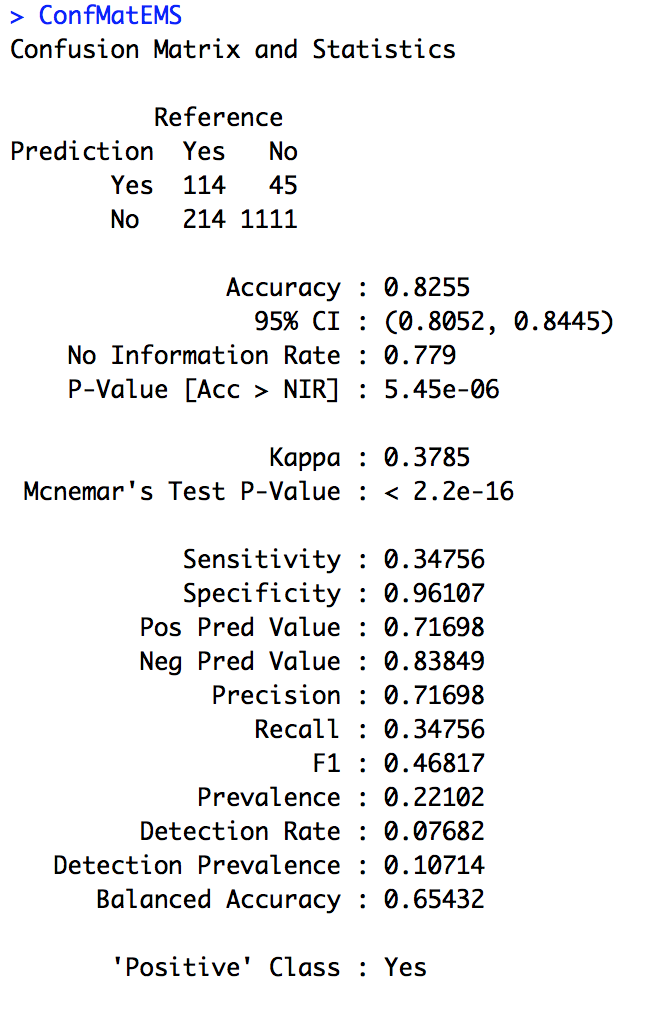
# predict & create confusion matrix

EnsPredSens2 <-predict(EnsModelSens2, testdata, na.action = na.omit)

ConfMatEMS2 <-confusionMatrix(EnsPredSens2, testdata$class, mode="everything")

ConfMatEMS2

t(ConfMatEMS2$table)



# improve with up sampling the ensemble model with accuracy as optimization metric for all models EnsModelSensUp <- caretEnsemble(AllModels, metric = "Sens",

trControl = trainControl(number = 5,

summaryFunction = twoClassSummary,

classProbs = TRUE, sampling = "up"))

summary(EnsModelSensUp)

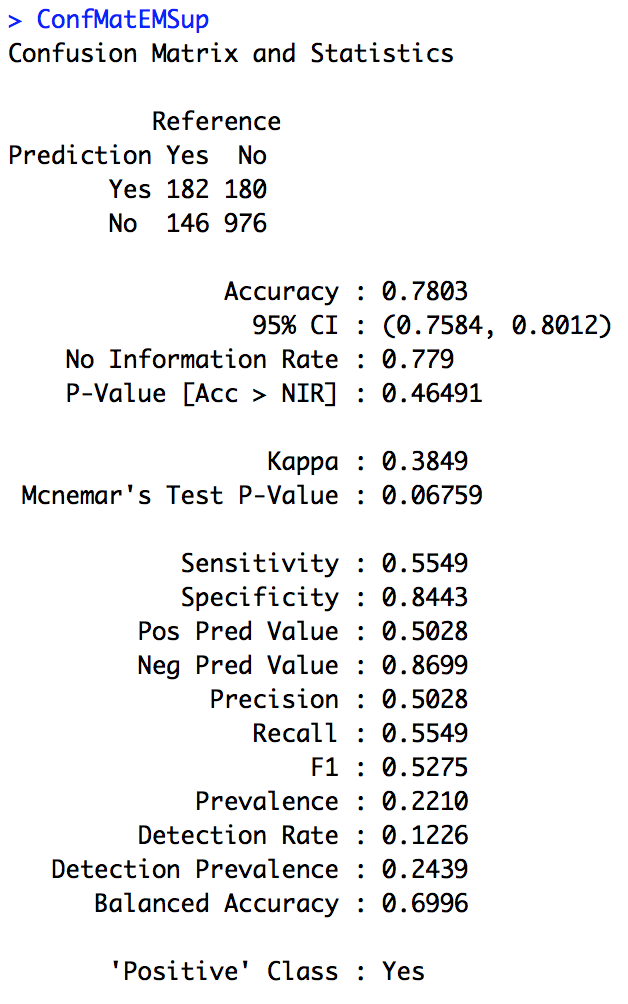
# predict & create confusion matrix

EnsPredSensUp <-predict(EnsModelSensUp, testdata, na.action = na.omit)

ConfMatEMSup <-confusionMatrix(EnsPredSensUp, testdata$class, mode="everything")

ConfMatEMSup

t(ConfMatEMSup$table)



#----------------------------Cost Analysis-----------------------------#

# create cost matrix

CostMatrix <- cbind(c(0,10), c(1,0))

t(CostMatrix)

# cost analysis for C5.0 algorithm

CostDecTree <- C5.0(class ~., trials = 5, data=trainingdata, cost=CostMatrix)

CostDecTree

summary(CostDecTree)

CostPred <- predict(CostDecTree, testdata)

CostPred

CostConfMat <-confusionMatrix(CostPred, testdata$class, mode = "everything")

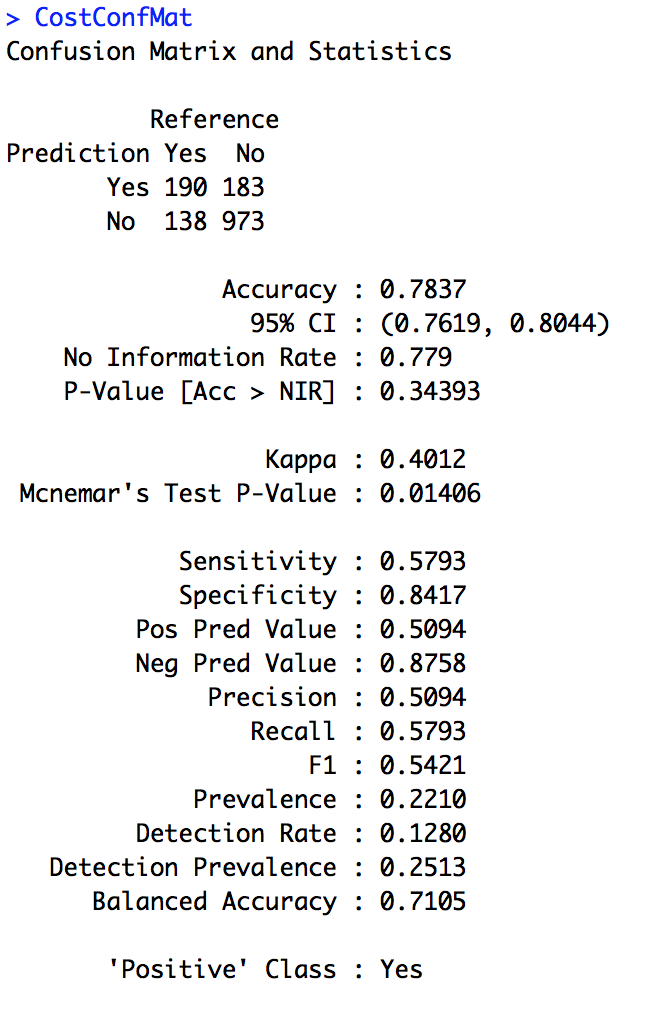
CostConfMat

t(CostConfMat$table)

# multiply cost and confusion matrix

TotCostDT <- CostMatrix\*CostConfMat$table

sum(TotCostDT)



Profit = 190\*10 – 190 – 183 = 1527

# cost analysis for Naive Bayes classifier

CostNaiveBayes <- naive\_bayes(class ~., trials = 5, data=trainingdata, cost=CostMatrix)

CostNaiveBayes

summary(CostNaiveBayes)

CostPredNB <- predict(CostNaiveBayes, testdata)

CostPredNB

CostConfMatNB <-confusionMatrix(CostPredNB, testdata$class, mode = "everything")

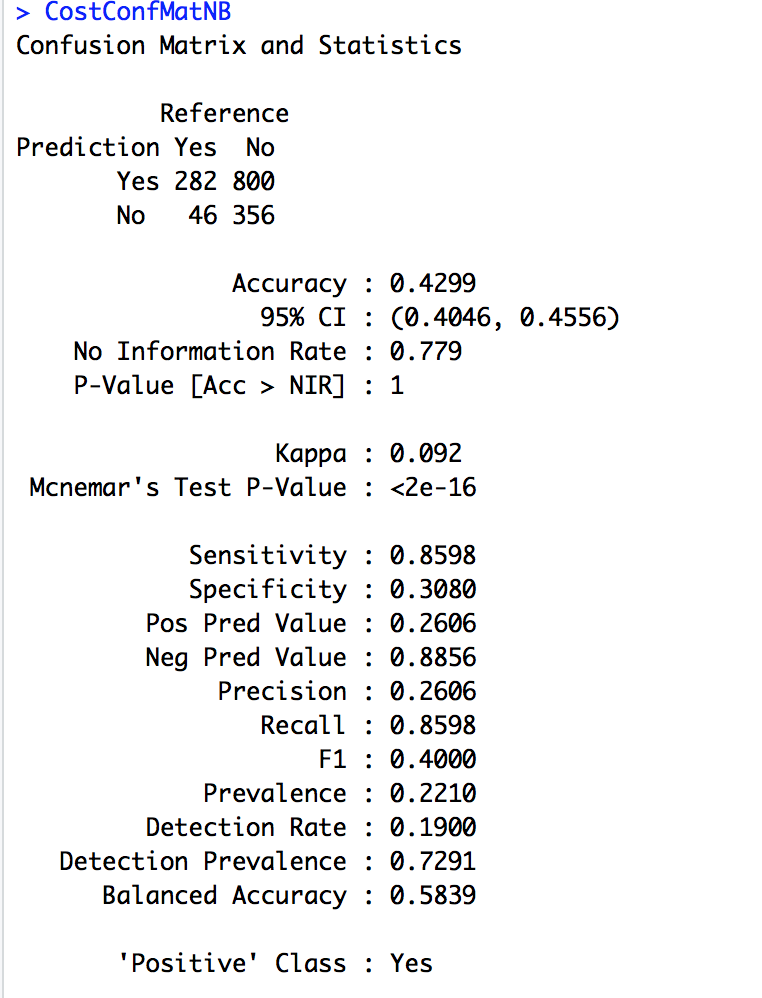
CostConfMatNB

t(CostConfMatNB$table)

#multiply cost and confusion matrix

TotCostNB <- CostMatrix\*CostConfMatNB$table

sum(TotCostNB)



Profit = 282\*10 – 282 – 800 = 1738

# cost analysis for Random Forest classifier

CostRndmFrst <- randomForest(class ~., trials = 5, data=trainingdata, cost=CostMatrix)

CostRndmFrst

summary(CostRndmFrst)

CostPredRF <- predict(CostRndmFrst, testdata)

CostPredRF

CostConfMatRF <-confusionMatrix(CostPredRF, testdata$class, mode = "everything")

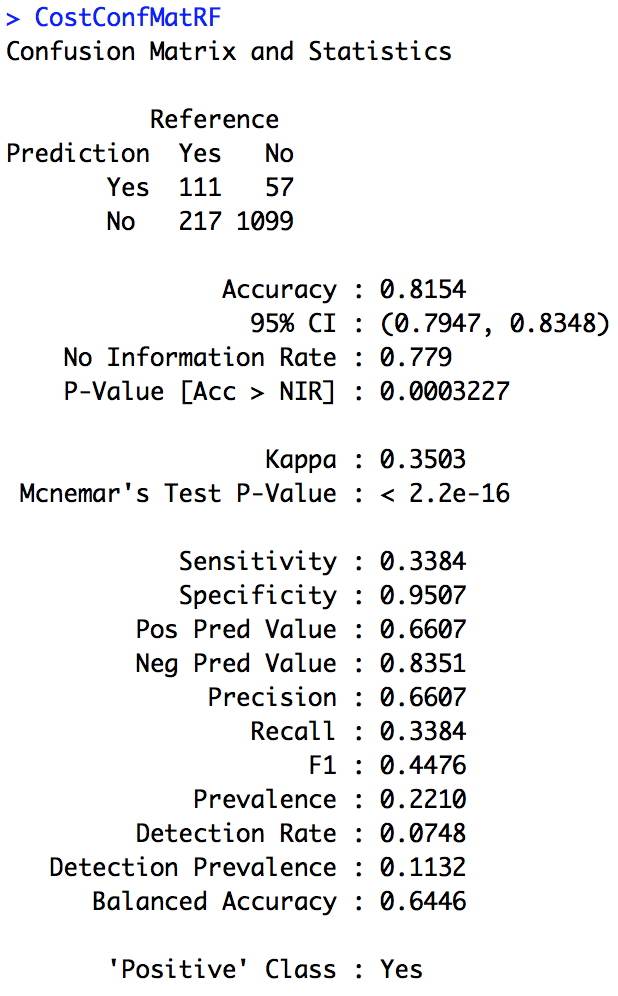
CostConfMatRF

t(CostConfMatRF$table)

#multiply cost and confusion matrix

TotCostRF <- CostMatrix\*CostConfMatRF$table

sum(TotCostRF)



Profit = 111\*10 – 111 – 57 = 942

#----------------------------End-----------------------------#