Data Mining & Machine Learning 1

Machine Learning Techniques in Financial Domain

Appendix for Code

Submitted to:

Michael Bradford

Submitted by:

Himanshu Rathee

20132689

Master of Science in Data Analytics

National College of Ireland

TABLE OF CONTENTS

[PART 1 : NAIVE BAYES MODELLING 2](#_Toc61170006)

[STEP 1 : Collecting data 2](#_Toc61170007)

[STEP 2 : Exploring and Preparing the data 2](#_Toc61170008)

[STEP 3 : Data preparation 8](#_Toc61170009)

[STEP 3.1 : Partioning the data into training and testing subsets 8](#_Toc61170010)

[STEP 3.1. : Removing Unbalancing from the dataset 10](#_Toc61170011)

[STEP 4 : Training Naive Bayes model 10](#_Toc61170012)

[STEP 5 : Evaluating Model Performance 10](#_Toc61170013)

[STEP 6 : Improving Model Performance 13](#_Toc61170014)

[PART 2 : RANDOM FOREST MODELLING 16](#_Toc61170015)

[STEP 1 - STEP 3 remains the same 16](#_Toc61170016)

[STEP 4 : Training the Random Forest Model 16](#_Toc61170017)

[STEP 5 : Evalutating the model 17](#_Toc61170018)

[STEP 6 : Improving the model performance 19](#_Toc61170019)

[STEP 5.1 : Further evaluating the best model by ROC curve 21](#_Toc61170020)

[STEP 6.2: COMPARING NAIVE BAYES AND RANDOM FOREST FOR THIS DATASET 23](#_Toc61170021)

[PART 1: LOGISTIC REGRESSION MODELLING 24](#_Toc61170022)

[STEP 1 : Collecting Data 24](#_Toc61170023)

[STEP 2 : Exploring and Preparing the Data 24](#_Toc61170024)

[STEP 3: Data Preparation 34](#_Toc61170025)

[STEP 3.2: Dealing with unbalanced data in the training data 35](#_Toc61170026)

[STEP 4: Model Training ( Logistic Regression) 36](#_Toc61170027)

[STEP 5: Evaluating Model Performance 37](#_Toc61170028)

[STEP 6: Improving Model Performance 44](#_Toc61170029)

[Part 1 : Decision Tree Modelling 48](#_Toc61170030)

[STEP 1 : Collecting Data 48](#_Toc61170031)

[STEP 2 : Exploring and Preparing the Data 50](#_Toc61170032)

[STEP 3 : Training a Model on the data 53](#_Toc61170033)

[STEP 4 : Evaluating Model Performance 55](#_Toc61170034)

[PART 2 : SUPPORT VECTOR MACHINE MODELLING 56](#_Toc61170035)

[STEP 1 till STEP 2.3 are as above used for Decision Tree Modelling 56](#_Toc61170036)

[STEP 2.4 : 56](#_Toc61170037)

[STEP 4 : Training a Model 57](#_Toc61170038)

[STEP 5 :Evaluating Model performance 57](#_Toc61170039)

[STEP 6 : Improving Model Performance 58](#_Toc61170040)

[STEP 6.1 : Training SVM model with vanilladot for comparison 58](#_Toc61170041)

[STEP 6.2 : Evaluating Kernel Model 58](#_Toc61170042)

Appendix 1: Credit Card Approval

## PART 1 : NAIVE BAYES MODELLING

## STEP 1 : Collecting data

#Setting the working directory where the datasets are stored.  
setwd("W:/DMML1/DMML\_1\_Project/DataSets/D2\_credit\_approval/Credit\_Card\_Approval")  
#Loading two files with application and record of credit of customers  
application <- read.csv("application\_record.csv")  
record <- read.csv("credit\_record.csv")  
print("The files are stored in application and record varibale as a dataframe")

## [1] "The files are stored in application and record varibale as a dataframe"

## STEP 2 : Exploring and Preparing the data

### STEP 2.1 : Checking the summary and structure of the datasets

#Summary and Structure of Application dataset  
print("The summary and structure of application data is:")

## [1] "The summary and structure of application data is:"

summary(application)

## ID CODE\_GENDER FLAG\_OWN\_CAR FLAG\_OWN\_REALTY   
## Min. :5008804 Length:438557 Length:438557 Length:438557   
## 1st Qu.:5609375 Class :character Class :character Class :character   
## Median :6047745 Mode :character Mode :character Mode :character   
## Mean :6022176   
## 3rd Qu.:6456971   
## Max. :7999952   
## CNT\_CHILDREN AMT\_INCOME\_TOTAL NAME\_INCOME\_TYPE NAME\_EDUCATION\_TYPE  
## Min. : 0.0000 Min. : 26100 Length:438557 Length:438557   
## 1st Qu.: 0.0000 1st Qu.: 121500 Class :character Class :character   
## Median : 0.0000 Median : 160780 Mode :character Mode :character   
## Mean : 0.4274 Mean : 187524   
## 3rd Qu.: 1.0000 3rd Qu.: 225000   
## Max. :19.0000 Max. :6750000   
## NAME\_FAMILY\_STATUS NAME\_HOUSING\_TYPE DAYS\_BIRTH DAYS\_EMPLOYED   
## Length:438557 Length:438557 Min. :-25201 Min. :-17531   
## Class :character Class :character 1st Qu.:-19483 1st Qu.: -3103   
## Mode :character Mode :character Median :-15630 Median : -1467   
## Mean :-15998 Mean : 60564   
## 3rd Qu.:-12514 3rd Qu.: -371   
## Max. : -7489 Max. :365243   
## FLAG\_MOBIL FLAG\_WORK\_PHONE FLAG\_PHONE FLAG\_EMAIL   
## Min. :1 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:1 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000   
## Median :1 Median :0.0000 Median :0.0000 Median :0.0000   
## Mean :1 Mean :0.2061 Mean :0.2878 Mean :0.1082   
## 3rd Qu.:1 3rd Qu.:0.0000 3rd Qu.:1.0000 3rd Qu.:0.0000   
## Max. :1 Max. :1.0000 Max. :1.0000 Max. :1.0000   
## OCCUPATION\_TYPE CNT\_FAM\_MEMBERS   
## Length:438557 Min. : 1.000   
## Class :character 1st Qu.: 2.000   
## Mode :character Median : 2.000   
## Mean : 2.194   
## 3rd Qu.: 3.000   
## Max. :20.000

str(application)

## 'data.frame': 438557 obs. of 18 variables:  
## $ ID : int 5008804 5008805 5008806 5008808 5008809 5008810 5008811 5008812 5008813 5008814 ...  
## $ CODE\_GENDER : chr "M" "M" "M" "F" ...  
## $ FLAG\_OWN\_CAR : chr "Y" "Y" "Y" "N" ...  
## $ FLAG\_OWN\_REALTY : chr "Y" "Y" "Y" "Y" ...  
## $ CNT\_CHILDREN : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ AMT\_INCOME\_TOTAL : num 427500 427500 112500 270000 270000 ...  
## $ NAME\_INCOME\_TYPE : chr "Working" "Working" "Working" "Commercial associate" ...  
## $ NAME\_EDUCATION\_TYPE: chr "Higher education" "Higher education" "Secondary / secondary special" "Secondary / secondary special" ...  
## $ NAME\_FAMILY\_STATUS : chr "Civil marriage" "Civil marriage" "Married" "Single / not married" ...  
## $ NAME\_HOUSING\_TYPE : chr "Rented apartment" "Rented apartment" "House / apartment" "House / apartment" ...  
## $ DAYS\_BIRTH : int -12005 -12005 -21474 -19110 -19110 -19110 -19110 -22464 -22464 -22464 ...  
## $ DAYS\_EMPLOYED : int -4542 -4542 -1134 -3051 -3051 -3051 -3051 365243 365243 365243 ...  
## $ FLAG\_MOBIL : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ FLAG\_WORK\_PHONE : int 1 1 0 0 0 0 0 0 0 0 ...  
## $ FLAG\_PHONE : int 0 0 0 1 1 1 1 0 0 0 ...  
## $ FLAG\_EMAIL : int 0 0 0 1 1 1 1 0 0 0 ...  
## $ OCCUPATION\_TYPE : chr "" "" "Security staff" "Sales staff" ...  
## $ CNT\_FAM\_MEMBERS : num 2 2 2 1 1 1 1 1 1 1 ...

#Summary and structure of Credit Record dataset  
print("The summary and structure of credit data is:")

## [1] "The summary and structure of credit data is:"

summary(record)

## ID MONTHS\_BALANCE STATUS   
## Min. :5001711 Min. :-60.00 Length:1048575   
## 1st Qu.:5023644 1st Qu.:-29.00 Class :character   
## Median :5062104 Median :-17.00 Mode :character   
## Mean :5068286 Mean :-19.14   
## 3rd Qu.:5113856 3rd Qu.: -7.00   
## Max. :5150487 Max. : 0.00

str(record)

## 'data.frame': 1048575 obs. of 3 variables:  
## $ ID : int 5001711 5001711 5001711 5001711 5001712 5001712 5001712 5001712 5001712 5001712 ...  
## $ MONTHS\_BALANCE: int 0 -1 -2 -3 0 -1 -2 -3 -4 -5 ...  
## $ STATUS : chr "X" "0" "0" "0" ...

#Checking number of unique entries  
print(paste0("Number of Unique records in Application dataset : ",length(unique(application$ID))))

## [1] "Number of Unique records in Application dataset : 438510"

print(paste0("Number of Unique records in Credit Record dataset : ",length(unique(record$ID))))

## [1] "Number of Unique records in Credit Record dataset : 45985"

### STEP 2.2 : Cleaning the application data

It is evident from the unique ID of both the datasets that the unique id count in application data is 4,38,510 and the unique id count in record data is 45,985. Therefore the duplicated entries from the application dataset needs to be removed.

#tidyverse used for duplicate function  
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()

#Removing duplicate rows except for the ID because ID is unique in each row  
application <- application[!duplicated(application[,-1]),]  
str(application)

## 'data.frame': 90085 obs. of 18 variables:  
## $ ID : int 5008804 5008806 5008808 5008812 5008815 5008819 5008825 5008830 5008834 5008836 ...  
## $ CODE\_GENDER : chr "M" "M" "F" "F" ...  
## $ FLAG\_OWN\_CAR : chr "Y" "Y" "N" "N" ...  
## $ FLAG\_OWN\_REALTY : chr "Y" "Y" "Y" "Y" ...  
## $ CNT\_CHILDREN : int 0 0 0 0 0 0 0 0 1 3 ...  
## $ AMT\_INCOME\_TOTAL : num 427500 112500 270000 283500 270000 ...  
## $ NAME\_INCOME\_TYPE : chr "Working" "Working" "Commercial associate" "Pensioner" ...  
## $ NAME\_EDUCATION\_TYPE: chr "Higher education" "Secondary / secondary special" "Secondary / secondary special" "Higher education" ...  
## $ NAME\_FAMILY\_STATUS : chr "Civil marriage" "Married" "Single / not married" "Separated" ...  
## $ NAME\_HOUSING\_TYPE : chr "Rented apartment" "House / apartment" "House / apartment" "House / apartment" ...  
## $ DAYS\_BIRTH : int -12005 -21474 -19110 -22464 -16872 -17778 -10669 -10031 -10968 -12689 ...  
## $ DAYS\_EMPLOYED : int -4542 -1134 -3051 365243 -769 -1194 -1103 -1469 -1620 -1163 ...  
## $ FLAG\_MOBIL : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ FLAG\_WORK\_PHONE : int 1 0 0 0 1 0 0 0 0 0 ...  
## $ FLAG\_PHONE : int 0 0 1 0 1 0 0 1 0 0 ...  
## $ FLAG\_EMAIL : int 0 0 1 0 1 0 0 0 0 0 ...  
## $ OCCUPATION\_TYPE : chr "" "Security staff" "Sales staff" "" ...  
## $ CNT\_FAM\_MEMBERS : num 2 2 1 1 2 2 2 2 2 5 ...

#Checking if there are any duplicate values still left in the application dataset  
print("The number of duplciate enteries in the application dataset are: ")

## [1] "The number of duplciate enteries in the application dataset are: "

sum(duplicated(application[,-1]))

## [1] 0

### STEP 2.3 : Cleaning the Record data

X = Paid dues daily C = Debt is clear 0 = dues pending from 0-29 days 1 = dues pending from 30-59 days 2 = dues pending from 60-89 days 3 = dues pending from 90-119 days 4 = dues pending from

#Setting the status of each record in the record dataset  
record$STATUS[record$STATUS=="X"] <- "1"  
record$STATUS[record$STATUS=="C"] <- "1"  
record$STATUS[record$STATUS=="0"] <- "1"  
record$STATUS[record$STATUS=="1"] <- "1"  
record$STATUS[record$STATUS=="2"] <- "0"  
record$STATUS[record$STATUS=="3"] <- "0"  
record$STATUS[record$STATUS=="4"] <- "0"  
record$STATUS[record$STATUS=="5"] <- "0"  
str(record)

## 'data.frame': 1048575 obs. of 3 variables:  
## $ ID : int 5001711 5001711 5001711 5001711 5001712 5001712 5001712 5001712 5001712 5001712 ...  
## $ MONTHS\_BALANCE: int 0 -1 -2 -3 0 -1 -2 -3 -4 -5 ...  
## $ STATUS : chr "1" "1" "1" "1" ...

#Setting outcome variable as categorical factor  
record$STATUS <- factor(record$STATUS,levels = c(0,1),labels = c("Not Approved","Approved"))

### STEP 2.4 : Combining the datasets basis their Unique ID’s

library(Amelia)

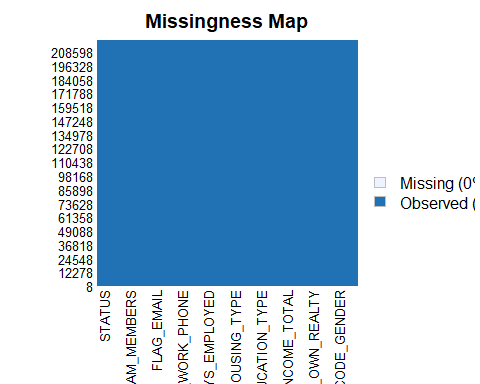
## Loading required package: Rcpp

## ##   
## ## Amelia II: Multiple Imputation  
## ## (Version 1.7.6, built: 2019-11-24)  
## ## Copyright (C) 2005-2021 James Honaker, Gary King and Matthew Blackwell  
## ## Refer to http://gking.harvard.edu/amelia/ for more information  
## ##

#Merging both the datasets based on the Unique ID's  
final <- merge(application,record,by="ID")  
str(final)

## 'data.frame': 219173 obs. of 20 variables:  
## $ ID : int 5008804 5008804 5008804 5008804 5008804 5008804 5008804 5008804 5008804 5008804 ...  
## $ CODE\_GENDER : chr "M" "M" "M" "M" ...  
## $ FLAG\_OWN\_CAR : chr "Y" "Y" "Y" "Y" ...  
## $ FLAG\_OWN\_REALTY : chr "Y" "Y" "Y" "Y" ...  
## $ CNT\_CHILDREN : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ AMT\_INCOME\_TOTAL : num 427500 427500 427500 427500 427500 ...  
## $ NAME\_INCOME\_TYPE : chr "Working" "Working" "Working" "Working" ...  
## $ NAME\_EDUCATION\_TYPE: chr "Higher education" "Higher education" "Higher education" "Higher education" ...  
## $ NAME\_FAMILY\_STATUS : chr "Civil marriage" "Civil marriage" "Civil marriage" "Civil marriage" ...  
## $ NAME\_HOUSING\_TYPE : chr "Rented apartment" "Rented apartment" "Rented apartment" "Rented apartment" ...  
## $ DAYS\_BIRTH : int -12005 -12005 -12005 -12005 -12005 -12005 -12005 -12005 -12005 -12005 ...  
## $ DAYS\_EMPLOYED : int -4542 -4542 -4542 -4542 -4542 -4542 -4542 -4542 -4542 -4542 ...  
## $ FLAG\_MOBIL : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ FLAG\_WORK\_PHONE : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ FLAG\_PHONE : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ FLAG\_EMAIL : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ OCCUPATION\_TYPE : chr "" "" "" "" ...  
## $ CNT\_FAM\_MEMBERS : num 2 2 2 2 2 2 2 2 2 2 ...  
## $ MONTHS\_BALANCE : int -1 -11 -6 -5 -3 -13 -9 -10 -15 -7 ...  
## $ STATUS : Factor w/ 2 levels "Not Approved",..: 2 2 2 2 2 2 2 2 2 2 ...

#Checking if there are any Missing Values in the dataset  
missmap(final)

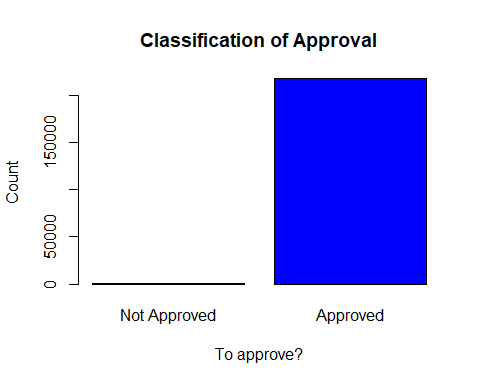


### STEP 2.5 : Visualizing the dataset

library(GGally)

## Registered S3 method overwritten by 'GGally':  
## method from   
## +.gg ggplot2

plot(final$STATUS,main= "Classification of Approval",ylab="Count",xlab="To approve?",col="blue")



#The number of Approved and Not approved applications in the dataset are  
as.data.frame(table(final$STATUS))

## Var1 Freq  
## 1 Not Approved 1039  
## 2 Approved 218134

### STEP 2.6 : Changing variables

#Factorizing all the character variables because for balancing the dataset all the varibales needs to be either numeric or factors  
final$CODE\_GENDER <- as.factor(final$CODE\_GENDER)  
final$FLAG\_OWN\_CAR <- as.factor(final$FLAG\_OWN\_CAR)  
final$FLAG\_OWN\_REALTY <- as.factor(final$FLAG\_OWN\_REALTY)  
final$NAME\_INCOME\_TYPE <- as.factor(final$NAME\_INCOME\_TYPE)  
final$NAME\_EDUCATION\_TYPE <- as.factor(final$NAME\_EDUCATION\_TYPE)  
final$NAME\_FAMILY\_STATUS <- as.factor(final$NAME\_FAMILY\_STATUS)  
final$NAME\_HOUSING\_TYPE <- as.factor(final$NAME\_HOUSING\_TYPE)  
final$OCCUPATION\_TYPE <- as.factor(final$OCCUPATION\_TYPE)

## STEP 3 : Data preparation

## STEP 3.1 : Partioning the data into training and testing subsets

library(caTools)  
library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

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

index <- createDataPartition(y=final$STATUS,p=0.75,list = FALSE)  
training <- final[index,]  
testing <- final[-index,]  
print("Dimensions of the training dataset are: ")

## [1] "Dimensions of the training dataset are: "

dim(training)

## [1] 164381 20

print("Dimensions of the testing dataset are: ")

## [1] "Dimensions of the testing dataset are: "

dim(testing)

## [1] 54792 20

print("The percentage of Approved and Not Approved fields in the training dataset are: ")

## [1] "The percentage of Approved and Not Approved fields in the training dataset are: "

prop.table(table(training$STATUS))

##   
## Not Approved Approved   
## 0.004745074 0.995254926

print("The percentage of Approved and Not Approved fields in the testing dataset are: ")

## [1] "The percentage of Approved and Not Approved fields in the testing dataset are: "

prop.table(table(testing$STATUS))

##   
## Not Approved Approved   
## 0.004726967 0.995273033

print("Number of approved and not approved values in the training data are:")

## [1] "Number of approved and not approved values in the training data are:"

as.data.frame(table(training$STATUS))

## Var1 Freq  
## 1 Not Approved 780  
## 2 Approved 163601

## STEP 3.1. : Removing Unbalancing from the dataset

set.seed(101)  
options(scipen=999)  
library(DMwR)

## Loading required package: grid

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

#Using SMOTE technique to balance the dataset  
balanced\_train <- SMOTE(STATUS~.,data=training,perc.over = 5500,k=2,perc.under = 300)  
#The number of approved and not approved counts after balancing the data   
as.data.frame(table(balanced\_train$STATUS))

## Var1 Freq  
## 1 Not Approved 43680  
## 2 Approved 128700

## STEP 4 : Training Naive Bayes model

library(e1071)  
#Building a model on the training data   
train\_classifier <- balanced\_train[,-20]  
train\_label <- balanced\_train$STATUS  
approval\_classifier <- naiveBayes(train\_classifier,train\_label)

## STEP 5 : Evaluating Model Performance

library(gmodels)  
library(caret)  
#Running the trained model on the testing data  
as.data.frame(table(testing$STATUS))

## Var1 Freq  
## 1 Not Approved 259  
## 2 Approved 54533

test\_classifier <- testing[,-20]  
test\_label <- testing$STATUS  
approval\_predict <- predict(approval\_classifier,test\_classifier)  
#Cross Table for evaluating the performance  
CrossTable(approval\_predict,test\_label,prop.chisq = F,prop.t=F,  
 dnn = c("predicted","actual"))

##   
##   
## Cell Contents  
## |-------------------------|  
## | N |  
## | N / Row Total |  
## | N / Col Total |  
## |-------------------------|  
##   
##   
## Total Observations in Table: 54792   
##   
##   
## | actual   
## predicted | Not Approved | Approved | Row Total |   
## -------------|--------------|--------------|--------------|  
## Not Approved | 56 | 8440 | 8496 |   
## | 0.007 | 0.993 | 0.155 |   
## | 0.216 | 0.155 | |   
## -------------|--------------|--------------|--------------|  
## Approved | 203 | 46093 | 46296 |   
## | 0.004 | 0.996 | 0.845 |   
## | 0.784 | 0.845 | |   
## -------------|--------------|--------------|--------------|  
## Column Total | 259 | 54533 | 54792 |   
## | 0.005 | 0.995 | |   
## -------------|--------------|--------------|--------------|  
##   
##

print("The accuracy of the simpel Naive Bayes model with laplace = 0 is : ")

## [1] "The accuracy of the simpel Naive Bayes model with laplace = 0 is : "

accuracy <- table(test\_label,approval\_predict)  
sum(diag(accuracy))/sum(accuracy)

## [1] 0.842258

#Evaluating the model on the training data itself.  
approval\_predicttrain <- predict(approval\_classifier,train\_classifier)  
#Cross Table for evaluating the performance  
CrossTable(approval\_predicttrain,train\_label,prop.chisq = F,prop.t=F,  
 dnn = c("predicted","actual"))

##   
##   
## Cell Contents  
## |-------------------------|  
## | N |  
## | N / Row Total |  
## | N / Col Total |  
## |-------------------------|  
##   
##   
## Total Observations in Table: 172380   
##   
##   
## | actual   
## predicted | Not Approved | Approved | Row Total |   
## -------------|--------------|--------------|--------------|  
## Not Approved | 36082 | 20391 | 56473 |   
## | 0.639 | 0.361 | 0.328 |   
## | 0.826 | 0.158 | |   
## -------------|--------------|--------------|--------------|  
## Approved | 7598 | 108309 | 115907 |   
## | 0.066 | 0.934 | 0.672 |   
## | 0.174 | 0.842 | |   
## -------------|--------------|--------------|--------------|  
## Column Total | 43680 | 128700 | 172380 |   
## | 0.253 | 0.747 | |   
## -------------|--------------|--------------|--------------|  
##   
##

print("The accuracy of the simpel Naive Bayes model with laplace = 0 is : ")

## [1] "The accuracy of the simpel Naive Bayes model with laplace = 0 is : "

accuracyabc <- table(train\_label,approval\_predicttrain)  
sum(diag(accuracyabc))/sum(accuracyabc)

## [1] 0.837632

#Confusion matrix for further evaluatind model's performance  
  
confusionMatrix(approval\_predict,test\_label)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Not Approved Approved  
## Not Approved 56 8440  
## Approved 203 46093  
##   
## Accuracy : 0.8423   
## 95% CI : (0.8392, 0.8453)   
## No Information Rate : 0.9953   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.0037   
##   
## Mcnemar's Test P-Value : <0.0000000000000002  
##   
## Sensitivity : 0.216216   
## Specificity : 0.845231   
## Pos Pred Value : 0.006591   
## Neg Pred Value : 0.995615   
## Prevalence : 0.004727   
## Detection Rate : 0.001022   
## Detection Prevalence : 0.155059   
## Balanced Accuracy : 0.530724   
##   
## 'Positive' Class : Not Approved   
##

## STEP 6 : Improving Model Performance

#Training the model with laplace value = 1   
approval\_classifier1 <- naiveBayes(train\_classifier,train\_label,laplace=1)  
#Running the model on testing subset and evaluating it's performance  
approval\_predict1 <- predict(approval\_classifier1,test\_classifier)  
accuracy1 <- table(test\_label,approval\_predict1)  
print("The accuracy of the simpel Naive Bayes model with laplace = 1 is : ")

## [1] "The accuracy of the simpel Naive Bayes model with laplace = 1 is : "

sum(diag(accuracy1))/sum(accuracy1)

## [1] 0.8428603

### STEP 6.1: Balancing the testing data

#Balancing the testing data as well and then prediction the values  
as.data.frame(table(testing$STATUS))

## Var1 Freq  
## 1 Not Approved 259  
## 2 Approved 54533

balanced\_test <- SMOTE(STATUS~.,data=testing,perc.over = 4500,k=2,perc.under = 300)  
as.data.frame(table(balanced\_test$STATUS))

## Var1 Freq  
## 1 Not Approved 11914  
## 2 Approved 34965

balanced\_test\_classifier <- balanced\_test[,-20]  
balanced\_test\_label <- balanced\_test$STATUS  
balanced\_approval\_predict <- predict(approval\_classifier1,balanced\_test\_classifier)  
confusionMatrix(balanced\_approval\_predict,balanced\_test\_label)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Not Approved Approved  
## Not Approved 6243 5332  
## Approved 5671 29633  
##   
## Accuracy : 0.7653   
## 95% CI : (0.7614, 0.7691)   
## No Information Rate : 0.7459   
## P-Value [Acc > NIR] : < 0.00000000000000022  
##   
## Kappa : 0.375   
##   
## Mcnemar's Test P-Value : 0.001272   
##   
## Sensitivity : 0.5240   
## Specificity : 0.8475   
## Pos Pred Value : 0.5394   
## Neg Pred Value : 0.8394   
## Prevalence : 0.2541   
## Detection Rate : 0.1332   
## Detection Prevalence : 0.2469   
## Balanced Accuracy : 0.6858   
##   
## 'Positive' Class : Not Approved   
##

bfscore <- confusionMatrix(balanced\_approval\_predict,balanced\_test\_label,mode="prec\_recall")  
print("The precision of the naive bayes is: ")

## [1] "The precision of the naive bayes is: "

bfscore$byClass["Precision"]

## Precision   
## 0.5393521

bfscore$byClass["Recall"]

## Recall   
## 0.5240054

bfscore$byClass["F1"]

## F1   
## 0.531568

### STEP 5.1 : Further Evaluating the performance of the best model

Evaluating model with laplace =1

library(caret)  
confusionMatrix(approval\_predict1,test\_label)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Not Approved Approved  
## Not Approved 56 8407  
## Approved 203 46126  
##   
## Accuracy : 0.8429   
## 95% CI : (0.8398, 0.8459)   
## No Information Rate : 0.9953   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.0037   
##   
## Mcnemar's Test P-Value : <0.0000000000000002  
##   
## Sensitivity : 0.216216   
## Specificity : 0.845836   
## Pos Pred Value : 0.006617   
## Neg Pred Value : 0.995618   
## Prevalence : 0.004727   
## Detection Rate : 0.001022   
## Detection Prevalence : 0.154457   
## Balanced Accuracy : 0.531026   
##   
## 'Positive' Class : Not Approved   
##

# F score of the model  
fscore <- confusionMatrix(approval\_predict1,test\_label,mode="prec\_recall")  
print("The precision of the naive bayes is: ")

## [1] "The precision of the naive bayes is: "

fscore$byClass["Precision"]

## Precision   
## 0.006617039

fscore$byClass["Recall"]

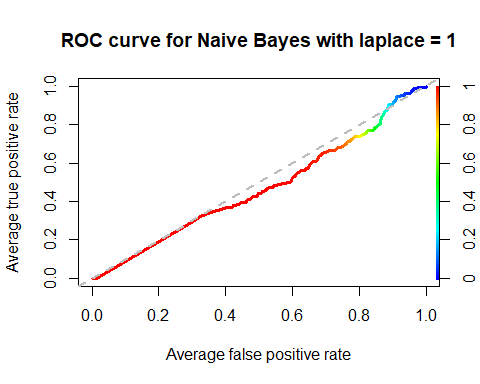
## Recall   
## 0.2162162

fscore$byClass["F1"]

## F1   
## 0.01284109

### STEP 5.2 : Evalutating Performance of the best model with ROC curve

library(ROCR)  
rocpredict <- predict(approval\_classifier1,test\_classifier,type = "raw")  
pred = prediction(rocpredict[,2],test\_label)  
pref = performance(pred,"tpr","fpr")  
plot(pref, avg="threshold", colorize=T, lwd=3, main="ROC curve for Naive Bayes with laplace = 1")  
abline(a=0,b=1,lwd=2,lty=2,col="grey")



#AUC  
a <- performance(pred,"auc")  
a <- unlist(slot(a,"y.values"))  
a

## [1] 0.4676902

# PART 2 : RANDOM FOREST MODELLING

## STEP 1 - STEP 3 remains the same

## STEP 4 : Training the Random Forest Model

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

library(caTools)  
memory.limit(size = 56000)

## [1] 56000

#Training Random Forest Model  
model\_random <- randomForest(train\_classifier,train\_label)  
model\_random

##   
## Call:  
## randomForest(x = train\_classifier, y = train\_label)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 4  
##   
## OOB estimate of error rate: 0.42%  
## Confusion matrix:  
## Not Approved Approved class.error  
## Not Approved 43033 647 0.0148122711  
## Approved 81 128619 0.0006293706

## STEP 5 : Evalutating the model

#Predicting values with the help of trained model for the training data itself  
random\_predict <- predict(model\_random,train\_classifier)  
table(random\_predict,train\_label)

## train\_label  
## random\_predict Not Approved Approved  
## Not Approved 43216 66  
## Approved 464 128634

accuracyr <- table(train\_label,random\_predict)  
print("The accuracy of the model on it's own training data is : ")

## [1] "The accuracy of the model on it's own training data is : "

sum(diag(accuracyr))/sum(accuracyr)

## [1] 0.9969254

#Predicting values with the help of trained model for the testing data  
random\_predict1 <- predict(model\_random,test\_classifier)  
accuracy\_test <- table(random\_predict1,test\_label)  
accuracy\_test

## test\_label  
## random\_predict1 Not Approved Approved  
## Not Approved 118 59  
## Approved 141 54474

print("The accuracy of the model on testing data is : ")

## [1] "The accuracy of the model on testing data is : "

sum(diag(accuracy\_test))/sum(accuracy\_test)

## [1] 0.9963498

confusionMatrix(random\_predict1,test\_label)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Not Approved Approved  
## Not Approved 118 59  
## Approved 141 54474  
##   
## Accuracy : 0.9963   
## 95% CI : (0.9958, 0.9968)  
## No Information Rate : 0.9953   
## P-Value [Acc > NIR] : 0.00007711246   
##   
## Kappa : 0.5395   
##   
## Mcnemar's Test P-Value : 0.00000001019   
##   
## Sensitivity : 0.455598   
## Specificity : 0.998918   
## Pos Pred Value : 0.666667   
## Neg Pred Value : 0.997418   
## Prevalence : 0.004727   
## Detection Rate : 0.002154   
## Detection Prevalence : 0.003230   
## Balanced Accuracy : 0.727258   
##   
## 'Positive' Class : Not Approved   
##

## STEP 6 : Improving the model performance

model\_random1 <- randomForest(train\_classifier,train\_label,ntree=500,mtry=6)  
model\_random1

##   
## Call:  
## randomForest(x = train\_classifier, y = train\_label, ntree = 500, mtry = 6)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 6  
##   
## OOB estimate of error rate: 0.31%  
## Confusion matrix:  
## Not Approved Approved class.error  
## Not Approved 43219 461 0.0105540293  
## Approved 79 128621 0.0006138306

random\_predict2 <- predict(model\_random1,test\_classifier)  
confusionMatrix(random\_predict2,test\_label)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Not Approved Approved  
## Not Approved 138 60  
## Approved 121 54473  
##   
## Accuracy : 0.9967   
## 95% CI : (0.9962, 0.9972)  
## No Information Rate : 0.9953   
## P-Value [Acc > NIR] : 0.0000001774   
##   
## Kappa : 0.6023   
##   
## Mcnemar's Test P-Value : 0.0000082050   
##   
## Sensitivity : 0.532819   
## Specificity : 0.998900   
## Pos Pred Value : 0.696970   
## Neg Pred Value : 0.997784   
## Prevalence : 0.004727   
## Detection Rate : 0.002519   
## Detection Prevalence : 0.003614   
## Balanced Accuracy : 0.765859   
##   
## 'Positive' Class : Not Approved   
##

model\_random2 <- randomForest(train\_classifier,train\_label,ntree=500,mtry=8,importance = T)  
model\_random2

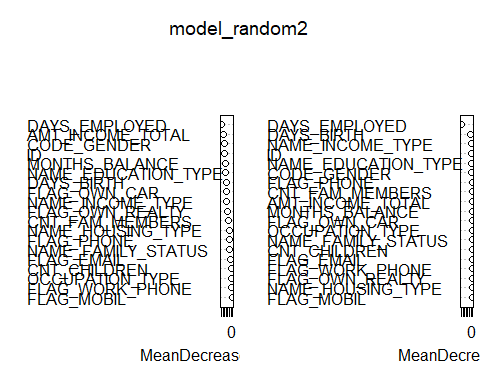
##   
## Call:  
## randomForest(x = train\_classifier, y = train\_label, ntree = 500, mtry = 8, importance = T)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 8  
##   
## OOB estimate of error rate: 0.27%  
## Confusion matrix:  
## Not Approved Approved class.error  
## Not Approved 43277 403 0.0092261905  
## Approved 58 128642 0.0004506605

random\_predict3 <- predict(model\_random2,test\_classifier)  
confusionMatrix(random\_predict3,test\_label)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Not Approved Approved  
## Not Approved 150 63  
## Approved 109 54470  
##   
## Accuracy : 0.9969   
## 95% CI : (0.9964, 0.9973)  
## No Information Rate : 0.9953   
## P-Value [Acc > NIR] : 0.000000005126   
##   
## Kappa : 0.634   
##   
## Mcnemar's Test P-Value : 0.0006009   
##   
## Sensitivity : 0.579151   
## Specificity : 0.998845   
## Pos Pred Value : 0.704225   
## Neg Pred Value : 0.998003   
## Prevalence : 0.004727   
## Detection Rate : 0.002738   
## Detection Prevalence : 0.003887   
## Balanced Accuracy : 0.788998   
##   
## 'Positive' Class : Not Approved   
##

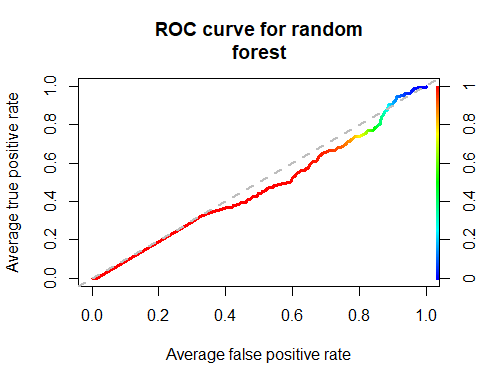
#Checking the importance of varibale in model training for the best model

varImpPlot(model\_random2)



## STEP 5.1 : Further evaluating the best model by ROC curve

rocrandom <- predict(model\_random2,test\_classifier,type = "prob")  
pred1 = prediction(rocrandom[,2],test\_label)  
pref1 = performance(pred,"tpr","fpr")  
plot(pref1, avg="threshold", colorize=T, lwd=3, main="ROC curve for random  
forest" )   
abline(a=0,b=1,lwd=2,lty=2,col="grey")



#AUC  
ar <- performance(pred1,"auc")  
ar <- unlist(slot(ar,"y.values"))  
ar

## [1] 0.03339563

### STEP 5.2 : Model statistics of the best model

# F score of the best random model   
frscore <- confusionMatrix(random\_predict3,test\_label,mode="prec\_recall")  
print("The precision of the naive bayes is: ")

## [1] "The precision of the naive bayes is: "

frscore$byClass["Precision"]

## Precision   
## 0.7042254

frscore$byClass["Recall"]

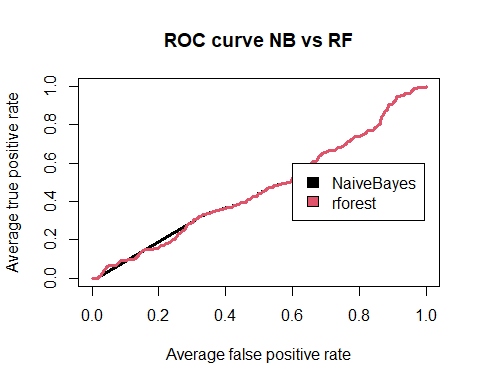
## Recall   
## 0.5791506

frscore$byClass["F1"]

## F1   
## 0.6355932

# STEP 6.2: COMPARING NAIVE BAYES AND RANDOM FOREST FOR THIS DATASET

plot(pref, col=1, lwd=3,avg= "threshold", main="ROC curve NB vs RF")  
plot(pref1, col=2, lwd=3, add=TRUE)  
legend(0.6, 0.6, c("NaiveBayes","rforest"), 1:2)



Appendix 2: Bank Default

# PART 1: LOGISTIC REGRESSION MODELLING

## STEP 1 : Collecting Data

setwd("W:/DMML1/DMML\_1\_Project/DataSets/D5\_russian\_banks")  
bank <- read.csv("russian\_banks.csv")

## STEP 2 : Exploring and Preparing the Data

print("The summary of bank data is:")

## [1] "The summary of bank data is:"

summary(bank)

## Ð.Ð.Ñ.ÐµÐ.Ð.Ð.Ñ. Ð.Ð.Ñ.Ð. net\_assets ROA   
## Min. : 0 Length:72439 Min. :0.000e+00 Min. :-64556.0   
## 1st Qu.:1222 Class :character 1st Qu.:1.280e+06 1st Qu.: 14.0   
## Median :2388 Mode :character Median :3.617e+06 Median : 91.0   
## Mean :2127 Mean :7.136e+07 Mean : 123.1   
## 3rd Qu.:3013 3rd Qu.:1.354e+07 3rd Qu.: 215.0   
## Max. :3481 Max. :2.378e+10 Max. : 76314.0   
## NA's :1956   
## liquid ibl stocks   
## Min. :0.000e+00 Min. :0.000e+00 Min. : -12236   
## 1st Qu.:1.865e+05 1st Qu.:1.500e+04 1st Qu.: 0   
## Median :4.905e+05 Median :1.770e+05 Median : 0   
## Mean :4.960e+06 Mean :7.609e+06 Mean : 659612   
## 3rd Qu.:1.456e+06 3rd Qu.:8.000e+05 3rd Qu.: 15604   
## Max. :1.815e+09 Max. :2.058e+09 Max. :255902149   
##   
## bond oth\_cap sunk\_retail\_credit   
## Min. :0.000e+00 Min. : 0 Min. : 0   
## 1st Qu.:0.000e+00 1st Qu.: 0 1st Qu.: 457   
## Median :6.047e+04 Median : 0 Median : 8633   
## Mean :7.998e+06 Mean : 1751494 Mean : 657227   
## 3rd Qu.:1.093e+06 3rd Qu.: 66 3rd Qu.: 58612   
## Max. :2.453e+09 Max. :982134216 Max. :177064674   
##   
## NI organization\_credit sunk\_organization\_credit  
## Min. :-351890144 Min. :0.000e+00 Min. : 0   
## 1st Qu.: 1041 1st Qu.:3.461e+05 1st Qu.: 575   
## Median : 11564 Median :1.178e+06 Median : 23944   
## Mean : 433166 Mean :3.097e+07 Mean : 1650831   
## 3rd Qu.: 59582 3rd Qu.:4.617e+06 3rd Qu.: 140174   
## Max. : 624187614 Max. :1.190e+10 Max. :454251327   
##   
## credit\_portf sunk\_credit\_portf organization\_deposit  
## Min. :0.000e+00 Min. : 0 Min. :0.000e+00   
## 1st Qu.:5.429e+05 1st Qu.: 5834 1st Qu.:3.185e+05   
## Median :1.705e+06 Median : 46804 Median :9.858e+05   
## Mean :4.110e+07 Mean : 2275845 Mean :2.181e+07   
## 3rd Qu.:6.515e+06 3rd Qu.: 229608 3rd Qu.:3.161e+06   
## Max. :1.603e+10 Max. :563771580 Max. :7.695e+09   
##   
## retail\_deposit security\_tot ROE retail\_credit   
## Min. :0.000e+00 Min. :0.000e+00 Min. :-4710789 Min. :0.000e+00   
## 1st Qu.:1.326e+05 1st Qu.:0.000e+00 1st Qu.: 90 1st Qu.:2.287e+05   
## Median :8.986e+05 Median :1.981e+05 Median : 478 Median :2.287e+05   
## Mean :2.042e+07 Mean :9.185e+06 Mean : 456 Mean :1.310e+06   
## 3rd Qu.:3.947e+06 3rd Qu.:1.597e+06 3rd Qu.: 1077 3rd Qu.:2.287e+05   
## Max. :1.146e+10 Max. :2.482e+09 Max. : 98776 Max. :1.673e+09   
##   
## reserv\_credit\_perc zalog\_credit\_perc foreign\_na\_fr retail\_deposit\_fr   
## Min. : 0 Min. : 0 Min. : 0 Min. :0.000e+00   
## 1st Qu.: 1094 1st Qu.:10409 1st Qu.: 2075 1st Qu.:4.822e+05   
## Median : 1094 Median :10409 Median : 2075 Median :4.822e+05   
## Mean : 1259 Mean :10545 Mean : 5315 Mean :5.851e+06   
## 3rd Qu.: 1094 3rd Qu.:10409 3rd Qu.: 2075 3rd Qu.:4.822e+05   
## Max. :10000 Max. :90250 Max. :99943 Max. :7.944e+09   
##   
## N3 N2 N1 capital   
## Min. : 0 Min. : 0 Min. : -2.76 Min. :-367608967   
## 1st Qu.: 73 1st Qu.: 9915 1st Qu.: 2058.00 1st Qu.: 310758   
## Median : 108 Median : 9915 Median : 2058.00 Median : 610627   
## Mean : 58153 Mean : 34739 Mean : 2346.69 Mean : 8746139   
## 3rd Qu.: 8219 3rd Qu.: 9915 3rd Qu.: 2058.00 3rd Qu.: 2013250   
## Max. :99999999 Max. :100000000 Max. :99969.00 Max. :3653136759   
##   
## msk\_spb INF\_SA NX\_growth micex\_std   
## Min. :0.0000 Min. :-0.020319 Min. :-6302.000 Min. :12.59   
## 1st Qu.:0.0000 1st Qu.: 0.005913 1st Qu.:-1598.000 1st Qu.:19.98   
## Median :1.0000 Median : 0.010956 Median : -33.000 Median :26.82   
## Mean :0.5771 Mean : 0.010984 Mean : -4.528 Mean :31.77   
## 3rd Qu.:1.0000 3rd Qu.: 0.015897 3rd Qu.: 1667.000 3rd Qu.:39.72   
## Max. :1.0000 Max. : 0.032372 Max. : 7322.000 Max. :87.37   
##   
## miacr\_std miacr\_amount usd\_rub\_std\_diff micex\_return   
## Min. :0.03742 Min. :1284605 Min. :-0.74823 Min. :-0.115092   
## 1st Qu.:0.16350 1st Qu.:3007318 1st Qu.:-0.37112 1st Qu.:-0.024347   
## Median :0.29328 Median :4016741 Median :-0.09847 Median : 0.010585   
## Mean :0.38108 Mean :4134695 Mean : 0.28134 Mean : 0.004759   
## 3rd Qu.:0.40715 3rd Qu.:4859773 3rd Qu.: 0.48449 3rd Qu.: 0.039273   
## Max. :5.62217 Max. :8009028 Max. : 5.33210 Max. : 0.107197   
##   
## net\_foreign\_assets\_diff net\_gov\_debt\_diff other\_fin\_debt\_diff  
## Min. :-4312523 Min. :-1634070 Min. :-179020   
## 1st Qu.: -296199 1st Qu.: -368635 1st Qu.: -82   
## Median : 124556 Median : -173220 Median : 30484   
## Mean : 172080 Mean : 5672 Mean : 72415   
## 3rd Qu.: 397184 3rd Qu.: 101793 3rd Qu.: 62168   
## Max. : 5351783 Max. : 2541323 Max. :1839484   
##   
## retail\_debt\_SA\_DETREND\_diff stocks\_capital\_diff i\_retail\_spread\_diff  
## Min. :-99458 Min. :-2072392 Min. :-25.94000   
## 1st Qu.:-24686 1st Qu.: -6898 1st Qu.: -0.58000   
## Median : -1595 Median : 41239 Median : -0.10000   
## Mean : 1574 Mean : 133170 Mean : -0.03651   
## 3rd Qu.: 27251 3rd Qu.: 97823 3rd Qu.: 0.30000   
## Max. :127125 Max. : 5460819 Max. : 16.75000   
##   
## usd\_rub\_return miacr\_diff default   
## Min. :-0.118357 Min. :-1.84835 Min. :0.000000   
## 1st Qu.:-0.015871 1st Qu.:-0.21346 1st Qu.:0.000000   
## Median :-0.001414 Median : 0.02118 Median :0.000000   
## Mean : 0.009324 Mean : 0.08081 Mean :0.005218   
## 3rd Qu.: 0.026706 3rd Qu.: 0.19327 3rd Qu.:0.000000   
## Max. : 0.206696 Max. : 5.25758 Max. :1.000000   
##

print("The structure of bank data is:")

## [1] "The structure of bank data is:"

str(bank)

## 'data.frame': 72439 obs. of 45 variables:  
## $ Ð.Ð.Ñ.ÐµÐ.Ð.Ð.Ñ. : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ Ð.Ð.Ñ.Ð. : chr "2010-02-01" "2010-03-01" "2010-04-01" "2010-05-01" ...  
## $ net\_assets : num 423017 498411 571220 523027 473713 ...  
## $ ROA : num 27 75 54 41 31 30 31 31 27 28 ...  
## $ liquid : num 112770 172628 211860 159970 131782 ...  
## $ ibl : num 60000 90000 90000 90000 135000 ...  
## $ stocks : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ bond : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ oth\_cap : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ sunk\_retail\_credit : num 0 0 997 481 485 235 235 235 235 235 ...  
## $ NI : num 102 569 651 677 644 ...  
## $ organization\_credit : num 225000 225000 227287 227130 191306 ...  
## $ sunk\_organization\_credit : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ credit\_portf : num 232150 232140 1704839 236631 200546 ...  
## $ sunk\_credit\_portf : num 0 0 997 481 485 235 235 235 235 235 ...  
## $ organization\_deposit : num 95828 156131 244427 200250 223835 ...  
## $ retail\_deposit : num 61007 83773 29306 37288 29394 ...  
## $ security\_tot : num 0 0 0 29977 0 ...  
## $ ROE : num 54 150 114 89 68 68 75 77 71 79 ...  
## $ retail\_credit : num 7150 7140 36868 9501 9240 ...  
## $ reserv\_credit\_perc : num 1094 1094 1094 1094 1094 ...  
## $ zalog\_credit\_perc : num 10409 10409 10409 10409 10409 ...  
## $ foreign\_na\_fr : num 2075 2075 2075 2075 2075 ...  
## $ retail\_deposit\_fr : num 482234 482234 482234 482234 482234 ...  
## $ N3 : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ N2 : num 9915 9915 9915 9915 9915 ...  
## $ N1 : num 2058 2058 2058 2058 2058 ...  
## $ capital : num 227962 228552 228565 228582 227896 ...  
## $ msk\_spb : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ INF\_SA : num 0.02 0.0275 0.025 0.0212 0.0324 ...  
## $ NX\_growth : num 2728 -980 -299 -767 -1736 ...  
## $ micex\_std : num 29.8 40.1 23.8 25.8 53.7 ...  
## $ miacr\_std : num 0.725 0.331 0.36 0.246 0.138 ...  
## $ miacr\_amount : num 2643995 2643995 2643995 2643995 2643995 ...  
## $ usd\_rub\_std\_diff : num 1.082 -0.527 0.345 -0.335 2.929 ...  
## $ micex\_return : num 0 -0.0571 0.0434 0.0484 -0.1055 ...  
## $ net\_foreign\_assets\_diff : num -156425 225429 -128438 -223667 251702 ...  
## $ net\_gov\_debt\_diff : num 1595305 -265831 430871 92737 -134963 ...  
## $ other\_fin\_debt\_diff : num 47521 -34392 -21876 -1075 81 ...  
## $ retail\_debt\_SA\_DETREND\_diff: num 33587 22186 -14984 -19524 -20743 ...  
## $ stocks\_capital\_diff : num 214757 59909 4293 64243 36008 ...  
## $ i\_retail\_spread\_diff : num 0 -0.1 0.1 -0.1 0.1 ...  
## $ usd\_rub\_return : num 0.00807 0.0107 -0.01985 -0.01239 0.04253 ...  
## $ miacr\_diff : num -0.83 -0.173 -0.297 -0.255 -0.533 ...  
## $ default : num 0 0 0 0 0 0 0 0 0 0 ...

### STEP 2.1 : Preparing the varibales

library(dplyr)

##   
## Attaching package: 'dplyr'

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

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

#Changing the names of the columns for better understanding  
library(lubridate)

##   
## Attaching package: 'lubridate'

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

names(bank)[1] <- "license"  
names(bank)[2] <- "date"  
#Changing the date structure and writing it into separate columns  
bank <- bank %>% mutate(date = ymd(date)) %>%   
 mutate\_at(vars(date), funs(year, month, day))

## Warning: `funs()` is deprecated as of dplyr 0.8.0.  
## Please use a list of either functions or lambdas:   
##   
## # Simple named list:   
## list(mean = mean, median = median)  
##   
## # Auto named with `tibble::lst()`:   
## tibble::lst(mean, median)  
##   
## # Using lambdas  
## list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_warnings()` to see where this warning was generated.

#Visualising missing values  
library(VIM)

## Loading required package: colorspace

## Loading required package: grid

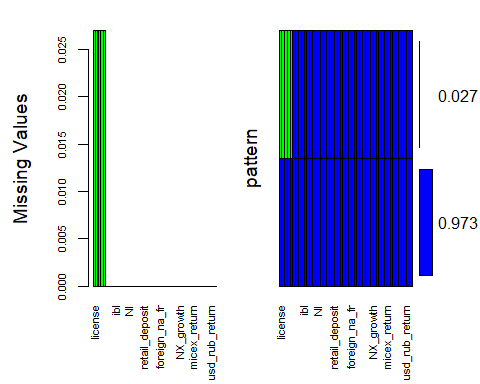
## VIM is ready to use.

## Suggestions and bug-reports can be submitted at: https://github.com/statistikat/VIM/issues

##   
## Attaching package: 'VIM'

## The following object is masked from 'package:datasets':  
##   
## sleep

mv <- aggr(bank,col=c("blue","green"),numbers=T,sortVars=T,labels=names(bank),cex.axis=.7,  
 gap=3,ylab=c("Missing Values","pattern"))



##   
## Variables sorted by number of missings:   
## Variable Count  
## license 0.02700203  
## date 0.02700203  
## year 0.02700203  
## month 0.02700203  
## day 0.02700203  
## net\_assets 0.00000000  
## ROA 0.00000000  
## liquid 0.00000000  
## ibl 0.00000000  
## stocks 0.00000000  
## bond 0.00000000  
## oth\_cap 0.00000000  
## sunk\_retail\_credit 0.00000000  
## NI 0.00000000  
## organization\_credit 0.00000000  
## sunk\_organization\_credit 0.00000000  
## credit\_portf 0.00000000  
## sunk\_credit\_portf 0.00000000  
## organization\_deposit 0.00000000  
## retail\_deposit 0.00000000  
## security\_tot 0.00000000  
## ROE 0.00000000  
## retail\_credit 0.00000000  
## reserv\_credit\_perc 0.00000000  
## zalog\_credit\_perc 0.00000000  
## foreign\_na\_fr 0.00000000  
## retail\_deposit\_fr 0.00000000  
## N3 0.00000000  
## N2 0.00000000  
## N1 0.00000000  
## capital 0.00000000  
## msk\_spb 0.00000000  
## INF\_SA 0.00000000  
## NX\_growth 0.00000000  
## micex\_std 0.00000000  
## miacr\_std 0.00000000  
## miacr\_amount 0.00000000  
## usd\_rub\_std\_diff 0.00000000  
## micex\_return 0.00000000  
## net\_foreign\_assets\_diff 0.00000000  
## net\_gov\_debt\_diff 0.00000000  
## other\_fin\_debt\_diff 0.00000000  
## retail\_debt\_SA\_DETREND\_diff 0.00000000  
## stocks\_capital\_diff 0.00000000  
## i\_retail\_spread\_diff 0.00000000  
## usd\_rub\_return 0.00000000  
## miacr\_diff 0.00000000  
## default 0.00000000

### STEP 2.2 : Dealing with NA values in the dataset

Since the NA values are only in the date and license column so they can be removed from the dataset

#Checking the number of NA values  
print(paste0("Total NA values in the bank data are: "))

## [1] "Total NA values in the bank data are: "

colSums(is.na(bank))

## license date   
## 1956 1956   
## net\_assets ROA   
## 0 0   
## liquid ibl   
## 0 0   
## stocks bond   
## 0 0   
## oth\_cap sunk\_retail\_credit   
## 0 0   
## NI organization\_credit   
## 0 0   
## sunk\_organization\_credit credit\_portf   
## 0 0   
## sunk\_credit\_portf organization\_deposit   
## 0 0   
## retail\_deposit security\_tot   
## 0 0   
## ROE retail\_credit   
## 0 0   
## reserv\_credit\_perc zalog\_credit\_perc   
## 0 0   
## foreign\_na\_fr retail\_deposit\_fr   
## 0 0   
## N3 N2   
## 0 0   
## N1 capital   
## 0 0   
## msk\_spb INF\_SA   
## 0 0   
## NX\_growth micex\_std   
## 0 0   
## miacr\_std miacr\_amount   
## 0 0   
## usd\_rub\_std\_diff micex\_return   
## 0 0   
## net\_foreign\_assets\_diff net\_gov\_debt\_diff   
## 0 0   
## other\_fin\_debt\_diff retail\_debt\_SA\_DETREND\_diff   
## 0 0   
## stocks\_capital\_diff i\_retail\_spread\_diff   
## 0 0   
## usd\_rub\_return miacr\_diff   
## 0 0   
## default year   
## 0 1956   
## month day   
## 1956 1956

#sum(is.na(bank))  
#Deleting the NA values as it won't affect the dataset  
bank <- na.omit(bank)

### STEP 2.3 : Finalizing preparing the variables

#Dropping the second column of date as it has already been separated.  
bank <- bank[,-2]  
#Setting Seed value to 999 so that results are constant  
set.seed(999)  
#Factorizing the predictor(dependent) variable  
bank$default <- as.factor(bank$default)  
#Finally checking the structure of the data  
str(bank)

## 'data.frame': 70483 obs. of 47 variables:  
## $ license : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ net\_assets : num 423017 498411 571220 523027 473713 ...  
## $ ROA : num 27 75 54 41 31 30 31 31 27 28 ...  
## $ liquid : num 112770 172628 211860 159970 131782 ...  
## $ ibl : num 60000 90000 90000 90000 135000 ...  
## $ stocks : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ bond : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ oth\_cap : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ sunk\_retail\_credit : num 0 0 997 481 485 235 235 235 235 235 ...  
## $ NI : num 102 569 651 677 644 ...  
## $ organization\_credit : num 225000 225000 227287 227130 191306 ...  
## $ sunk\_organization\_credit : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ credit\_portf : num 232150 232140 1704839 236631 200546 ...  
## $ sunk\_credit\_portf : num 0 0 997 481 485 235 235 235 235 235 ...  
## $ organization\_deposit : num 95828 156131 244427 200250 223835 ...  
## $ retail\_deposit : num 61007 83773 29306 37288 29394 ...  
## $ security\_tot : num 0 0 0 29977 0 ...  
## $ ROE : num 54 150 114 89 68 68 75 77 71 79 ...  
## $ retail\_credit : num 7150 7140 36868 9501 9240 ...  
## $ reserv\_credit\_perc : num 1094 1094 1094 1094 1094 ...  
## $ zalog\_credit\_perc : num 10409 10409 10409 10409 10409 ...  
## $ foreign\_na\_fr : num 2075 2075 2075 2075 2075 ...  
## $ retail\_deposit\_fr : num 482234 482234 482234 482234 482234 ...  
## $ N3 : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ N2 : num 9915 9915 9915 9915 9915 ...  
## $ N1 : num 2058 2058 2058 2058 2058 ...  
## $ capital : num 227962 228552 228565 228582 227896 ...  
## $ msk\_spb : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ INF\_SA : num 0.02 0.0275 0.025 0.0212 0.0324 ...  
## $ NX\_growth : num 2728 -980 -299 -767 -1736 ...  
## $ micex\_std : num 29.8 40.1 23.8 25.8 53.7 ...  
## $ miacr\_std : num 0.725 0.331 0.36 0.246 0.138 ...  
## $ miacr\_amount : num 2643995 2643995 2643995 2643995 2643995 ...  
## $ usd\_rub\_std\_diff : num 1.082 -0.527 0.345 -0.335 2.929 ...  
## $ micex\_return : num 0 -0.0571 0.0434 0.0484 -0.1055 ...  
## $ net\_foreign\_assets\_diff : num -156425 225429 -128438 -223667 251702 ...  
## $ net\_gov\_debt\_diff : num 1595305 -265831 430871 92737 -134963 ...  
## $ other\_fin\_debt\_diff : num 47521 -34392 -21876 -1075 81 ...  
## $ retail\_debt\_SA\_DETREND\_diff: num 33587 22186 -14984 -19524 -20743 ...  
## $ stocks\_capital\_diff : num 214757 59909 4293 64243 36008 ...  
## $ i\_retail\_spread\_diff : num 0 -0.1 0.1 -0.1 0.1 ...  
## $ usd\_rub\_return : num 0.00807 0.0107 -0.01985 -0.01239 0.04253 ...  
## $ miacr\_diff : num -0.83 -0.173 -0.297 -0.255 -0.533 ...  
## $ default : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ year : num 2010 2010 2010 2010 2010 2010 2010 2010 2010 2010 ...  
## $ month : num 2 3 4 5 6 7 8 9 10 11 ...  
## $ day : int 1 1 1 1 1 1 1 1 1 1 ...

### STEP 2.4 : Visualizing the dataset

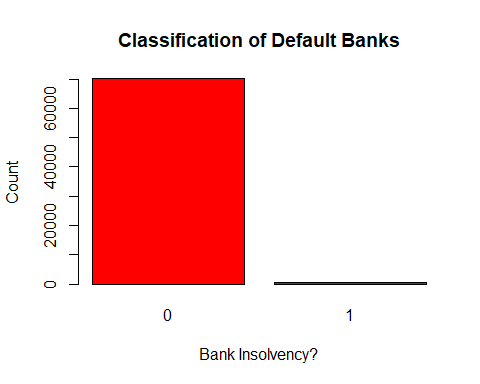
#Checking if the dataset in unbalanced  
print("Total number of insolvent banks in the dataset are : ")

## [1] "Total number of insolvent banks in the dataset are : "

as.data.frame(table(bank$default))

## Var1 Freq  
## 1 0 70110  
## 2 1 373

#Plotting the predictor variable  
plot(bank$default,main= "Classification of Default Banks",ylab="Count",xlab="Bank Insolvency?",col="red")



print("The number of default banks in the data set represented by 1 are: ")

## [1] "The number of default banks in the data set represented by 1 are: "

as.data.frame(table(bank$default))

## Var1 Freq  
## 1 0 70110  
## 2 1 373

## STEP 3: Data Preparation

### STEP 3.1: Creating data partition

#Stratified data partition  
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

index <- createDataPartition(y=bank$default,p=0.75,list = FALSE)  
training <- bank[index,]  
testing <- bank[-index,]  
print("The dimensions of the training and testing datasets are:")

## [1] "The dimensions of the training and testing datasets are:"

dim(training)

## [1] 52863 47

dim(testing)

## [1] 17620 47

print("Percentage division of default and non default status of banks in the training and testing datasets are:")

## [1] "Percentage division of default and non default status of banks in the training and testing datasets are:"

prop.table(table(training$default))\*100

##   
## 0 1   
## 99.470329 0.529671

prop.table(table(testing$default))\*100

##   
## 0 1   
## 99.4721907 0.5278093

print("Number of default and non default status of banks in the training and testing datasets are:")

## [1] "Number of default and non default status of banks in the training and testing datasets are:"

as.data.frame(table(training$default))

## Var1 Freq  
## 1 0 52583  
## 2 1 280

as.data.frame(table(testing$default))

## Var1 Freq  
## 1 0 17527  
## 2 1 93

## STEP 3.2: Dealing with unbalanced data in the training data

#Using SMOTE to balance the training dataset  
set.seed(1040)  
library(DMwR)

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

##   
## Attaching package: 'DMwR'

## The following object is masked from 'package:VIM':  
##   
## kNN

balancedbank <- SMOTE(default~.,training,perc.over = 5500,k=2,perc.under = 200)  
print("Training data after balancing:")

## [1] "Training data after balancing:"

as.data.frame(table(balancedbank$default))

## Var1 Freq  
## 1 0 30800  
## 2 1 15680

## STEP 4: Model Training ( Logistic Regression)

#Logistic Regression model with all the columns in the training dataset  
model <- glm(default~.,data = balancedbank,family = binomial)

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

summary(model)

##   
## Call:  
## glm(formula = default ~ ., family = binomial, data = balancedbank)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -8.49 0.00 0.00 0.00 8.49   
##   
## Coefficients: (1 not defined because of singularities)  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -3.360e+16 3.316e+08 -101312474 <2e-16 \*\*\*  
## license -1.152e+11 3.227e+02 -357079018 <2e-16 \*\*\*  
## net\_assets -6.570e+06 2.125e-02 -309129044 <2e-16 \*\*\*  
## ROA -6.231e+10 2.053e+02 -303501227 <2e-16 \*\*\*  
## liquid -6.484e+06 3.411e-02 -190076766 <2e-16 \*\*\*  
## ibl 4.116e+06 2.482e-02 165817007 <2e-16 \*\*\*  
## stocks 7.173e+07 1.046e-01 686045971 <2e-16 \*\*\*  
## bond 9.568e+06 5.185e-02 184542165 <2e-16 \*\*\*  
## oth\_cap 1.105e+07 4.594e-02 240427194 <2e-16 \*\*\*  
## sunk\_retail\_credit 1.221e+07 2.156e-01 56617327 <2e-16 \*\*\*  
## NI 1.013e+07 7.626e-02 132811942 <2e-16 \*\*\*  
## organization\_credit 1.294e+07 1.761e-02 734659018 <2e-16 \*\*\*  
## sunk\_organization\_credit -1.048e+07 1.585e-01 -66096571 <2e-16 \*\*\*  
## credit\_portf -2.468e+06 2.241e-02 -110158426 <2e-16 \*\*\*  
## sunk\_credit\_portf 1.095e+07 1.576e-01 69476828 <2e-16 \*\*\*  
## organization\_deposit 2.277e+06 1.079e-02 211025764 <2e-16 \*\*\*  
## retail\_deposit 8.134e+06 6.277e-03 1295743836 <2e-16 \*\*\*  
## security\_tot -2.672e+07 6.017e-02 -444032854 <2e-16 \*\*\*  
## ROE -1.744e+10 3.727e+01 -467903875 <2e-16 \*\*\*  
## retail\_credit -1.063e+06 1.833e-02 -58001960 <2e-16 \*\*\*  
## reserv\_credit\_perc -1.223e+11 3.405e+02 -359177630 <2e-16 \*\*\*  
## zalog\_credit\_perc -1.718e+09 6.848e+01 -25084660 <2e-16 \*\*\*  
## foreign\_na\_fr -5.486e+08 2.985e+01 -18378867 <2e-16 \*\*\*  
## retail\_deposit\_fr -8.060e+04 6.689e-03 -12050198 <2e-16 \*\*\*  
## N3 2.328e+07 9.647e-02 241366741 <2e-16 \*\*\*  
## N2 -1.897e+07 3.359e-01 -56486323 <2e-16 \*\*\*  
## N1 -1.755e+10 2.058e+02 -85282205 <2e-16 \*\*\*  
## capital -1.075e+07 3.580e-02 -300260563 <2e-16 \*\*\*  
## msk\_spb 2.191e+14 7.263e+05 301610993 <2e-16 \*\*\*  
## INF\_SA -7.373e+15 4.994e+07 -147619681 <2e-16 \*\*\*  
## NX\_growth 3.615e+10 1.470e+02 245869730 <2e-16 \*\*\*  
## micex\_std -1.498e+12 2.620e+04 -57183524 <2e-16 \*\*\*  
## miacr\_std -2.455e+13 9.657e+05 -25422508 <2e-16 \*\*\*  
## miacr\_amount 1.607e+07 3.103e-01 51774278 <2e-16 \*\*\*  
## usd\_rub\_std\_diff -1.185e+14 3.666e+05 -323238224 <2e-16 \*\*\*  
## micex\_return 1.125e+15 9.761e+06 115290359 <2e-16 \*\*\*  
## net\_foreign\_assets\_diff -1.996e+08 4.742e-01 -420928162 <2e-16 \*\*\*  
## net\_gov\_debt\_diff 5.311e+07 7.051e-01 75327100 <2e-16 \*\*\*  
## other\_fin\_debt\_diff 2.791e+08 1.705e+00 163728560 <2e-16 \*\*\*  
## retail\_debt\_SA\_DETREND\_diff 1.711e+09 8.806e+00 194327736 <2e-16 \*\*\*  
## stocks\_capital\_diff -3.447e+07 8.086e-01 -42627498 <2e-16 \*\*\*  
## i\_retail\_spread\_diff 3.628e+12 1.187e+05 30561685 <2e-16 \*\*\*  
## usd\_rub\_return 1.389e+15 1.191e+07 116664905 <2e-16 \*\*\*  
## miacr\_diff 8.790e+13 7.584e+05 115896593 <2e-16 \*\*\*  
## year 1.665e+13 1.647e+05 101102417 <2e-16 \*\*\*  
## month 4.503e+12 1.014e+05 44428289 <2e-16 \*\*\*  
## day NA NA NA NA   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 59426 on 46479 degrees of freedom  
## Residual deviance: 498412 on 46434 degrees of freedom  
## AIC: 498504  
##   
## Number of Fisher Scoring iterations: 25

## STEP 5: Evaluating Model Performance

#Pseduo R2  
library(pscl)

## Classes and Methods for R developed in the  
## Political Science Computational Laboratory  
## Department of Political Science  
## Stanford University  
## Simon Jackman  
## hurdle and zeroinfl functions by Achim Zeileis

pR2(model)

## fitting null model for pseudo-r2

## llh llhNull G2 McFadden r2ML   
## -249205.81953 -29712.88263 -438985.87381 -7.38713 -12638.96157   
## r2CU   
## -17516.35989

library(caret)  
varImp(model)

## Overall  
## license 357079018  
## net\_assets 309129044  
## ROA 303501227  
## liquid 190076766  
## ibl 165817007  
## stocks 686045971  
## bond 184542165  
## oth\_cap 240427194  
## sunk\_retail\_credit 56617327  
## NI 132811942  
## organization\_credit 734659018  
## sunk\_organization\_credit 66096571  
## credit\_portf 110158426  
## sunk\_credit\_portf 69476828  
## organization\_deposit 211025764  
## retail\_deposit 1295743836  
## security\_tot 444032854  
## ROE 467903875  
## retail\_credit 58001960  
## reserv\_credit\_perc 359177630  
## zalog\_credit\_perc 25084660  
## foreign\_na\_fr 18378867  
## retail\_deposit\_fr 12050198  
## N3 241366741  
## N2 56486323  
## N1 85282205  
## capital 300260563  
## msk\_spb 301610993  
## INF\_SA 147619681  
## NX\_growth 245869730  
## micex\_std 57183524  
## miacr\_std 25422508  
## miacr\_amount 51774278  
## usd\_rub\_std\_diff 323238224  
## micex\_return 115290359  
## net\_foreign\_assets\_diff 420928162  
## net\_gov\_debt\_diff 75327100  
## other\_fin\_debt\_diff 163728560  
## retail\_debt\_SA\_DETREND\_diff 194327736  
## stocks\_capital\_diff 42627498  
## i\_retail\_spread\_diff 30561685  
## usd\_rub\_return 116664905  
## miacr\_diff 115896593  
## year 101102417  
## month 44428289

Annova test to check model variable significance

#Anova test  
anova(model, test="Chisq")

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Analysis of Deviance Table  
##   
## Model: binomial, link: logit  
##   
## Response: default  
##   
## Terms added sequentially (first to last)  
##   
##   
## Df Deviance Resid. Df Resid. Dev Pr(>Chi)   
## NULL 46479 59426   
## license 1 8542 46478 50884 <2e-16 \*\*\*  
## net\_assets 1 217 46477 50667 <2e-16 \*\*\*  
## ROA 1 1842 46476 48825 <2e-16 \*\*\*  
## liquid 1 610 46475 48215 <2e-16 \*\*\*  
## ibl 1 312 46474 47902 <2e-16 \*\*\*  
## stocks 1 1622 46473 46280 <2e-16 \*\*\*  
## bond 1 1398 46472 44883 <2e-16 \*\*\*  
## oth\_cap 1 0 46471 1179493 1.000   
## sunk\_retail\_credit 1 89677 46470 1089816 <2e-16 \*\*\*  
## NI 1 75980 46469 1013836 <2e-16 \*\*\*  
## organization\_credit 1 0 46468 1200254 1.000   
## sunk\_organization\_credit 1 1157057 46467 43197 <2e-16 \*\*\*  
## credit\_portf 1 0 46466 1181583 1.000   
## sunk\_credit\_portf 1 0 46465 1192612 1.000   
## organization\_deposit 1 188436 46464 1004176 <2e-16 \*\*\*  
## retail\_deposit 1 94362 46463 909814 <2e-16 \*\*\*  
## security\_tot 1 0 46462 977648 1.000   
## ROE 1 0 46461 1098250 1.000   
## retail\_credit 1 162917 46460 935333 <2e-16 \*\*\*  
## reserv\_credit\_perc 1 0 46459 1022342 1.000   
## zalog\_credit\_perc 1 110005 46458 912337 <2e-16 \*\*\*  
## foreign\_na\_fr 1 0 46457 928412 1.000   
## retail\_deposit\_fr 1 0 46456 1061053 1.000   
## N3 1 6704 46455 1054349 <2e-16 \*\*\*  
## N2 1 0 46454 1058602 1.000   
## N1 1 132929 46453 925673 <2e-16 \*\*\*  
## capital 1 0 46452 1012682 1.000   
## msk\_spb 1 492068 46451 520615 <2e-16 \*\*\*  
## INF\_SA 1 68915 46450 451699 <2e-16 \*\*\*  
## NX\_growth 1 0 46449 522489 1.000   
## micex\_std 1 0 46448 1181511 1.000   
## miacr\_std 1 405852 46447 775659 <2e-16 \*\*\*  
## miacr\_amount 1 0 46446 1234783 1.000   
## usd\_rub\_std\_diff 1 1204205 46445 30578 <2e-16 \*\*\*  
## micex\_return 1 1 46444 30577 0.311   
## net\_foreign\_assets\_diff 1 0 46443 446869 1.000   
## net\_gov\_debt\_diff 1 0 46442 943046 1.000   
## other\_fin\_debt\_diff 1 216190 46441 726856 <2e-16 \*\*\*  
## retail\_debt\_SA\_DETREND\_diff 1 0 46440 859641 1.000   
## stocks\_capital\_diff 1 316607 46439 543034 <2e-16 \*\*\*  
## i\_retail\_spread\_diff 1 99048 46438 443986 <2e-16 \*\*\*  
## usd\_rub\_return 1 11966 46437 432019 <2e-16 \*\*\*  
## miacr\_diff 1 0 46436 830734 1.000   
## year 1 316680 46435 514055 <2e-16 \*\*\*  
## month 1 15643 46434 498412 <2e-16 \*\*\*  
## day 0 0 46434 498412   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

### STEP 5.2: Applying the trained model on the testing data

#Applying the model on the testing dataset  
predict1 <- predict(model,newdata=testing,type="response")

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

bank.predict <- ifelse(predict1 > 0.5 ,1,0)

#Misclassification error  
mError <- mean(bank.predict != testing$default)  
mError

## [1] 0.0514756

### STEP 5.3: Checking model accuracy

#CHECKING ACCURACY  
  
print("The accuracy of the model is:")

## [1] "The accuracy of the model is:"

accuracy <- 1-mError  
#  
accuracy

## [1] 0.9485244

table(testing$default)

##   
## 0 1   
## 17527 93

table(testing$default, predict1 > 0.5)

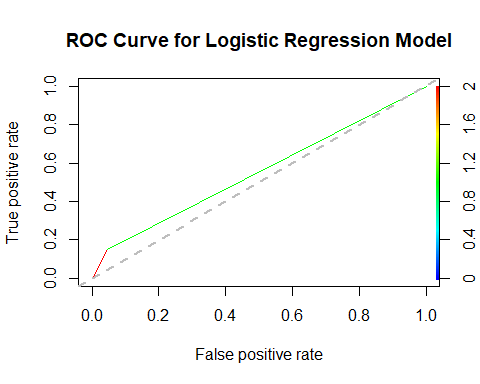
##   
## FALSE TRUE  
## 0 16699 828  
## 1 79 14

accuracyclass <- table(testing$default,bank.predict)  
sum(diag(accuracyclass))/sum(accuracyclass)

## [1] 0.9485244

### STEP 5.3: ROC curve of the model

library(ROCR)  
ROCRpred <- prediction(predict1, testing$default)  
ROCRperf <- performance(ROCRpred, 'tpr','fpr')  
plot(ROCRperf, colorize = TRUE, text.adj = c(-0.2,1.7),main="ROC Curve for Logistic Regression Model")  
abline(a=0,b=1,lwd=2,lty=2,col="grey")



aucvalue <- performance(ROCRpred, measure = "auc")  
aucvalue <- aucvalue@y.values[[1]]  
aucvalue

## [1] 0.5516481

## STEP 6: Improving Model Performance

print("Building a model by selecting appropriate variables accessed by looking at Chi square results in the Annova table.")

## [1] "Building a model by selecting appropriate variables accessed by looking at Chi square results in the Annova table."

model1 <- glm(default~sunk\_retail\_credit+NI+sunk\_organization\_credit  
 +organization\_deposit+retail\_deposit+retail\_credit+  
 zalog\_credit\_perc+N1+msk\_spb+INF\_SA+miacr\_std+usd\_rub\_std\_diff  
 +other\_fin\_debt\_diff+i\_retail\_spread\_diff+usd\_rub\_return+  
 stocks\_capital\_diff+year,data = balancedbank,  
 family = binomial(link = "logit"))

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

summary(model1)

##   
## Call:  
## glm(formula = default ~ sunk\_retail\_credit + NI + sunk\_organization\_credit +   
## organization\_deposit + retail\_deposit + retail\_credit + zalog\_credit\_perc +   
## N1 + msk\_spb + INF\_SA + miacr\_std + usd\_rub\_std\_diff + other\_fin\_debt\_diff +   
## i\_retail\_spread\_diff + usd\_rub\_return + stocks\_capital\_diff +   
## year, family = binomial(link = "logit"), data = balancedbank)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -5.0125 -0.8690 -0.5171 0.9697 3.1637   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.276e+02 1.175e+01 -19.378 < 2e-16 \*\*\*  
## sunk\_retail\_credit -1.482e-07 9.088e-09 -16.303 < 2e-16 \*\*\*  
## NI -2.847e-08 4.133e-09 -6.888 5.64e-12 \*\*\*  
## sunk\_organization\_credit -3.445e-09 6.359e-09 -0.542 0.5880   
## organization\_deposit -2.188e-08 1.229e-09 -17.793 < 2e-16 \*\*\*  
## retail\_deposit 1.194e-08 5.855e-10 20.389 < 2e-16 \*\*\*  
## retail\_credit -4.706e-07 4.630e-08 -10.165 < 2e-16 \*\*\*  
## zalog\_credit\_perc -5.336e-05 2.772e-06 -19.252 < 2e-16 \*\*\*  
## N1 -2.462e-04 1.050e-05 -23.450 < 2e-16 \*\*\*  
## msk\_spb 1.327e+00 2.705e-02 49.058 < 2e-16 \*\*\*  
## INF\_SA -4.963e+01 1.628e+00 -30.490 < 2e-16 \*\*\*  
## miacr\_std -2.908e-01 2.913e-02 -9.985 < 2e-16 \*\*\*  
## usd\_rub\_std\_diff -2.896e-01 1.520e-02 -19.051 < 2e-16 \*\*\*  
## other\_fin\_debt\_diff 8.176e-07 5.356e-08 15.266 < 2e-16 \*\*\*  
## i\_retail\_spread\_diff 5.301e-02 4.797e-03 11.053 < 2e-16 \*\*\*  
## usd\_rub\_return -1.377e+00 3.234e-01 -4.258 2.07e-05 \*\*\*  
## stocks\_capital\_diff 3.976e-08 2.066e-08 1.924 0.0543 .   
## year 1.131e-01 5.832e-03 19.398 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 59426 on 46479 degrees of freedom  
## Residual deviance: 49845 on 46462 degrees of freedom  
## AIC: 49881  
##   
## Number of Fisher Scoring iterations: 10

predict2 <- predict(model1,newdata=testing,type="response")  
  
bank.predict1 <- ifelse(predict2 > 0.5 ,1,0)

print("Misclassification of the improved model is:")

## [1] "Misclassification of the improved model is:"

mError1 <- mean(bank.predict1 != testing$default)  
mError1

## [1] 0.1240068

print("The accuracy of the improved model is:")

## [1] "The accuracy of the improved model is:"

accuracy1 <- 1-mError1  
accuracy1

## [1] 0.8759932

table(testing$default)

##   
## 0 1   
## 17527 93

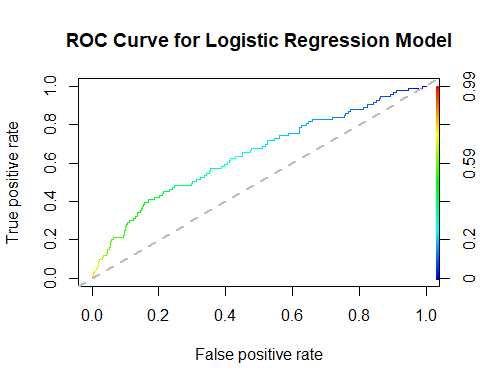
table(testing$default, predict2 > 0.5)

##   
## FALSE TRUE  
## 0 15407 2120  
## 1 65 28

accuracyclass1 <- table(testing$default,bank.predict1)  
sum(diag(accuracyclass1))/sum(accuracyclass1)

## [1] 0.8759932

library(ROCR)  
ROCRpred1 <- prediction(predict2, testing$default)  
ROCRperf1 <- performance(ROCRpred1, 'tpr','fpr')  
plot(ROCRperf1, colorize = TRUE, text.adj = c(-0.2,1.7),main="ROC Curve for Logistic Regression Model")  
abline(a=0,b=1,lwd=2,lty=2,col="grey")

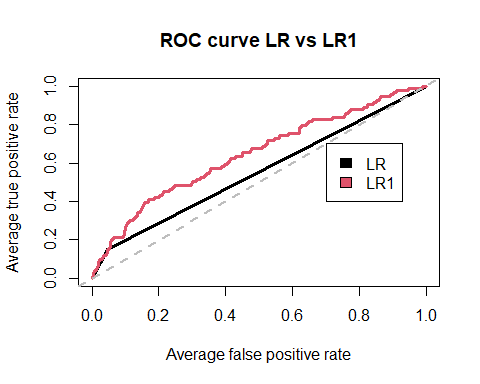


aucvalue1 <- performance(ROCRpred1, measure = "auc")  
aucvalue1 <- aucvalue1@y.values[[1]]  
aucvalue1

## [1] 0.6448981

### Step 6.2: Comparing Both the models

plot(ROCRperf, col=1, lwd=3,avg= "threshold", main="ROC curve LR vs LR1")  
plot(ROCRperf1, col=2, lwd=3, add=TRUE)  
legend(0.7, 0.7, c("LR","LR1"), 1:2)  
abline(a=0,b=1,lwd=2,lty=2,col="grey")



Appendix 3: Credit\_card\_fraud

# Part 1 : Decision Tree Modelling

## STEP 1 : Collecting Data

setwd("W:/DMML1/DMML\_1\_Project/DataSets/D4\_credit\_fraud")  
credit\_card <- read.csv("creditcard.csv")  
summary(credit\_card)

## Time V1 V2 V3   
## Min. : 0 Min. :-56.40751 Min. :-72.71573 Min. :-48.3256   
## 1st Qu.: 54202 1st Qu.: -0.92037 1st Qu.: -0.59855 1st Qu.: -0.8904   
## Median : 84692 Median : 0.01811 Median : 0.06549 Median : 0.1799   
## Mean : 94814 Mean : 0.00000 Mean : 0.00000 Mean : 0.0000   
## 3rd Qu.:139321 3rd Qu.: 1.31564 3rd Qu.: 0.80372 3rd Qu.: 1.0272   
## Max. :172792 Max. : 2.45493 Max. : 22.05773 Max. : 9.3826   
## V4 V5 V6 V7   
## Min. :-5.68317 Min. :-113.74331 Min. :-26.1605 Min. :-43.5572   
## 1st Qu.:-0.84864 1st Qu.: -0.69160 1st Qu.: -0.7683 1st Qu.: -0.5541   
## Median :-0.01985 Median : -0.05434 Median : -0.2742 Median : 0.0401   
## Mean : 0.00000 Mean : 0.00000 Mean : 0.0000 Mean : 0.0000   
## 3rd Qu.: 0.74334 3rd Qu.: 0.61193 3rd Qu.: 0.3986 3rd Qu.: 0.5704   
## Max. :16.87534 Max. : 34.80167 Max. : 73.3016 Max. :120.5895   
## V8 V9 V10 V11   
## Min. :-73.21672 Min. :-13.43407 Min. :-24.58826 Min. :-4.79747   
## 1st Qu.: -0.20863 1st Qu.: -0.64310 1st Qu.: -0.53543 1st Qu.:-0.76249   
## Median : 0.02236 Median : -0.05143 Median : -0.09292 Median :-0.03276   
## Mean : 0.00000 Mean : 0.00000 Mean : 0.00000 Mean : 0.00000   
## 3rd Qu.: 0.32735 3rd Qu.: 0.59714 3rd Qu.: 0.45392 3rd Qu.: 0.73959   
## Max. : 20.00721 Max. : 15.59500 Max. : 23.74514 Max. :12.01891   
## V12 V13 V14 V15   
## Min. :-18.6837 Min. :-5.79188 Min. :-19.2143 Min. :-4.49894   
## 1st Qu.: -0.4056 1st Qu.:-0.64854 1st Qu.: -0.4256 1st Qu.:-0.58288   
## Median : 0.1400 Median :-0.01357 Median : 0.0506 Median : 0.04807   
## Mean : 0.0000 Mean : 0.00000 Mean : 0.0000 Mean : 0.00000   
## 3rd Qu.: 0.6182 3rd Qu.: 0.66251 3rd Qu.: 0.4931 3rd Qu.: 0.64882   
## Max. : 7.8484 Max. : 7.12688 Max. : 10.5268 Max. : 8.87774   
## V16 V17 V18   
## Min. :-14.12985 Min. :-25.16280 Min. :-9.498746   
## 1st Qu.: -0.46804 1st Qu.: -0.48375 1st Qu.:-0.498850   
## Median : 0.06641 Median : -0.06568 Median :-0.003636   
## Mean : 0.00000 Mean : 0.00000 Mean : 0.000000   
## 3rd Qu.: 0.52330 3rd Qu.: 0.39968 3rd Qu.: 0.500807   
## Max. : 17.31511 Max. : 9.25353 Max. : 5.041069   
## V19 V20 V21   
## Min. :-7.213527 Min. :-54.49772 Min. :-34.83038   
## 1st Qu.:-0.456299 1st Qu.: -0.21172 1st Qu.: -0.22839   
## Median : 0.003735 Median : -0.06248 Median : -0.02945   
## Mean : 0.000000 Mean : 0.00000 Mean : 0.00000   
## 3rd Qu.: 0.458949 3rd Qu.: 0.13304 3rd Qu.: 0.18638   
## Max. : 5.591971 Max. : 39.42090 Max. : 27.20284   
## V22 V23 V24   
## Min. :-10.933144 Min. :-44.80774 Min. :-2.83663   
## 1st Qu.: -0.542350 1st Qu.: -0.16185 1st Qu.:-0.35459   
## Median : 0.006782 Median : -0.01119 Median : 0.04098   
## Mean : 0.000000 Mean : 0.00000 Mean : 0.00000   
## 3rd Qu.: 0.528554 3rd Qu.: 0.14764 3rd Qu.: 0.43953   
## Max. : 10.503090 Max. : 22.52841 Max. : 4.58455   
## V25 V26 V27   
## Min. :-10.29540 Min. :-2.60455 Min. :-22.565679   
## 1st Qu.: -0.31715 1st Qu.:-0.32698 1st Qu.: -0.070840   
## Median : 0.01659 Median :-0.05214 Median : 0.001342   
## Mean : 0.00000 Mean : 0.00000 Mean : 0.000000   
## 3rd Qu.: 0.35072 3rd Qu.: 0.24095 3rd Qu.: 0.091045   
## Max. : 7.51959 Max. : 3.51735 Max. : 31.612198   
## V28 Amount Class   
## Min. :-15.43008 Min. : 0.00 Min. :0.000000   
## 1st Qu.: -0.05296 1st Qu.: 5.60 1st Qu.:0.000000   
## Median : 0.01124 Median : 22.00 Median :0.000000   
## Mean : 0.00000 Mean : 88.35 Mean :0.001728   
## 3rd Qu.: 0.07828 3rd Qu.: 77.17 3rd Qu.:0.000000   
## Max. : 33.84781 Max. :25691.16 Max. :1.000000

## STEP 2 : Exploring and Preparing the Data

### STEP 2.1 : Checking the structure and NA values

str(credit\_card)

## 'data.frame': 284807 obs. of 31 variables:  
## $ Time : num 0 0 1 1 2 2 4 7 7 9 ...  
## $ V1 : num -1.36 1.192 -1.358 -0.966 -1.158 ...  
## $ V2 : num -0.0728 0.2662 -1.3402 -0.1852 0.8777 ...  
## $ V3 : num 2.536 0.166 1.773 1.793 1.549 ...  
## $ V4 : num 1.378 0.448 0.38 -0.863 0.403 ...  
## $ V5 : num -0.3383 0.06 -0.5032 -0.0103 -0.4072 ...  
## $ V6 : num 0.4624 -0.0824 1.8005 1.2472 0.0959 ...  
## $ V7 : num 0.2396 -0.0788 0.7915 0.2376 0.5929 ...  
## $ V8 : num 0.0987 0.0851 0.2477 0.3774 -0.2705 ...  
## $ V9 : num 0.364 -0.255 -1.515 -1.387 0.818 ...  
## $ V10 : num 0.0908 -0.167 0.2076 -0.055 0.7531 ...  
## $ V11 : num -0.552 1.613 0.625 -0.226 -0.823 ...  
## $ V12 : num -0.6178 1.0652 0.0661 0.1782 0.5382 ...  
## $ V13 : num -0.991 0.489 0.717 0.508 1.346 ...  
## $ V14 : num -0.311 -0.144 -0.166 -0.288 -1.12 ...  
## $ V15 : num 1.468 0.636 2.346 -0.631 0.175 ...  
## $ V16 : num -0.47 0.464 -2.89 -1.06 -0.451 ...  
## $ V17 : num 0.208 -0.115 1.11 -0.684 -0.237 ...  
## $ V18 : num 0.0258 -0.1834 -0.1214 1.9658 -0.0382 ...  
## $ V19 : num 0.404 -0.146 -2.262 -1.233 0.803 ...  
## $ V20 : num 0.2514 -0.0691 0.525 -0.208 0.4085 ...  
## $ V21 : num -0.01831 -0.22578 0.248 -0.1083 -0.00943 ...  
## $ V22 : num 0.27784 -0.63867 0.77168 0.00527 0.79828 ...  
## $ V23 : num -0.11 0.101 0.909 -0.19 -0.137 ...  
## $ V24 : num 0.0669 -0.3398 -0.6893 -1.1756 0.1413 ...  
## $ V25 : num 0.129 0.167 -0.328 0.647 -0.206 ...  
## $ V26 : num -0.189 0.126 -0.139 -0.222 0.502 ...  
## $ V27 : num 0.13356 -0.00898 -0.05535 0.06272 0.21942 ...  
## $ V28 : num -0.0211 0.0147 -0.0598 0.0615 0.2152 ...  
## $ Amount: num 149.62 2.69 378.66 123.5 69.99 ...  
## $ Class : int 0 0 0 0 0 0 0 0 0 0 ...

print(paste0("Total NA values in the dataset: ", sum(is.na(credit\_card))))

## [1] "Total NA values in the dataset: 0"

### STEP 2.2 : Cleaning the Data

#Discretizing the feature variable i.e. class  
credit\_card$Class <- as.factor(credit\_card$Class)  
str(credit\_card$Class)

## Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...

#Seeing how many fraud transactions are there in the dataset  
print(paste0("Distribution of Normal Tx (0) and Fraud Tx (1)"))

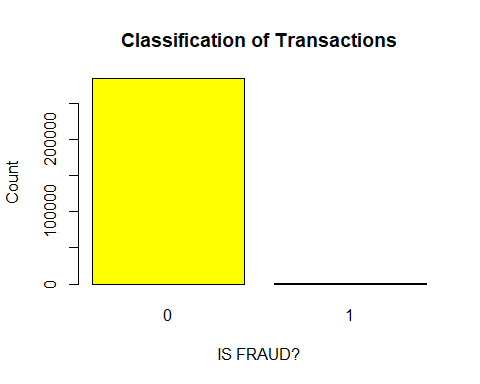
## [1] "Distribution of Normal Tx (0) and Fraud Tx (1)"

table(credit\_card$Class)

##   
## 0 1   
## 284315 492

### STEP 2.3 : Checking the Class ( predictor ) variable

#Plotting the classification of transactions to check the dataset  
plot(credit\_card$Class,main= "Classification of Transactions",ylab="Count",xlab="IS FRAUD?",col="yellow")



### STEP 2.4 : Data Preparation (Creating Random Training and Test datasets)

library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

index <- createDataPartition(y=credit\_card$Class,p=0.75,list = FALSE)  
training <- credit\_card[index,]  
testing <- credit\_card[-index,]  
dim(training)

## [1] 213606 31

dim(testing)

## [1] 71201 31

prop.table(table(training$Class))

##   
## 0 1   
## 0.99827252 0.00172748

prop.table(table(testing$Class))

##   
## 0 1   
## 0.998272496 0.001727504

print("Distribution of normal and fraud transaction in training and testing dataset is:")

## [1] "Distribution of normal and fraud transaction in training and testing dataset is:"

as.data.frame(table(training$Class))

## Var1 Freq  
## 1 0 213237  
## 2 1 369

as.data.frame(table(testing$Class))

## Var1 Freq  
## 1 0 71078  
## 2 1 123

### STEP 2.5 : Removing Unbalancing from the Dataset

#Setting the seed so that further evalutaions are constant  
set.seed(4321)  
options(scipen=999)  
library(DMwR)

## Loading required package: grid

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

balancedcc <- SMOTE(Class~.,data=training,perc.over = 7500,k=2,perc.under = 300)  
print("Distribution of normal and fraud transactions in the training dataset after oversampling is :")

## [1] "Distribution of normal and fraud transactions in the training dataset after oversampling is :"

as.data.frame(table(balancedcc$Class))

## Var1 Freq  
## 1 0 83025  
## 2 1 28044

## STEP 3 : Training a Model on the data

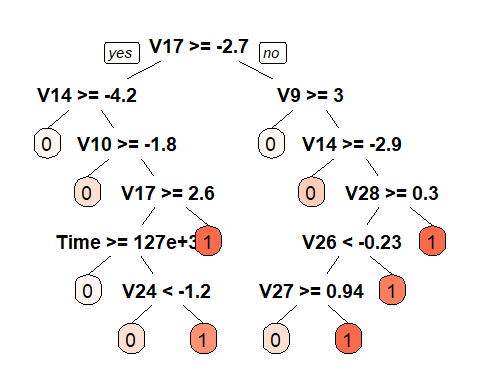
library(caret)  
library(rpart)  
library(rpart.plot)  
library(e1071)  
set.seed(123)  
control <- trainControl(method = "repeatedcv", number = 10, repeats = 3)  
ccf\_fit <- train(Class ~., data = training, method = "rpart",  
 parms = list(split = "information"),  
 trControl=control,  
 tuneLength = 10)

### STEP 3.1 : Checking the Trained Model

ccf\_fit

## CART   
##   
## 213606 samples  
## 30 predictor  
## 2 classes: '0', '1'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 192245, 192245, 192245, 192246, 192245, 192247, ...   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.0004516712 0.9994132 0.8173594  
## 0.0027100271 0.9994445 0.8245425  
## 0.0054200542 0.9994585 0.8287327  
## 0.0135501355 0.9994320 0.8220589  
## 0.0149051491 0.9994164 0.8190098  
## 0.0162601626 0.9993914 0.8147098  
## 0.0243902439 0.9993696 0.8143310  
## 0.0514905149 0.9992385 0.7558718  
## 0.0542005420 0.9992385 0.7558718  
## 0.5094850949 0.9983693 0.1071025  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.005420054.

prp(ccf\_fit$finalModel, box.palette = "Reds", tweak = 1.2)

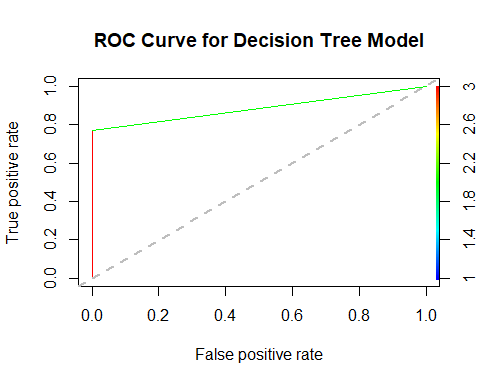


## STEP 4 : Evaluating Model Performance

cc\_pred <- predict(ccf\_fit,testing)  
confusionMatrix(cc\_pred,testing$Class)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 71065 28  
## 1 13 95  
##   
## Accuracy : 0.9994   
## 95% CI : (0.9992, 0.9996)   
## No Information Rate : 0.9983   
## P-Value [Acc > NIR] : < 0.0000000000000002  
##   
## Kappa : 0.8222   
##   
## Mcnemar's Test P-Value : 0.02878   
##   
## Sensitivity : 0.9998   
## Specificity : 0.7724   
## Pos Pred Value : 0.9996   
## Neg Pred Value : 0.8796   
## Prevalence : 0.9983   
## Detection Rate : 0.9981   
## Detection Prevalence : 0.9985   
## Balanced Accuracy : 0.8861   
##   
## 'Positive' Class : 0   
##

library(ROCR)  
predd <- prediction(as.numeric(cc\_pred), testing$Class)  
perfd <- performance(predd, 'tpr','fpr')  
plot(perfd, colorize = TRUE, text.adj = c(-0.2,1.7),main="ROC Curve for Decision Tree Model")  
abline(a=0,b=1,lwd=2,lty=2,col="grey")



# PART 2 : SUPPORT VECTOR MACHINE MODELLING

## STEP 1 till STEP 2.3 are as above used for Decision Tree Modelling

## STEP 2.4 :

library(caret)  
library(kernlab)

##   
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':  
##   
## alpha

set.seed(1404)  
svm\_ccf <- ksvm(Class~.,data=training,kernel="rbfdot")  
svm\_ccf

## Support Vector Machine object of class "ksvm"   
##   
## SV type: C-svc (classification)   
## parameter : cost C = 1   
##   
## Gaussian Radial Basis kernel function.   
## Hyperparameter : sigma = 0.0299466626224945   
##   
## Number of Support Vectors : 2862   
##   
## Objective Function Value : -298.0005   
## Training error : 0.000356

## STEP 4 : Evalutaing Model

svm\_ccf

## Support Vector Machine object of class "ksvm"   
##   
## SV type: C-svc (classification)   
## parameter : cost C = 1   
##   
## Gaussian Radial Basis kernel function.   
## Hyperparameter : sigma = 0.0299466626224945   
##   
## Number of Support Vectors : 2862   
##   
## Objective Function Value : -298.0005   
## Training error : 0.000356

## STEP 5 :

ccf\_prediction <- predict(svm\_ccf,testing)  
confusionMatrix(ccf\_prediction,testing$Class)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 71070 35  
## 1 8 88  
##   
## Accuracy : 0.9994   
## 95% CI : (0.9992, 0.9996)   
## No Information Rate : 0.9983   
## P-Value [Acc > NIR] : < 0.00000000000000022  
##   
## Kappa : 0.8034   
##   
## Mcnemar's Test P-Value : 0.00007341   
##   
## Sensitivity : 0.9999   
## Specificity : 0.7154   
## Pos Pred Value : 0.9995   
## Neg Pred Value : 0.9167   
## Prevalence : 0.9983   
## Detection Rate : 0.9982   
## Detection Prevalence : 0.9987   
## Balanced Accuracy : 0.8577   
##   
## 'Positive' Class : 0   
##

## STEP 6 : Improving Model Performance

## STEP 6.1 : Training SVM model with Kernel package for comparison

library(kernlab)  
svm\_ccf1 <- ksvm(Class~.,data=training,kernel="vanilladot")

## Setting default kernel parameters

svm\_ccf1

## Support Vector Machine object of class "ksvm"   
##   
## SV type: C-svc (classification)   
## parameter : cost C = 1   
##   
## Linear (vanilla) kernel function.   
##   
## Number of Support Vectors : 360   
##   
## Objective Function Value : -323.1737   
## Training error : 0.000571

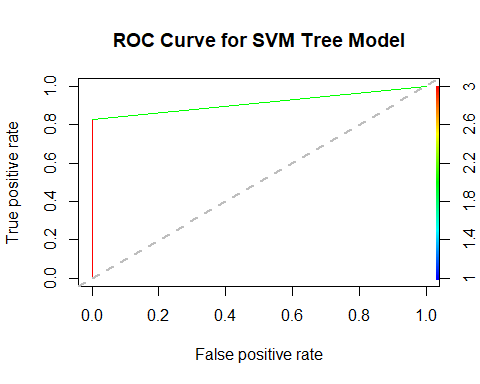
## STEP 6.2 : Evaluating Kernel Model

ccf\_prediction1 <- predict(svm\_ccf1,testing)  
confusionMatrix(ccf\_prediction1,testing$Class)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 71044 21  
## 1 34 102  
##   
## Accuracy : 0.9992   
## 95% CI : (0.999, 0.9994)   
## No Information Rate : 0.9983   
## P-Value [Acc > NIR] : 0.000000000004583  
##   
## Kappa : 0.7873   
##   
## Mcnemar's Test P-Value : 0.1056   
##   
## Sensitivity : 0.9995   
## Specificity : 0.8293   
## Pos Pred Value : 0.9997   
## Neg Pred Value : 0.7500   
## Prevalence : 0.9983   
## Detection Rate : 0.9978   
## Detection Prevalence : 0.9981   
## Balanced Accuracy : 0.9144   
##   
## 'Positive' Class : 0   
##

### STEP 6.3: ROC curve for SVM best model

library(ROCR)  
preds <- prediction(as.numeric(ccf\_prediction1), testing$Class)  
perfs <- performance(preds, 'tpr','fpr')  
plot(perfs, colorize = TRUE, text.adj = c(-0.2,1.7),main="ROC Curve for SVM Tree Model")  
abline(a=0,b=1,lwd=2,lty=2,col="grey")



### STEP 6.4: Comparing both the models

plot(perfd, col=1, lwd=3,avg= "threshold", main="ROC curve Decision Tree Vs SVM")  
plot(perfs, col=2, lwd=3, add=TRUE)  
legend(0.7, 0.7, c("Decision Tree","SVM"), 1:2)  
abline(a=0,b=1,lwd=2,lty=2,col="grey")

