R Notebook

# Load packages and libraries

# install.packages("moments")  
# install.packages("ggplot2")  
# install.packages("corrplot")  
# install.packages("tidyr")  
# install.packages("dplyr")  
# install.packages("ggridges")  
# install.packages("mlbench")  
# install.packages("lattice")  
# install.packages("caret")  
# install.packages("broom")  
# install.packages("C50")  
# install.packages("rpart")  
# install.packages("pROC")  
# install.packages("DMwR")  
# install.packages("ROSE")  
# install.packages('rattle')  
# install.packages('rpart.plot')  
# install.packages('RColorBrewer')  
# install.packages("glmboost")  
# install.packages("ROCR")  
library(moments)  
library(ggplot2)

## Warning: package 'ggplot2' was built under R version 3.4.4

library("corrplot")

## Warning: package 'corrplot' was built under R version 3.4.4

## corrplot 0.84 loaded

library(tidyr)

## Warning: package 'tidyr' was built under R version 3.4.4

library(dplyr)

## Warning: package 'dplyr' was built under R version 3.4.4

##   
## Attaching package: 'dplyr'

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

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

library(ggridges)

## Warning: package 'ggridges' was built under R version 3.4.4

library(mlbench)

## Warning: package 'mlbench' was built under R version 3.4.4

library(caret)

## Warning: package 'caret' was built under R version 3.4.4

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 3.4.4

library("C50")

## Warning: package 'C50' was built under R version 3.4.4

library(rpart)  
library(pROC)

## Warning: package 'pROC' was built under R version 3.4.4

## Type 'citation("pROC")' for a citation.

##   
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':  
##   
## cov, smooth, var

library(DMwR)

## Warning: package 'DMwR' was built under R version 3.4.4

## Loading required package: grid

library(ROSE)

## Warning: package 'ROSE' was built under R version 3.4.4

## Loaded ROSE 0.0-3

library(rattle)

## Warning: package 'rattle' was built under R version 3.4.4

## Rattle: A free graphical interface for data science with R.  
## Version 5.1.0 Copyright (c) 2006-2017 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.

library(rpart.plot)  
library(RColorBrewer)  
library(ROCR)

## Warning: package 'ROCR' was built under R version 3.4.4

## Loading required package: gplots

## Warning: package 'gplots' was built under R version 3.4.4

##   
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':  
##   
## lowess

# Load the dataset

ccfraud <- read.csv("C:/Users/Kiran Kandhola/Documents/creditcardfraud/creditcard.csv",stringsAsFactors = FALSE)

# Reorder the levels of the “Class” attribute

cc <- ccfraud # Make a copy of the data  
cc$Class <- as.factor(cc$Class) # Convert the class to factors  
cc$Class <- factor(cc$Class, levels = c("1", "0")) # Change the order of levels   
ccfd <- cc   
levels(ccfd$Class) <- c("Fraud", "Genuine") # Change the name of levels to Fraud and Genuine

## Stratified 80%- 20% splitting of the dataset into training and testing (time excluded)

set.seed(7)  
library(caret)  
train\_index <- createDataPartition(ccfd$Class,times = 1, p=0.8, list=FALSE)  
ccfd <- ccfd[ ,-c(1)]  
ccfd\_train <- ccfd[train\_index, ]  
ccfd\_test <- ccfd[-train\_index, ]  
  
zscorenormalize <- function(x){   
 return((x-mean(x))/(sd(x)))}  
  
Amount\_trn <- as.data.frame(lapply(ccfd\_train[29], zscorenormalize))  
ccfd\_train <- as.data.frame(c(ccfd\_train[1:28],Amount\_trn,ccfd\_train[30]))  
Amount\_tst <- as.data.frame(lapply(ccfd\_test[29], zscorenormalize))  
ccfd\_test <- as.data.frame(c(ccfd\_test[1:28],Amount\_tst,ccfd\_test[30]))  
  
X\_trn <- ccfd\_train[ ,-30]  
X\_tst <- ccfd\_test[ ,-30]  
  
Y\_trn <- ccfd\_train[ ,30]  
Y\_tst <- ccfd\_test[ ,30]  
  
## stratified folds for cross-validation:  
foldInd <- createFolds(Y\_trn, k = 10, list = TRUE, returnTrain = TRUE)  
lapply(foldInd, function(ii) table(Y\_trn[ii]))

## $Fold01  
##   
## Fraud Genuine   
## 354 204706   
##   
## $Fold02  
##   
## Fraud Genuine   
## 355 204707   
##   
## $Fold03  
##   
## Fraud Genuine   
## 355 204707   
##   
## $Fold04  
##   
## Fraud Genuine   
## 355 204707   
##   
## $Fold05  
##   
## Fraud Genuine   
## 354 204707   
##   
## $Fold06  
##   
## Fraud Genuine   
## 355 204707   
##   
## $Fold07  
##   
## Fraud Genuine   
## 355 204707   
##   
## $Fold08  
##   
## Fraud Genuine   
## 354 204707   
##   
## $Fold09  
##   
## Fraud Genuine   
## 354 204707   
##   
## $Fold10  
##   
## Fraud Genuine   
## 355 204706

set.seed(7)  
library(caret)  
train\_index <- createDataPartition(cc$Class,times = 1, p=0.8, list=FALSE)  
  
cc <- cc[ ,-c(1)]  
cc\_train <- cc[train\_index, ]  
cc\_test <- cc[-train\_index, ]  
  
zscorenormalize <- function(x){   
 return((x-mean(x))/(sd(x)))}  
Amount\_train <- as.data.frame(lapply(cc\_train[29], zscorenormalize))  
cc\_train <- as.data.frame(c(cc\_train[1:28],Amount\_train,cc\_train[30]))  
Amount\_test <- as.data.frame(lapply(ccfd\_test[29], zscorenormalize))  
cc\_test <- as.data.frame(c(ccfd\_test[1:28],Amount\_test,cc\_test[30]))  
  
X\_train <- cc\_train[ ,-30]  
X\_test <- cc\_test[ ,-30]  
  
Y\_train <- cc\_train[ ,30]  
Y\_test <- cc\_test[ ,30]  
  
## stratified folds for cross-validation:  
foldInds <- createFolds(Y\_train, k = 10, list = TRUE, returnTrain = TRUE)  
lapply(foldInds, function(ii) table(Y\_train[ii]))

## $Fold01  
##   
## 1 0   
## 355 204706   
##   
## $Fold02  
##   
## 1 0   
## 354 204707   
##   
## $Fold03  
##   
## 1 0   
## 355 204707   
##   
## $Fold04  
##   
## 1 0   
## 354 204707   
##   
## $Fold05  
##   
## 1 0   
## 354 204706   
##   
## $Fold06  
##   
## 1 0   
## 354 204707   
##   
## $Fold07  
##   
## 1 0   
## 355 204707   
##   
## $Fold08  
##   
## 1 0   
## 355 204707   
##   
## $Fold09  
##   
## 1 0   
## 355 204707   
##   
## $Fold10  
##   
## 1 0   
## 355 204707

# Training scheme for dataset using “smote” resampling

ctr\_s <- trainControl(index = foldInd,  
 method = "repeatedcv",  
 number = 10,  
 repeats = 3,  
 verboseIter = TRUE,  
 classProbs = TRUE,  
 sampling = "smote",  
 summaryFunction = twoClassSummary,  
 savePredictions = TRUE)

# rpart using “smote” resampling

set.seed(10)  
rpart\_s <- train(Class ~., data = ccfd\_train, method = "rpart", parms = list(split = "information"), metric = "ROC", trControl=ctr\_s, tuneLength = 30)

## + Fold01: cp=0   
## - Fold01: cp=0   
## + Fold02: cp=0   
## - Fold02: cp=0   
## + Fold03: cp=0   
## - Fold03: cp=0   
## + Fold04: cp=0   
## - Fold04: cp=0   
## + Fold05: cp=0   
## - Fold05: cp=0   
## + Fold06: cp=0   
## - Fold06: cp=0   
## + Fold07: cp=0   
## - Fold07: cp=0   
## + Fold08: cp=0   
## - Fold08: cp=0   
## + Fold09: cp=0   
## - Fold09: cp=0   
## + Fold10: cp=0   
## - Fold10: cp=0   
## Aggregating results  
## Selecting tuning parameters  
## Fitting cp = 0 on full training set

rpart\_s

## CART   
##   
## 227846 samples  
## 29 predictor  
## 2 classes: 'Fraud', 'Genuine'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 205060, 205062, 205062, 205062, 205061, 205062, ...   
## Addtional sampling using SMOTE  
##   
## Resampling results across tuning parameters:  
##   
## cp ROC Sens Spec   
## 0.00000000 0.9544226 0.8759615 0.9525130  
## 0.01654122 0.9110890 0.8352564 0.9840581  
## 0.03308244 0.9100925 0.8352564 0.9849286  
## 0.04962367 0.9100925 0.8352564 0.9849286  
## 0.06616489 0.9100925 0.8352564 0.9849286  
## 0.08270611 0.9100925 0.8352564 0.9849286  
## 0.09924733 0.9100925 0.8352564 0.9849286  
## 0.11578855 0.9100925 0.8352564 0.9849286  
## 0.13232977 0.9100925 0.8352564 0.9849286  
## 0.14887100 0.9100925 0.8352564 0.9849286  
## 0.16541222 0.9100925 0.8352564 0.9849286  
## 0.18195344 0.9100925 0.8352564 0.9849286  
## 0.19849466 0.9100925 0.8352564 0.9849286  
## 0.21503588 0.9100925 0.8352564 0.9849286  
## 0.23157710 0.9100925 0.8352564 0.9849286  
## 0.24811833 0.9100925 0.8352564 0.9849286  
## 0.26465955 0.9100925 0.8352564 0.9849286  
## 0.28120077 0.9100925 0.8352564 0.9849286  
## 0.29774199 0.9100925 0.8352564 0.9849286  
## 0.31428321 0.9100925 0.8352564 0.9849286  
## 0.33082444 0.9100925 0.8352564 0.9849286  
## 0.34736566 0.9100925 0.8352564 0.9849286  
## 0.36390688 0.9100925 0.8352564 0.9849286  
## 0.38044810 0.9100925 0.8352564 0.9849286  
## 0.39698932 0.9100925 0.8352564 0.9849286  
## 0.41353054 0.9100925 0.8352564 0.9849286  
## 0.43007177 0.9100925 0.8352564 0.9849286  
## 0.44661299 0.9100925 0.8352564 0.9849286  
## 0.46315421 0.9100925 0.8352564 0.9849286  
## 0.47969543 0.9100925 0.8352564 0.9849286  
##   
## ROC was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.

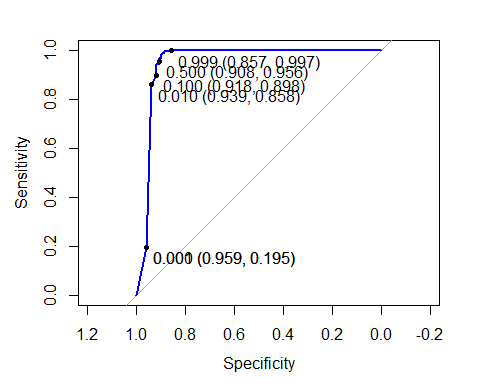
# test set results using confusion matrix  
pred\_rpart\_s <- predict(rpart\_s, ccfd\_test)  
confusionMatrix(pred\_rpart\_s, Y\_tst)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Fraud Genuine  
## Fraud 89 2519  
## Genuine 9 54344  
##   
## Accuracy : 0.9556   
## 95% CI : (0.9539, 0.9573)  
## No Information Rate : 0.9983   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.0627   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.908163   
## Specificity : 0.955701   
## Pos Pred Value : 0.034126   
## Neg Pred Value : 0.999834   
## Prevalence : 0.001720   
## Detection Rate : 0.001562   
## Detection Prevalence : 0.045786   
## Balanced Accuracy : 0.931932   
##   
## 'Positive' Class : Fraud   
##

# Predicting Class probabilities  
prob\_rpart\_s <- predict(rpart\_s, ccfd\_test, type = "prob")  
head(prob\_rpart\_s)

## Fraud Genuine  
## 1 0.000000000 1.0000000  
## 2 0.857142857 0.1428571  
## 3 0.127272727 0.8727273  
## 4 0.004562044 0.9954380  
## 5 0.004562044 0.9954380  
## 6 0.004562044 0.9954380

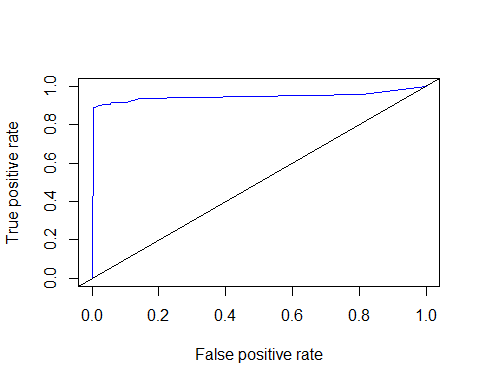
# AUC using pROC package  
ROC\_rpart\_s <- roc(Y\_tst, prob\_rpart\_s [ ,"Fraud"])  
plot(ROC\_rpart\_s, print.thres = c(0.0001,0.001, 0.01, 0.1, 0.5, 0.999), col = "blue")



ROC\_rpart\_s

##   
## Call:  
## roc.default(response = Y\_tst, predictor = prob\_rpart\_s[, "Fraud"])  
##   
## Data: prob\_rpart\_s[, "Fraud"] in 98 controls (Y\_tst Fraud) > 56863 cases (Y\_tst Genuine).  
## Area under the curve: 0.9485

# AUC using ROCR package  
pr\_rpart\_s <- prediction(prob\_rpart\_s[ , 1], Y\_test)  
prf\_rpart\_s <- performance(pr\_rpart\_s, measure = "tpr", x.measure = "fpr")  
plot(prf\_rpart\_s, col = "blue")  
abline(0,1)



auc\_rs <- performance(pr\_rpart\_s, measure = "auc")  
auc\_rs <- auc\_rs@y.values[[1]]  
auc\_rs

## [1] 0.9484651

# C5.0

set.seed(10)  
grid <- expand.grid(.model = "tree",   
 .trials = c(1:10),  
 .winnow = FALSE)  
C5\_s <- train(X\_trn, Y\_trn, method = "C5.0", metric = "ROC", trControl=ctr\_s, tuneGrid = grid)

## + Fold01: model=tree, winnow=FALSE, trials=10   
## - Fold01: model=tree, winnow=FALSE, trials=10   
## + Fold02: model=tree, winnow=FALSE, trials=10   
## - Fold02: model=tree, winnow=FALSE, trials=10   
## + Fold03: model=tree, winnow=FALSE, trials=10   
## - Fold03: model=tree, winnow=FALSE, trials=10   
## + Fold04: model=tree, winnow=FALSE, trials=10   
## - Fold04: model=tree, winnow=FALSE, trials=10   
## + Fold05: model=tree, winnow=FALSE, trials=10   
## - Fold05: model=tree, winnow=FALSE, trials=10   
## + Fold06: model=tree, winnow=FALSE, trials=10   
## - Fold06: model=tree, winnow=FALSE, trials=10   
## + Fold07: model=tree, winnow=FALSE, trials=10   
## - Fold07: model=tree, winnow=FALSE, trials=10   
## + Fold08: model=tree, winnow=FALSE, trials=10   
## - Fold08: model=tree, winnow=FALSE, trials=10   
## + Fold09: model=tree, winnow=FALSE, trials=10   
## - Fold09: model=tree, winnow=FALSE, trials=10   
## + Fold10: model=tree, winnow=FALSE, trials=10   
## - Fold10: model=tree, winnow=FALSE, trials=10   
## Aggregating results  
## Selecting tuning parameters  
## Fitting trials = 10, model = tree, winnow = FALSE on full training set

C5\_s

## C5.0   
##   
## 227846 samples  
## 29 predictor  
## 2 classes: 'Fraud', 'Genuine'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 205060, 205062, 205062, 205062, 205061, 205062, ...   
## Addtional sampling using SMOTE  
##   
## Resampling results across tuning parameters:  
##   
## trials ROC Sens Spec   
## 1 0.9591713 0.8858333 0.9599168  
## 2 0.9481912 0.8858333 0.9676766  
## 3 0.9579108 0.8833974 0.9718929  
## 4 0.9634560 0.8858974 0.9795649  
## 5 0.9654306 0.8911538 0.9791208  
## 6 0.9668697 0.8883974 0.9828095  
## 7 0.9699152 0.8935256 0.9812971  
## 8 0.9729013 0.8885256 0.9844011  
## 9 0.9722670 0.8960897 0.9834515  
## 10 0.9742447 0.8960897 0.9859707  
##   
## Tuning parameter 'model' was held constant at a value of tree  
##   
## Tuning parameter 'winnow' was held constant at a value of FALSE  
## ROC was used to select the optimal model using the largest value.  
## The final values used for the model were trials = 10, model = tree  
## and winnow = FALSE.

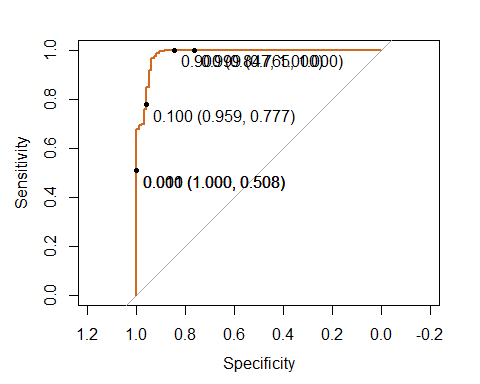
# test set results using confusion matrix  
pred\_C5\_s <- predict(C5\_s, ccfd\_test)  
confusionMatrix(pred\_C5\_s, Y\_tst)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Fraud Genuine  
## Fraud 89 759  
## Genuine 9 56104  
##   
## Accuracy : 0.9865   
## 95% CI : (0.9855, 0.9874)  
## No Information Rate : 0.9983   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.1856   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.908163   
## Specificity : 0.986652   
## Pos Pred Value : 0.104953   
## Neg Pred Value : 0.999840   
## Prevalence : 0.001720   
## Detection Rate : 0.001562   
## Detection Prevalence : 0.014887   
## Balanced Accuracy : 0.947408   
##   
## 'Positive' Class : Fraud   
##

# Predicting Class probabilities  
prob\_C5\_s <- predict(C5\_s, ccfd\_test, type = "prob")  
head(prob\_C5\_s)

## Fraud Genuine  
## 1 0.27787085 0.7221291  
## 2 0.00000000 1.0000000  
## 3 0.09812859 0.9018714  
## 4 0.09783363 0.9021664  
## 5 0.00000000 1.0000000  
## 6 0.00000000 1.0000000

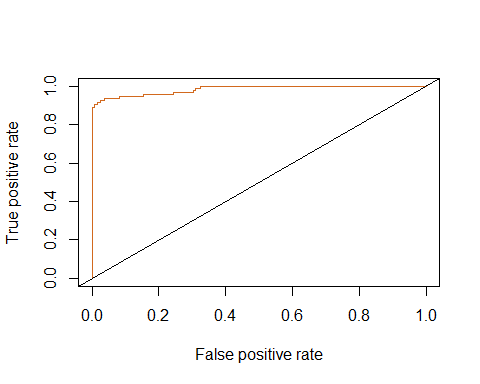
#library(pROC)  
ROC\_C5\_s <- roc(Y\_tst, prob\_C5\_s [ ,"Fraud"])  
plot(ROC\_C5\_s, print.thres = c(0.0001,0.001, 0.01, 0.1, 0.9, 0.999), col = "chocolate")



ROC\_C5\_s

##   
## Call:  
## roc.default(response = Y\_tst, predictor = prob\_C5\_s[, "Fraud"])  
##   
## Data: prob\_C5\_s[, "Fraud"] in 98 controls (Y\_tst Fraud) > 56863 cases (Y\_tst Genuine).  
## Area under the curve: 0.9845

# using ROCR package  
pr\_C5\_s <- prediction(prob\_C5\_s[ , 1], Y\_test)  
prf\_C5\_s <- performance(pr\_C5\_s, measure = "tpr", x.measure = "fpr")  
plot(prf\_C5\_s, col = "chocolate")  
abline(0,1)



auc\_C5s <- performance(pr\_C5\_s, measure = "auc")  
auc\_C5s <- auc\_C5s@y.values[[1]]  
auc\_C5s

## [1] 0.9844874

# Random forest

set.seed(7)  
rf\_s <- train(Class ~ ., data = ccfd\_train, method = "rf", trControl = ctr\_s, verbose = F, metric = "ROC", tuneLength = 9)

## + Fold01: mtry= 2   
## - Fold01: mtry= 2   
## + Fold01: mtry= 5   
## - Fold01: mtry= 5   
## + Fold01: mtry= 8   
## - Fold01: mtry= 8   
## + Fold01: mtry=12   
## - Fold01: mtry=12   
## + Fold01: mtry=15   
## - Fold01: mtry=15   
## + Fold01: mtry=18   
## - Fold01: mtry=18   
## + Fold01: mtry=22   
## - Fold01: mtry=22   
## + Fold01: mtry=25   
## - Fold01: mtry=25   
## + Fold01: mtry=29   
## - Fold01: mtry=29   
## + Fold02: mtry= 2   
## - Fold02: mtry= 2   
## + Fold02: mtry= 5   
## - Fold02: mtry= 5   
## + Fold02: mtry= 8   
## - Fold02: mtry= 8   
## + Fold02: mtry=12   
## - Fold02: mtry=12   
## + Fold02: mtry=15   
## - Fold02: mtry=15   
## + Fold02: mtry=18   
## - Fold02: mtry=18   
## + Fold02: mtry=22   
## - Fold02: mtry=22   
## + Fold02: mtry=25   
## - Fold02: mtry=25   
## + Fold02: mtry=29   
## - Fold02: mtry=29   
## + Fold03: mtry= 2   
## - Fold03: mtry= 2   
## + Fold03: mtry= 5   
## - Fold03: mtry= 5   
## + Fold03: mtry= 8   
## - Fold03: mtry= 8   
## + Fold03: mtry=12   
## - Fold03: mtry=12   
## + Fold03: mtry=15   
## - Fold03: mtry=15   
## + Fold03: mtry=18   
## - Fold03: mtry=18   
## + Fold03: mtry=22   
## - Fold03: mtry=22   
## + Fold03: mtry=25   
## - Fold03: mtry=25   
## + Fold03: mtry=29   
## - Fold03: mtry=29   
## + Fold04: mtry= 2   
## - Fold04: mtry= 2   
## + Fold04: mtry= 5   
## - Fold04: mtry= 5   
## + Fold04: mtry= 8   
## - Fold04: mtry= 8   
## + Fold04: mtry=12   
## - Fold04: mtry=12   
## + Fold04: mtry=15   
## - Fold04: mtry=15   
## + Fold04: mtry=18   
## - Fold04: mtry=18   
## + Fold04: mtry=22   
## - Fold04: mtry=22   
## + Fold04: mtry=25   
## - Fold04: mtry=25   
## + Fold04: mtry=29   
## - Fold04: mtry=29   
## + Fold05: mtry= 2   
## - Fold05: mtry= 2   
## + Fold05: mtry= 5   
## - Fold05: mtry= 5   
## + Fold05: mtry= 8   
## - Fold05: mtry= 8   
## + Fold05: mtry=12   
## - Fold05: mtry=12   
## + Fold05: mtry=15   
## - Fold05: mtry=15   
## + Fold05: mtry=18   
## - Fold05: mtry=18   
## + Fold05: mtry=22   
## - Fold05: mtry=22   
## + Fold05: mtry=25   
## - Fold05: mtry=25   
## + Fold05: mtry=29   
## - Fold05: mtry=29   
## + Fold06: mtry= 2   
## - Fold06: mtry= 2   
## + Fold06: mtry= 5   
## - Fold06: mtry= 5   
## + Fold06: mtry= 8   
## - Fold06: mtry= 8   
## + Fold06: mtry=12   
## - Fold06: mtry=12   
## + Fold06: mtry=15   
## - Fold06: mtry=15   
## + Fold06: mtry=18   
## - Fold06: mtry=18   
## + Fold06: mtry=22   
## - Fold06: mtry=22   
## + Fold06: mtry=25   
## - Fold06: mtry=25   
## + Fold06: mtry=29   
## - Fold06: mtry=29   
## + Fold07: mtry= 2   
## - Fold07: mtry= 2   
## + Fold07: mtry= 5   
## - Fold07: mtry= 5   
## + Fold07: mtry= 8   
## - Fold07: mtry= 8   
## + Fold07: mtry=12   
## - Fold07: mtry=12   
## + Fold07: mtry=15   
## - Fold07: mtry=15   
## + Fold07: mtry=18   
## - Fold07: mtry=18   
## + Fold07: mtry=22   
## - Fold07: mtry=22   
## + Fold07: mtry=25   
## - Fold07: mtry=25   
## + Fold07: mtry=29   
## - Fold07: mtry=29   
## + Fold08: mtry= 2   
## - Fold08: mtry= 2   
## + Fold08: mtry= 5   
## - Fold08: mtry= 5   
## + Fold08: mtry= 8   
## - Fold08: mtry= 8   
## + Fold08: mtry=12   
## - Fold08: mtry=12   
## + Fold08: mtry=15   
## - Fold08: mtry=15   
## + Fold08: mtry=18   
## - Fold08: mtry=18   
## + Fold08: mtry=22   
## - Fold08: mtry=22   
## + Fold08: mtry=25   
## - Fold08: mtry=25   
## + Fold08: mtry=29   
## - Fold08: mtry=29   
## + Fold09: mtry= 2   
## - Fold09: mtry= 2   
## + Fold09: mtry= 5   
## - Fold09: mtry= 5   
## + Fold09: mtry= 8   
## - Fold09: mtry= 8   
## + Fold09: mtry=12   
## - Fold09: mtry=12   
## + Fold09: mtry=15   
## - Fold09: mtry=15   
## + Fold09: mtry=18   
## - Fold09: mtry=18   
## + Fold09: mtry=22   
## - Fold09: mtry=22   
## + Fold09: mtry=25   
## - Fold09: mtry=25   
## + Fold09: mtry=29   
## - Fold09: mtry=29   
## + Fold10: mtry= 2   
## - Fold10: mtry= 2   
## + Fold10: mtry= 5   
## - Fold10: mtry= 5   
## + Fold10: mtry= 8   
## - Fold10: mtry= 8   
## + Fold10: mtry=12   
## - Fold10: mtry=12   
## + Fold10: mtry=15   
## - Fold10: mtry=15   
## + Fold10: mtry=18   
## - Fold10: mtry=18   
## + Fold10: mtry=22   
## - Fold10: mtry=22   
## + Fold10: mtry=25   
## - Fold10: mtry=25   
## + Fold10: mtry=29   
## - Fold10: mtry=29   
## Aggregating results  
## Selecting tuning parameters  
## Fitting mtry = 8 on full training set

rf\_s

## Random Forest   
##   
## 227846 samples  
## 29 predictor  
## 2 classes: 'Fraud', 'Genuine'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 205060, 205062, 205062, 205062, 205061, 205062, ...   
## Addtional sampling using SMOTE  
##   
## Resampling results across tuning parameters:  
##   
## mtry ROC Sens Spec   
## 2 0.9808728 0.8707051 0.9945835  
## 5 0.9810790 0.8834615 0.9925127  
## 8 0.9824736 0.8884615 0.9917389  
## 12 0.9802622 0.8885897 0.9910838  
## 15 0.9811005 0.8858333 0.9893032  
## 18 0.9815406 0.8910256 0.9890219  
## 22 0.9802109 0.8884615 0.9866741  
## 25 0.9800984 0.8908974 0.9872764  
## 29 0.9802600 0.8909615 0.9840890  
##   
## ROC was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 8.

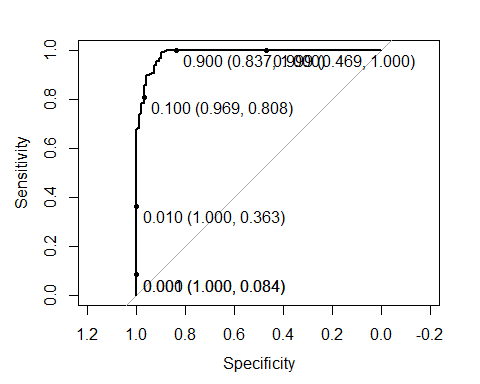
pred\_rf\_s <- predict(rf\_s, ccfd\_test)  
confusionMatrix(pred\_rf\_s, Y\_tst)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Fraud Genuine  
## Fraud 88 529  
## Genuine 10 56334  
##   
## Accuracy : 0.9905   
## 95% CI : (0.9897, 0.9913)  
## No Information Rate : 0.9983   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.2439   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.897959   
## Specificity : 0.990697   
## Pos Pred Value : 0.142626   
## Neg Pred Value : 0.999823   
## Prevalence : 0.001720   
## Detection Rate : 0.001545   
## Detection Prevalence : 0.010832   
## Balanced Accuracy : 0.944328   
##   
## 'Positive' Class : Fraud   
##

# Predicting Class probabilities  
prob\_rf\_s <- predict(rf\_s, ccfd\_test, type = "prob")  
head(prob\_rf\_s)

## Fraud Genuine  
## 1 0.058 0.942  
## 2 0.300 0.700  
## 3 0.122 0.878  
## 4 0.004 0.996  
## 5 0.026 0.974  
## 6 0.018 0.982

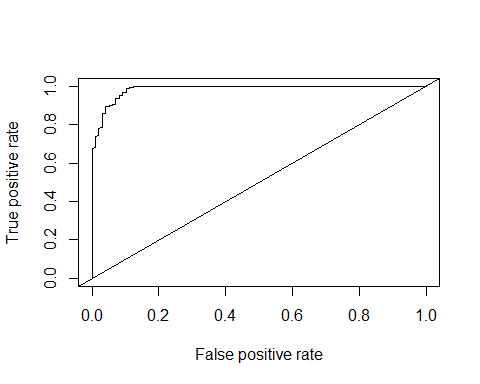
#library(pROC)  
ROC\_rf\_s <- roc(Y\_tst, prob\_rf\_s [ ,"Fraud"])  
plot(ROC\_rf\_s, print.thres = c(0.0001,0.001, 0.01, 0.1, 0.9, 0.999), col = "black")



ROC\_rf\_s

##   
## Call:  
## roc.default(response = Y\_tst, predictor = prob\_rf\_s[, "Fraud"])  
##   
## Data: prob\_rf\_s[, "Fraud"] in 98 controls (Y\_tst Fraud) > 56863 cases (Y\_tst Genuine).  
## Area under the curve: 0.9857

# using ROCR package  
pr\_rf\_s <- prediction(prob\_rf\_s[ , 2], Y\_tst)  
prf\_rf\_s <- performance(pr\_rf\_s, measure = "tpr", x.measure = "fpr")  
plot(prf\_rf\_s, col = "black")  
abline(0,1)



auc\_rf <- performance(pr\_rf\_s, measure = "auc")  
auc\_rf <- auc\_rf@y.values[[1]]  
auc\_rf

## [1] 0.9857106

# logistic Regression with smote resampling

set.seed(777)  
glm\_s <- train(Class ~ ., data=ccfd\_train, trControl=ctr\_s, method="glm", family=binomial(), metric = "ROC")

## + Fold01: parameter=none

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

## - Fold01: parameter=none   
## + Fold02: parameter=none

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

## - Fold02: parameter=none   
## + Fold03: parameter=none

## Warning: glm.fit: algorithm did not converge  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## - Fold03: parameter=none   
## + Fold04: parameter=none

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

## - Fold04: parameter=none   
## + Fold05: parameter=none

## Warning: glm.fit: algorithm did not converge  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## - Fold05: parameter=none   
## + Fold06: parameter=none

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

## - Fold06: parameter=none   
## + Fold07: parameter=none

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

## - Fold07: parameter=none   
## + Fold08: parameter=none

## Warning: glm.fit: algorithm did not converge  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## - Fold08: parameter=none   
## + Fold09: parameter=none

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

## - Fold09: parameter=none   
## + Fold10: parameter=none

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

## - Fold10: parameter=none   
## Aggregating results  
## Fitting final model on full training set

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

summary(glm\_s)

##   
## Call:  
## NULL  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.2651 -0.0019 0.0542 0.2494 7.2518   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 3.876338 0.231765 16.725 < 2e-16 \*\*\*  
## V1 0.004238 0.083611 0.051 0.959572   
## V2 -0.031261 0.202379 -0.154 0.877239   
## V3 0.121901 0.085970 1.418 0.156207   
## V4 -1.043031 0.130320 -8.004 1.21e-15 \*\*\*  
## V5 -0.189536 0.152815 -1.240 0.214865   
## V6 0.156063 0.121626 1.283 0.199444   
## V7 -0.376473 0.172681 -2.180 0.029246 \*   
## V8 0.425991 0.095593 4.456 8.34e-06 \*\*\*  
## V9 -0.037315 0.184748 -0.202 0.839934   
## V10 0.743150 0.191753 3.876 0.000106 \*\*\*  
## V11 -0.547650 0.111193 -4.925 8.43e-07 \*\*\*  
## V12 0.446267 0.112542 3.965 7.33e-05 \*\*\*  
## V13 0.425697 0.109158 3.900 9.63e-05 \*\*\*  
## V14 0.581081 0.097977 5.931 3.02e-09 \*\*\*  
## V15 -0.036744 0.113334 -0.324 0.745783   
## V16 -0.356836 0.232818 -1.533 0.125354   
## V17 0.084044 0.117528 0.715 0.474549   
## V18 0.354943 0.236882 1.498 0.134031   
## V19 -0.211967 0.161568 -1.312 0.189540   
## V20 -0.205349 0.210237 -0.977 0.328693   
## V21 -0.403944 0.154159 -2.620 0.008785 \*\*   
## V22 -0.544381 0.186048 -2.926 0.003433 \*\*   
## V23 0.242914 0.185668 1.308 0.190763   
## V24 -0.197579 0.207420 -0.953 0.340817   
## V25 -0.137405 0.213774 -0.643 0.520382   
## V26 0.463412 0.261882 1.770 0.076802 .   
## V27 -0.265797 0.288275 -0.922 0.356516   
## V28 -0.404553 0.379661 -1.066 0.286621   
## Amount -0.099496 0.439216 -0.227 0.820788   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 3766.92 on 2757 degrees of freedom  
## Residual deviance: 722.67 on 2728 degrees of freedom  
## AIC: 782.67  
##   
## Number of Fisher Scoring iterations: 10

summary(glm\_s)

##   
## Call:  
## NULL  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.2651 -0.0019 0.0542 0.2494 7.2518   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 3.876338 0.231765 16.725 < 2e-16 \*\*\*  
## V1 0.004238 0.083611 0.051 0.959572   
## V2 -0.031261 0.202379 -0.154 0.877239   
## V3 0.121901 0.085970 1.418 0.156207   
## V4 -1.043031 0.130320 -8.004 1.21e-15 \*\*\*  
## V5 -0.189536 0.152815 -1.240 0.214865   
## V6 0.156063 0.121626 1.283 0.199444   
## V7 -0.376473 0.172681 -2.180 0.029246 \*   
## V8 0.425991 0.095593 4.456 8.34e-06 \*\*\*  
## V9 -0.037315 0.184748 -0.202 0.839934   
## V10 0.743150 0.191753 3.876 0.000106 \*\*\*  
## V11 -0.547650 0.111193 -4.925 8.43e-07 \*\*\*  
## V12 0.446267 0.112542 3.965 7.33e-05 \*\*\*  
## V13 0.425697 0.109158 3.900 9.63e-05 \*\*\*  
## V14 0.581081 0.097977 5.931 3.02e-09 \*\*\*  
## V15 -0.036744 0.113334 -0.324 0.745783   
## V16 -0.356836 0.232818 -1.533 0.125354   
## V17 0.084044 0.117528 0.715 0.474549   
## V18 0.354943 0.236882 1.498 0.134031   
## V19 -0.211967 0.161568 -1.312 0.189540   
## V20 -0.205349 0.210237 -0.977 0.328693   
## V21 -0.403944 0.154159 -2.620 0.008785 \*\*   
## V22 -0.544381 0.186048 -2.926 0.003433 \*\*   
## V23 0.242914 0.185668 1.308 0.190763   
## V24 -0.197579 0.207420 -0.953 0.340817   
## V25 -0.137405 0.213774 -0.643 0.520382   
## V26 0.463412 0.261882 1.770 0.076802 .   
## V27 -0.265797 0.288275 -0.922 0.356516   
## V28 -0.404553 0.379661 -1.066 0.286621   
## Amount -0.099496 0.439216 -0.227 0.820788   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 3766.92 on 2757 degrees of freedom  
## Residual deviance: 722.67 on 2728 degrees of freedom  
## AIC: 782.67  
##   
## Number of Fisher Scoring iterations: 10

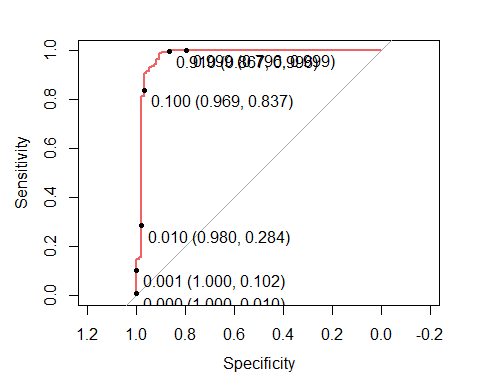
pred\_glm\_s <- predict(glm\_s, ccfd\_test)  
confusionMatrix(pred\_glm\_s, Y\_tst)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Fraud Genuine  
## Fraud 89 1226  
## Genuine 9 55637  
##   
## Accuracy : 0.9783   
## 95% CI : (0.9771, 0.9795)  
## No Information Rate : 0.9983   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.1232   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.908163   
## Specificity : 0.978439   
## Pos Pred Value : 0.067681   
## Neg Pred Value : 0.999838   
## Prevalence : 0.001720   
## Detection Rate : 0.001562   
## Detection Prevalence : 0.023086   
## Balanced Accuracy : 0.943301   
##   
## 'Positive' Class : Fraud   
##

# Predicting Class probabilities  
prob\_glm\_s <- predict(glm\_s, ccfd\_test, type = "prob")  
head(prob\_glm\_s)

## Fraud Genuine  
## 1 0.01584631 0.9841537  
## 2 0.13862224 0.8613778  
## 3 0.20760031 0.7923997  
## 4 0.11440742 0.8855926  
## 5 0.01818496 0.9818150  
## 6 0.01413300 0.9858670

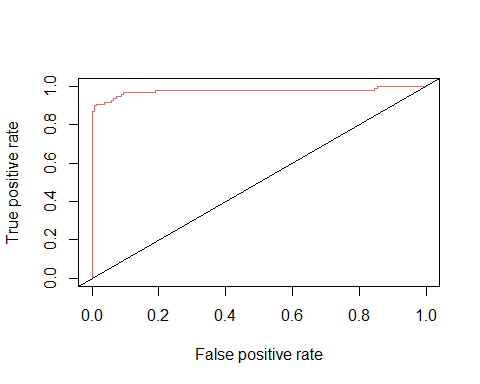
#library(pROC)  
ROC\_glm\_s <- roc(Y\_tst, prob\_glm\_s [ ,"Fraud"])  
plot(ROC\_glm\_s, print.thres = c(0.0001,0.001, 0.01, 0.1,0.91, 0.999), col = "indianred2")



ROC\_glm\_s

##   
## Call:  
## roc.default(response = Y\_tst, predictor = prob\_glm\_s[, "Fraud"])  
##   
## Data: prob\_glm\_s[, "Fraud"] in 98 controls (Y\_tst Fraud) > 56863 cases (Y\_tst Genuine).  
## Area under the curve: 0.9759

# using ROCR package  
pr\_glm\_s <- prediction(prob\_glm\_s[ , 1], Y\_test)  
prf\_glm\_s <- performance(pr\_glm\_s, measure = "tpr", x.measure = "fpr")  
plot(prf\_glm\_s, col = "indianred2")  
abline(0,1)



auc\_glm <- performance(pr\_glm\_s, measure = "auc")  
auc\_glm <- auc\_glm@y.values[[1]]  
auc\_glm

## [1] 0.9759029

# k-nn with smote resampling

gridk = expand.grid(k =c(5, 7, 9, 13, 17, 21, 27, 31, 37))

set.seed(7)  
knn\_s <- train(X\_trn, Y\_trn, method = "knn", metric = "ROC", trControl=ctr\_s, tuneGrid = gridk)

## + Fold01: k= 5   
## - Fold01: k= 5   
## + Fold01: k= 7   
## - Fold01: k= 7   
## + Fold01: k= 9   
## - Fold01: k= 9   
## + Fold01: k=13   
## - Fold01: k=13   
## + Fold01: k=17   
## - Fold01: k=17   
## + Fold01: k=21   
## - Fold01: k=21   
## + Fold01: k=27   
## - Fold01: k=27   
## + Fold01: k=31   
## - Fold01: k=31   
## + Fold01: k=37   
## - Fold01: k=37   
## + Fold02: k= 5   
## - Fold02: k= 5   
## + Fold02: k= 7   
## - Fold02: k= 7   
## + Fold02: k= 9   
## - Fold02: k= 9   
## + Fold02: k=13   
## - Fold02: k=13   
## + Fold02: k=17   
## - Fold02: k=17   
## + Fold02: k=21   
## - Fold02: k=21   
## + Fold02: k=27   
## - Fold02: k=27   
## + Fold02: k=31   
## - Fold02: k=31   
## + Fold02: k=37   
## - Fold02: k=37   
## + Fold03: k= 5   
## - Fold03: k= 5   
## + Fold03: k= 7   
## - Fold03: k= 7   
## + Fold03: k= 9   
## - Fold03: k= 9   
## + Fold03: k=13   
## - Fold03: k=13   
## + Fold03: k=17   
## - Fold03: k=17   
## + Fold03: k=21   
## - Fold03: k=21   
## + Fold03: k=27   
## - Fold03: k=27   
## + Fold03: k=31   
## - Fold03: k=31   
## + Fold03: k=37   
## - Fold03: k=37   
## + Fold04: k= 5   
## - Fold04: k= 5   
## + Fold04: k= 7   
## - Fold04: k= 7   
## + Fold04: k= 9   
## - Fold04: k= 9   
## + Fold04: k=13   
## - Fold04: k=13   
## + Fold04: k=17   
## - Fold04: k=17   
## + Fold04: k=21   
## - Fold04: k=21   
## + Fold04: k=27   
## - Fold04: k=27   
## + Fold04: k=31   
## - Fold04: k=31   
## + Fold04: k=37   
## - Fold04: k=37   
## + Fold05: k= 5   
## - Fold05: k= 5   
## + Fold05: k= 7   
## - Fold05: k= 7   
## + Fold05: k= 9   
## - Fold05: k= 9   
## + Fold05: k=13   
## - Fold05: k=13   
## + Fold05: k=17   
## - Fold05: k=17   
## + Fold05: k=21   
## - Fold05: k=21   
## + Fold05: k=27   
## - Fold05: k=27   
## + Fold05: k=31   
## - Fold05: k=31   
## + Fold05: k=37   
## - Fold05: k=37   
## + Fold06: k= 5   
## - Fold06: k= 5   
## + Fold06: k= 7   
## - Fold06: k= 7   
## + Fold06: k= 9   
## - Fold06: k= 9   
## + Fold06: k=13   
## - Fold06: k=13   
## + Fold06: k=17   
## - Fold06: k=17   
## + Fold06: k=21   
## - Fold06: k=21   
## + Fold06: k=27   
## - Fold06: k=27   
## + Fold06: k=31   
## - Fold06: k=31   
## + Fold06: k=37   
## - Fold06: k=37   
## + Fold07: k= 5   
## - Fold07: k= 5   
## + Fold07: k= 7   
## - Fold07: k= 7   
## + Fold07: k= 9   
## - Fold07: k= 9   
## + Fold07: k=13   
## - Fold07: k=13   
## + Fold07: k=17   
## - Fold07: k=17   
## + Fold07: k=21   
## - Fold07: k=21   
## + Fold07: k=27   
## - Fold07: k=27   
## + Fold07: k=31   
## - Fold07: k=31   
## + Fold07: k=37   
## - Fold07: k=37   
## + Fold08: k= 5   
## - Fold08: k= 5   
## + Fold08: k= 7   
## - Fold08: k= 7   
## + Fold08: k= 9   
## - Fold08: k= 9   
## + Fold08: k=13   
## - Fold08: k=13   
## + Fold08: k=17   
## - Fold08: k=17   
## + Fold08: k=21   
## - Fold08: k=21   
## + Fold08: k=27   
## - Fold08: k=27   
## + Fold08: k=31   
## - Fold08: k=31   
## + Fold08: k=37   
## - Fold08: k=37   
## + Fold09: k= 5   
## - Fold09: k= 5   
## + Fold09: k= 7   
## - Fold09: k= 7   
## + Fold09: k= 9   
## - Fold09: k= 9   
## + Fold09: k=13   
## - Fold09: k=13   
## + Fold09: k=17   
## - Fold09: k=17   
## + Fold09: k=21   
## - Fold09: k=21   
## + Fold09: k=27   
## - Fold09: k=27   
## + Fold09: k=31   
## - Fold09: k=31   
## + Fold09: k=37   
## - Fold09: k=37   
## + Fold10: k= 5   
## - Fold10: k= 5   
## + Fold10: k= 7   
## - Fold10: k= 7   
## + Fold10: k= 9   
## - Fold10: k= 9   
## + Fold10: k=13   
## - Fold10: k=13   
## + Fold10: k=17   
## - Fold10: k=17   
## + Fold10: k=21   
## - Fold10: k=21   
## + Fold10: k=27   
## - Fold10: k=27   
## + Fold10: k=31   
## - Fold10: k=31   
## + Fold10: k=37   
## - Fold10: k=37   
## Aggregating results  
## Selecting tuning parameters  
## Fitting k = 21 on full training set

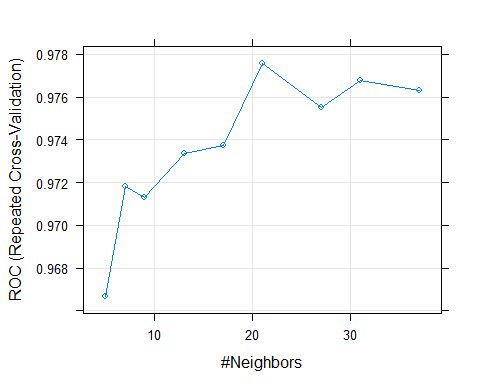
knn\_s

## k-Nearest Neighbors   
##   
## 227846 samples  
## 29 predictor  
## 2 classes: 'Fraud', 'Genuine'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 205060, 205062, 205062, 205062, 205061, 205062, ...   
## Addtional sampling using SMOTE  
##   
## Resampling results across tuning parameters:  
##   
## k ROC Sens Spec   
## 5 0.9666514 0.9036538 0.9688022  
## 7 0.9718198 0.9064103 0.9702882  
## 9 0.9713004 0.9037179 0.9750409  
## 13 0.9733484 0.8911538 0.9789714  
## 17 0.9737235 0.8911538 0.9806245  
## 21 0.9775901 0.8912179 0.9855486  
## 27 0.9755272 0.8759615 0.9875094  
## 31 0.9767819 0.8784615 0.9891054  
## 37 0.9763154 0.8683333 0.9903804  
##   
## ROC was used to select the optimal model using the largest value.  
## The final value used for the model was k = 21.

pred\_knn\_s <- predict(knn\_s, ccfd\_test)  
confusionMatrix(pred\_knn\_s, Y\_tst)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Fraud Genuine  
## Fraud 87 841  
## Genuine 11 56022  
##   
## Accuracy : 0.985   
## 95% CI : (0.984, 0.986)  
## No Information Rate : 0.9983   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.167   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.887755   
## Specificity : 0.985210   
## Pos Pred Value : 0.093750   
## Neg Pred Value : 0.999804   
## Prevalence : 0.001720   
## Detection Rate : 0.001527   
## Detection Prevalence : 0.016292   
## Balanced Accuracy : 0.936483   
##   
## 'Positive' Class : Fraud   
##

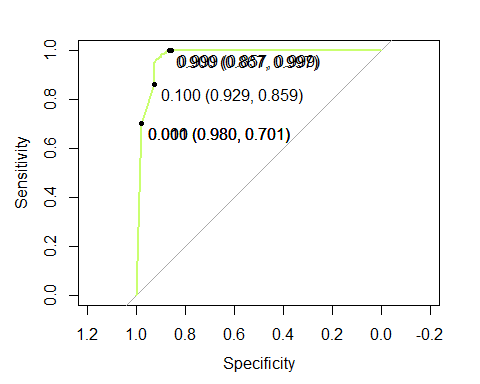
# Predicting Class probabilities  
plot(knn\_s)



prob\_knn\_s <- predict(knn\_s, ccfd\_test, type = "prob")  
head(prob\_knn\_s)

## Fraud Genuine  
## 1 0.0000000 1.0000000  
## 2 0.2380952 0.7619048  
## 3 0.0000000 1.0000000  
## 4 0.0000000 1.0000000  
## 5 0.0000000 1.0000000  
## 6 0.0000000 1.0000000

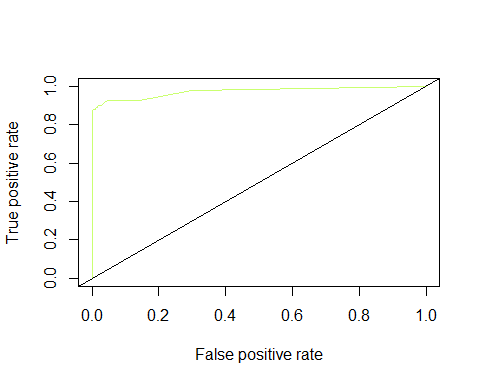
#library(pROC)  
ROC\_knn\_s <- roc(Y\_tst, prob\_knn\_s [ ,"Fraud"])  
plot(ROC\_knn\_s, print.thres = c(0.0001,0.001, 0.01, 0.1, 0.90, 0.999),col = "darkolivegreen1")



ROC\_knn\_s

##   
## Call:  
## roc.default(response = Y\_tst, predictor = prob\_knn\_s[, "Fraud"])  
##   
## Data: prob\_knn\_s[, "Fraud"] in 98 controls (Y\_tst Fraud) > 56863 cases (Y\_tst Genuine).  
## Area under the curve: 0.9738

# using ROCR package  
pr\_knn\_s <- prediction(prob\_knn\_s[ , 1], Y\_test)  
prf\_knn\_s <- performance(pr\_knn\_s, measure = "tpr", x.measure = "fpr")  
plot(prf\_knn\_s, col = "darkolivegreen1")  
abline(0,1)



auc\_knns <- performance(pr\_knn\_s, measure = "auc")  
auc\_knns <- auc\_knns@y.values[[1]]  
auc\_knns

## [1] 0.9737514

# SVM with smote resampling

set.seed(10)  
svm\_s <- train(Class ~ ., data = ccfd\_train, method = "svmRadial", trControl = ctr\_s, metric = "ROC", tuneLength = 9)

## + Fold01: sigma=0.03194, C= 0.25   
## - Fold01: sigma=0.03194, C= 0.25   
## + Fold01: sigma=0.03194, C= 0.50   
## - Fold01: sigma=0.03194, C= 0.50   
## + Fold01: sigma=0.03194, C= 1.00   
## - Fold01: sigma=0.03194, C= 1.00   
## + Fold01: sigma=0.03194, C= 2.00   
## - Fold01: sigma=0.03194, C= 2.00   
## + Fold01: sigma=0.03194, C= 4.00   
## - Fold01: sigma=0.03194, C= 4.00   
## + Fold01: sigma=0.03194, C= 8.00   
## - Fold01: sigma=0.03194, C= 8.00   
## + Fold01: sigma=0.03194, C=16.00   
## - Fold01: sigma=0.03194, C=16.00   
## + Fold01: sigma=0.03194, C=32.00   
## - Fold01: sigma=0.03194, C=32.00   
## + Fold01: sigma=0.03194, C=64.00   
## - Fold01: sigma=0.03194, C=64.00   
## + Fold02: sigma=0.03194, C= 0.25   
## - Fold02: sigma=0.03194, C= 0.25   
## + Fold02: sigma=0.03194, C= 0.50   
## - Fold02: sigma=0.03194, C= 0.50   
## + Fold02: sigma=0.03194, C= 1.00   
## - Fold02: sigma=0.03194, C= 1.00   
## + Fold02: sigma=0.03194, C= 2.00   
## - Fold02: sigma=0.03194, C= 2.00   
## + Fold02: sigma=0.03194, C= 4.00   
## - Fold02: sigma=0.03194, C= 4.00   
## + Fold02: sigma=0.03194, C= 8.00   
## - Fold02: sigma=0.03194, C= 8.00   
## + Fold02: sigma=0.03194, C=16.00   
## - Fold02: sigma=0.03194, C=16.00   
## + Fold02: sigma=0.03194, C=32.00   
## - Fold02: sigma=0.03194, C=32.00   
## + Fold02: sigma=0.03194, C=64.00   
## - Fold02: sigma=0.03194, C=64.00   
## + Fold03: sigma=0.03194, C= 0.25   
## - Fold03: sigma=0.03194, C= 0.25   
## + Fold03: sigma=0.03194, C= 0.50   
## - Fold03: sigma=0.03194, C= 0.50   
## + Fold03: sigma=0.03194, C= 1.00   
## - Fold03: sigma=0.03194, C= 1.00   
## + Fold03: sigma=0.03194, C= 2.00   
## - Fold03: sigma=0.03194, C= 2.00   
## + Fold03: sigma=0.03194, C= 4.00   
## - Fold03: sigma=0.03194, C= 4.00   
## + Fold03: sigma=0.03194, C= 8.00   
## - Fold03: sigma=0.03194, C= 8.00   
## + Fold03: sigma=0.03194, C=16.00   
## - Fold03: sigma=0.03194, C=16.00   
## + Fold03: sigma=0.03194, C=32.00   
## - Fold03: sigma=0.03194, C=32.00   
## + Fold03: sigma=0.03194, C=64.00   
## - Fold03: sigma=0.03194, C=64.00   
## + Fold04: sigma=0.03194, C= 0.25   
## - Fold04: sigma=0.03194, C= 0.25   
## + Fold04: sigma=0.03194, C= 0.50   
## - Fold04: sigma=0.03194, C= 0.50   
## + Fold04: sigma=0.03194, C= 1.00   
## - Fold04: sigma=0.03194, C= 1.00   
## + Fold04: sigma=0.03194, C= 2.00   
## - Fold04: sigma=0.03194, C= 2.00   
## + Fold04: sigma=0.03194, C= 4.00   
## - Fold04: sigma=0.03194, C= 4.00   
## + Fold04: sigma=0.03194, C= 8.00   
## - Fold04: sigma=0.03194, C= 8.00   
## + Fold04: sigma=0.03194, C=16.00   
## - Fold04: sigma=0.03194, C=16.00   
## + Fold04: sigma=0.03194, C=32.00   
## - Fold04: sigma=0.03194, C=32.00   
## + Fold04: sigma=0.03194, C=64.00   
## - Fold04: sigma=0.03194, C=64.00   
## + Fold05: sigma=0.03194, C= 0.25   
## - Fold05: sigma=0.03194, C= 0.25   
## + Fold05: sigma=0.03194, C= 0.50   
## - Fold05: sigma=0.03194, C= 0.50   
## + Fold05: sigma=0.03194, C= 1.00   
## - Fold05: sigma=0.03194, C= 1.00   
## + Fold05: sigma=0.03194, C= 2.00   
## - Fold05: sigma=0.03194, C= 2.00   
## + Fold05: sigma=0.03194, C= 4.00   
## - Fold05: sigma=0.03194, C= 4.00   
## + Fold05: sigma=0.03194, C= 8.00   
## - Fold05: sigma=0.03194, C= 8.00   
## + Fold05: sigma=0.03194, C=16.00   
## - Fold05: sigma=0.03194, C=16.00   
## + Fold05: sigma=0.03194, C=32.00   
## - Fold05: sigma=0.03194, C=32.00   
## + Fold05: sigma=0.03194, C=64.00   
## - Fold05: sigma=0.03194, C=64.00   
## + Fold06: sigma=0.03194, C= 0.25   
## - Fold06: sigma=0.03194, C= 0.25   
## + Fold06: sigma=0.03194, C= 0.50   
## - Fold06: sigma=0.03194, C= 0.50   
## + Fold06: sigma=0.03194, C= 1.00   
## - Fold06: sigma=0.03194, C= 1.00   
## + Fold06: sigma=0.03194, C= 2.00   
## - Fold06: sigma=0.03194, C= 2.00   
## + Fold06: sigma=0.03194, C= 4.00   
## - Fold06: sigma=0.03194, C= 4.00   
## + Fold06: sigma=0.03194, C= 8.00   
## - Fold06: sigma=0.03194, C= 8.00   
## + Fold06: sigma=0.03194, C=16.00   
## - Fold06: sigma=0.03194, C=16.00   
## + Fold06: sigma=0.03194, C=32.00   
## - Fold06: sigma=0.03194, C=32.00   
## + Fold06: sigma=0.03194, C=64.00   
## - Fold06: sigma=0.03194, C=64.00   
## + Fold07: sigma=0.03194, C= 0.25   
## - Fold07: sigma=0.03194, C= 0.25   
## + Fold07: sigma=0.03194, C= 0.50   
## - Fold07: sigma=0.03194, C= 0.50   
## + Fold07: sigma=0.03194, C= 1.00   
## - Fold07: sigma=0.03194, C= 1.00   
## + Fold07: sigma=0.03194, C= 2.00   
## - Fold07: sigma=0.03194, C= 2.00   
## + Fold07: sigma=0.03194, C= 4.00   
## - Fold07: sigma=0.03194, C= 4.00   
## + Fold07: sigma=0.03194, C= 8.00   
## - Fold07: sigma=0.03194, C= 8.00   
## + Fold07: sigma=0.03194, C=16.00   
## - Fold07: sigma=0.03194, C=16.00   
## + Fold07: sigma=0.03194, C=32.00   
## - Fold07: sigma=0.03194, C=32.00   
## + Fold07: sigma=0.03194, C=64.00   
## - Fold07: sigma=0.03194, C=64.00   
## + Fold08: sigma=0.03194, C= 0.25   
## - Fold08: sigma=0.03194, C= 0.25   
## + Fold08: sigma=0.03194, C= 0.50   
## - Fold08: sigma=0.03194, C= 0.50   
## + Fold08: sigma=0.03194, C= 1.00   
## - Fold08: sigma=0.03194, C= 1.00   
## + Fold08: sigma=0.03194, C= 2.00   
## - Fold08: sigma=0.03194, C= 2.00   
## + Fold08: sigma=0.03194, C= 4.00   
## - Fold08: sigma=0.03194, C= 4.00   
## + Fold08: sigma=0.03194, C= 8.00   
## - Fold08: sigma=0.03194, C= 8.00   
## + Fold08: sigma=0.03194, C=16.00   
## - Fold08: sigma=0.03194, C=16.00   
## + Fold08: sigma=0.03194, C=32.00   
## - Fold08: sigma=0.03194, C=32.00   
## + Fold08: sigma=0.03194, C=64.00   
## - Fold08: sigma=0.03194, C=64.00   
## + Fold09: sigma=0.03194, C= 0.25   
## - Fold09: sigma=0.03194, C= 0.25   
## + Fold09: sigma=0.03194, C= 0.50   
## - Fold09: sigma=0.03194, C= 0.50   
## + Fold09: sigma=0.03194, C= 1.00   
## - Fold09: sigma=0.03194, C= 1.00   
## + Fold09: sigma=0.03194, C= 2.00   
## - Fold09: sigma=0.03194, C= 2.00   
## + Fold09: sigma=0.03194, C= 4.00   
## - Fold09: sigma=0.03194, C= 4.00   
## + Fold09: sigma=0.03194, C= 8.00   
## - Fold09: sigma=0.03194, C= 8.00   
## + Fold09: sigma=0.03194, C=16.00   
## - Fold09: sigma=0.03194, C=16.00   
## + Fold09: sigma=0.03194, C=32.00   
## - Fold09: sigma=0.03194, C=32.00   
## + Fold09: sigma=0.03194, C=64.00   
## - Fold09: sigma=0.03194, C=64.00   
## + Fold10: sigma=0.03194, C= 0.25   
## - Fold10: sigma=0.03194, C= 0.25   
## + Fold10: sigma=0.03194, C= 0.50   
## - Fold10: sigma=0.03194, C= 0.50   
## + Fold10: sigma=0.03194, C= 1.00   
## - Fold10: sigma=0.03194, C= 1.00   
## + Fold10: sigma=0.03194, C= 2.00   
## - Fold10: sigma=0.03194, C= 2.00   
## + Fold10: sigma=0.03194, C= 4.00   
## - Fold10: sigma=0.03194, C= 4.00   
## + Fold10: sigma=0.03194, C= 8.00   
## - Fold10: sigma=0.03194, C= 8.00   
## + Fold10: sigma=0.03194, C=16.00   
## - Fold10: sigma=0.03194, C=16.00   
## + Fold10: sigma=0.03194, C=32.00   
## - Fold10: sigma=0.03194, C=32.00   
## + Fold10: sigma=0.03194, C=64.00   
## - Fold10: sigma=0.03194, C=64.00   
## Aggregating results  
## Selecting tuning parameters  
## Fitting sigma = 0.0319, C = 0.25 on full training set

svm\_s

## Support Vector Machines with Radial Basis Function Kernel   
##   
## 227846 samples  
## 29 predictor  
## 2 classes: 'Fraud', 'Genuine'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 205060, 205062, 205062, 205062, 205061, 205062, ...   
## Addtional sampling using SMOTE  
##   
## Resampling results across tuning parameters:  
##   
## C ROC Sens Spec   
## 0.25 0.9831416 0.8858974 0.9738714  
## 0.50 0.9827293 0.9062179 0.9759729  
## 1.00 0.9824632 0.9037179 0.9766544  
## 2.00 0.9820502 0.9087821 0.9793406  
## 4.00 0.9812171 0.9062821 0.9796660  
## 8.00 0.9793465 0.9037179 0.9796177  
## 16.00 0.9790329 0.8862179 0.9809718  
## 32.00 0.9758256 0.8886538 0.9782327  
## 64.00 0.9675512 0.8732692 0.9792528  
##   
## Tuning parameter 'sigma' was held constant at a value of 0.03194264  
## ROC was used to select the optimal model using the largest value.  
## The final values used for the model were sigma = 0.03194264 and C = 0.25.

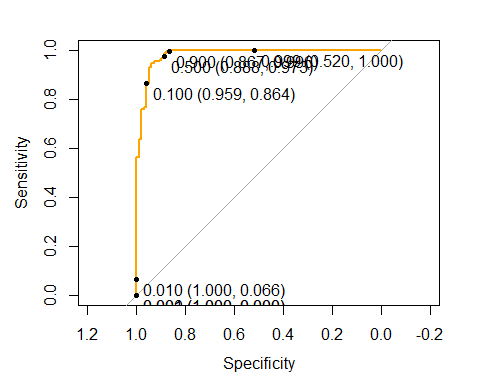
pred\_svm\_s <- predict(svm\_s, ccfd\_test)  
confusionMatrix(pred\_svm\_s, Y\_tst)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Fraud Genuine  
## Fraud 87 1522  
## Genuine 11 55341  
##   
## Accuracy : 0.9731   
## 95% CI : (0.9717, 0.9744)  
## No Information Rate : 0.9983   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.099   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.887755   
## Specificity : 0.973234   
## Pos Pred Value : 0.054071   
## Neg Pred Value : 0.999801   
## Prevalence : 0.001720   
## Detection Rate : 0.001527   
## Detection Prevalence : 0.028247   
## Balanced Accuracy : 0.930495   
##   
## 'Positive' Class : Fraud   
##

# Predicting Class probabilities  
prob\_svm\_s <- predict(svm\_s, ccfd\_test, type = "prob")  
head(prob\_svm\_s)

## Fraud Genuine  
## 1 0.03051257 0.9694874  
## 2 0.39767933 0.6023207  
## 3 0.12857562 0.8714244  
## 4 0.06987936 0.9301206  
## 5 0.02584747 0.9741525  
## 6 0.02750934 0.9724907

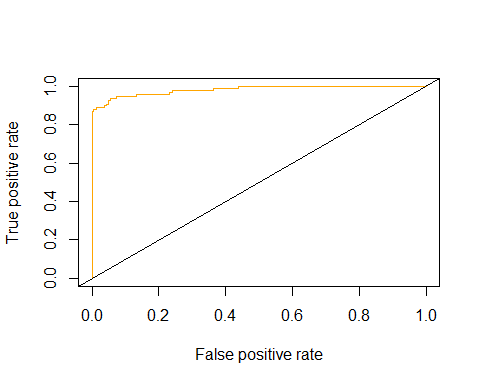
#library(pROC)  
ROC\_svm\_s <- roc(Y\_tst, prob\_svm\_s [ ,"Fraud"])  
plot(ROC\_svm\_s, print.thres = c(0.0001,0.001, 0.01, 0.1, 0.5, 0.9, 0.999), col = "orange")



ROC\_svm\_s

##   
## Call:  
## roc.default(response = Y\_tst, predictor = prob\_svm\_s[, "Fraud"])  
##   
## Data: prob\_svm\_s[, "Fraud"] in 98 controls (Y\_tst Fraud) > 56863 cases (Y\_tst Genuine).  
## Area under the curve: 0.9822

# using ROCR package  
pr\_svm\_s <- prediction(prob\_svm\_s[ , 1], Y\_test)  
prf\_svm\_s <- performance(pr\_svm\_s, measure = "tpr", x.measure = "fpr")  
plot(prf\_svm\_s, col = "orange")  
abline(0,1)



auc\_svm <- performance(pr\_svm\_s, measure = "auc")  
auc\_svm <- auc\_svm@y.values[[1]]  
auc\_svm

## [1] 0.9822129

plot(prf\_rpart\_s,lty = 1,lwd = 2, col = "blue")  
plot(prf\_rf\_s,lty = 1, lwd = 2, col = "black", add = TRUE)  
plot(prf\_glm\_s,lty = 1, lwd = 2, col = "indianred2", add = TRUE)  
plot(prf\_knn\_s,lty = 1, lwd = 2, col = "darkolivegreen1", add = TRUE)  
plot(prf\_svm\_s,lty = 1, lwd = 2, col = "orange", add = TRUE)  
legend ("topright", legend = c("rpart","rf", "glm", "knn", "svm"), col = c("blue", "black", "indianred2", "darkolivegreen1", "orange"), lwd = c(2,2,2,2,2,2))

