DATA 622 - Homework 2

OMER OZEREN

Table of Contents

# PART A

STEP#0: Pick any two classifiers of (SVM,Logistic,DecisionTree,NaiveBayes). Pick heart or ecoli dataset. Heart is simpler and ecoli compounds the problem as it is NOT a balanced dataset. From a grading perspective both carry the same weight.

STEP#1 For each classifier, Set a seed (43)

STEP#2 Do a 80/20 split and determine the Accuracy, AUC and as many metrics as returned by the Caret package (confusionMatrix) Call this the base\_metric. Note down as best as you can development (engineering) cost as well as computing cost(elapsed time). Start with the original dataset and set a seed (43). Then run a cross validation of 5 and 10 of the model on the training set. Determine the same set of metrics and compare the cv\_metrics with the base\_metric. Note down as best as you can development (engineering) cost as well as computing cost(elapsed time). Start with the original dataset and set a seed (43) Then run a bootstrap of 200 resamples and compute the same set of metrics and for each of the two classifiers build a three column table for each experiment (base, bootstrap, cross-validated). Note down as best as you can development (engineering) cost as well as computing cost(elapsed time).

## Import Data

data <- read.csv("https://raw.githubusercontent.com/omerozeren/DATA622/master/heart.csv",head=T,sep=',',stringsAsFactors=F, fileEncoding = "UTF-8-BOM")

## Split Data into Train(80%) and Test data(20%)

set.seed(43)  
split\_df <- createDataPartition(data$target, p = .80, list = FALSE)  
data\_train <- data[split\_df,]  
data\_test <- data[-split\_df,]

## Model Performance Estimater

estimate\_model\_performance <- function(y\_true, y\_pred, model\_name){  
 cm <- confusionMatrix(table(y\_true, y\_pred))  
 cm\_table <- cm$table  
 tpr <- cm\_table[[1]] / (cm\_table[[1]] + cm\_table[[4]])  
 fnr <- 1 - tpr  
 fpr <- cm\_table[[3]] / (cm\_table[[3]] + cm\_table[[4]])  
 tnr <- 1 - fpr  
 accuracy <- cm$overall[[1]]  
 for\_auc <- prediction(c(y\_pred), y\_true)  
 auc <- performance(for\_auc, "auc")  
 auc <- auc@y.values[[1]]  
 return(data.frame(Algo = model\_name, AUC = auc, ACCURACY = accuracy, TPR = tpr, FPR = fpr, TNR = tnr, FNR = fnr))  
}

## NaiveBayes Model

start\_tm <- proc.time()  
nb\_model<-naiveBayes(data\_train$target~.,data=data\_train)  
object.size(nb\_model)

## 16344 bytes

nb\_testpred<-predict(nb\_model,data\_test,type='raw')  
nb\_testclass<-unlist(apply(round(nb\_testpred),1,which.max))-1  
nb\_table<-table(data\_test$target, nb\_testclass)  
base\_metric\_nb<-caret::confusionMatrix(nb\_table)  
base\_metric\_nb

## Confusion Matrix and Statistics  
##   
## nb\_testclass  
## 0 1  
## 0 22 9  
## 1 2 27  
##   
## Accuracy : 0.8167   
## 95% CI : (0.6956, 0.9048)  
## No Information Rate : 0.6   
## P-Value [Acc > NIR] : 0.0002826   
##   
## Kappa : 0.6358   
##   
## Mcnemar's Test P-Value : 0.0704404   
##   
## Sensitivity : 0.9167   
## Specificity : 0.7500   
## Pos Pred Value : 0.7097   
## Neg Pred Value : 0.9310   
## Prevalence : 0.4000   
## Detection Rate : 0.3667   
## Detection Prevalence : 0.5167   
## Balanced Accuracy : 0.8333   
##   
## 'Positive' Class : 0   
##

end\_tm<-proc.time()   
print(paste("time taken to run NaiveBayes Model",(end\_tm-start\_tm),sep=":"))

## [1] "time taken to run NaiveBayes Model:0.0300000000000002"  
## [2] "time taken to run NaiveBayes Model:0"   
## [3] "time taken to run NaiveBayes Model:0.04"   
## [4] "time taken to run NaiveBayes Model:NA"   
## [5] "time taken to run NaiveBayes Model:NA"

### Estimate NB model test data () performance

base\_metric\_nb\_table\_standalone<-estimate\_model\_performance(data\_test$target,nb\_testclass,'NB')  
base\_metric\_nb\_table\_standalone

## Algo AUC ACCURACY TPR FPR TNR FNR  
## 1 NB 0.820356 0.8166667 0.4489796 0.25 0.75 0.5510204

### NaiveBayes with Cross Validation folds = 5

set.seed(43)  
start\_tm <- proc.time()  
df <- data[sample(nrow(data)),]  
folds <- cut(seq(1,nrow(data)),breaks=5,labels=FALSE)  
nb\_pred <- list()  
nb\_testclass <- list()  
nb\_testclass<-list()  
nb\_table <- list()  
base\_metric\_nb <- list()  
base\_metric\_nb\_table\_cv\_5 <- list()  
for(i in 1:5){  
 testIndexes <- which(folds==i,arr.ind=TRUE)  
 testData <- df[testIndexes, ]  
 trainData <- df[-testIndexes, ]  
 nb\_model <- naiveBayes(trainData$target ~ .,data=trainData) # naiveBayes(data\_train$target~.,data=data\_train)  
 nb\_pred[[i]]<-predict(nb\_model,testData,type='raw')  
 nb\_testclass[[i]]<-unlist(apply(round(nb\_pred[[i]]),1,which.max))-1  
 nb\_table[[i]]<-table(testData$target, nb\_testclass[[i]])  
 base\_metric\_nb[[i]]<-caret::confusionMatrix(nb\_table[[i]])  
 base\_metric\_nb\_table\_cv\_5[[i]]<-estimate\_model\_performance(testData$target,nb\_testclass[[i]],paste('NB fold',i,sep =":" ))  
}  
  
end\_tm<-proc.time()   
  
print(paste("time taken to run NaiveBayes Model with CV with 5 Folds",(end\_tm-start\_tm),sep=":"))

## [1] "time taken to run NaiveBayes Model with CV with 5 Folds:0.17"  
## [2] "time taken to run NaiveBayes Model with CV with 5 Folds:0"   
## [3] "time taken to run NaiveBayes Model with CV with 5 Folds:0.2"   
## [4] "time taken to run NaiveBayes Model with CV with 5 Folds:NA"   
## [5] "time taken to run NaiveBayes Model with CV with 5 Folds:NA"

### Base Metric for NaiveBayes with Cross Validation folds = 5

base\_metric\_nb

## [[1]]  
## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 20 4  
## 1 6 31  
##   
## Accuracy : 0.8361   
## 95% CI : (0.7191, 0.9185)  
## No Information Rate : 0.5738   
## P-Value [Acc > NIR] : 1.18e-05   
##   
## Kappa : 0.6615   
##   
## Mcnemar's Test P-Value : 0.7518   
##   
## Sensitivity : 0.7692   
## Specificity : 0.8857   
## Pos Pred Value : 0.8333   
## Neg Pred Value : 0.8378   
## Prevalence : 0.4262   
## Detection Rate : 0.3279   
## Detection Prevalence : 0.3934   
## Balanced Accuracy : 0.8275   
##   
## 'Positive' Class : 0   
##   
##   
## [[2]]  
## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 22 3  
## 1 6 29  
##   
## Accuracy : 0.85   
## 95% CI : (0.7343, 0.929)  
## No Information Rate : 0.5333   
## P-Value [Acc > NIR] : 2.293e-07   
##   
## Kappa : 0.6966   
##   
## Mcnemar's Test P-Value : 0.505   
##   
## Sensitivity : 0.7857   
## Specificity : 0.9062   
## Pos Pred Value : 0.8800   
## Neg Pred Value : 0.8286   
## Prevalence : 0.4667   
## Detection Rate : 0.3667   
## Detection Prevalence : 0.4167   
## Balanced Accuracy : 0.8460   
##   
## 'Positive' Class : 0   
##   
##   
## [[3]]  
## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 20 10  
## 1 3 28  
##   
## Accuracy : 0.7869   
## 95% CI : (0.6632, 0.8814)  
## No Information Rate : 0.623   
## P-Value [Acc > NIR] : 0.004731   
##   
## Kappa : 0.572   
##   
## Mcnemar's Test P-Value : 0.096092   
##   
## Sensitivity : 0.8696   
## Specificity : 0.7368   
## Pos Pred Value : 0.6667   
## Neg Pred Value : 0.9032   
## Prevalence : 0.3770   
## Detection Rate : 0.3279   
## Detection Prevalence : 0.4918   
## Balanced Accuracy : 0.8032   
##   
## 'Positive' Class : 0   
##   
##   
## [[4]]  
## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 24 4  
## 1 7 25  
##   
## Accuracy : 0.8167   
## 95% CI : (0.6956, 0.9048)  
## No Information Rate : 0.5167   
## P-Value [Acc > NIR] : 1.322e-06   
##   
## Kappa : 0.6341   
##   
## Mcnemar's Test P-Value : 0.5465   
##   
## Sensitivity : 0.7742   
## Specificity : 0.8621   
## Pos Pred Value : 0.8571   
## Neg Pred Value : 0.7812   
## Prevalence : 0.5167   
## Detection Rate : 0.4000   
## Detection Prevalence : 0.4667   
## Balanced Accuracy : 0.8181   
##   
## 'Positive' Class : 0   
##   
##   
## [[5]]  
## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 22 9  
## 1 2 28  
##   
## Accuracy : 0.8197   
## 95% CI : (0.7002, 0.9064)  
## No Information Rate : 0.6066   
## P-Value [Acc > NIR] : 0.000298   
##   
## Kappa : 0.6406   
##   
## Mcnemar's Test P-Value : 0.070440   
##   
## Sensitivity : 0.9167   
## Specificity : 0.7568   
## Pos Pred Value : 0.7097   
## Neg Pred Value : 0.9333   
## Prevalence : 0.3934   
## Detection Rate : 0.3607   
## Detection Prevalence : 0.5082   
## Balanced Accuracy : 0.8367   
##   
## 'Positive' Class : 0   
##

### The Mean of NaiveBayes with Cross Validation folds = 5

rst<-do.call(rbind.data.frame, base\_metric\_nb\_table\_cv\_5)  
base\_metric\_nb\_table\_cv\_5\_mean<-data.frame(cbind(Algo='NB\_CV\_5',AUC=mean(rst$ACCURACY),ACCURACY=mean(rst$ACCURACY),TPR=mean(rst$TPR),FPR=mean(rst$FPR),TNR=mean(rst$TNR),FNR=mean(rst$FNR)))  
base\_metric\_nb\_table\_cv\_5\_mean

## Algo AUC ACCURACY TPR  
## 1 NB\_CV\_5 0.821857923497268 0.821857923497268 0.433998399359744  
## FPR TNR FNR  
## 1 0.170473577349712 0.829526422650288 0.566001600640256

### NaiveBayes with Cross Validation folds = 10

set.seed(43)  
df <- data[sample(nrow(data)),]  
folds <- cut(seq(1,nrow(data)),breaks=10,labels=FALSE)  
nb\_pred <- list()  
nb\_testclass <- list()  
nb\_testclass<-list()  
nb\_table <- list()  
base\_metric\_nb <- list()  
base\_metric\_nb\_table\_cv\_10 <- list()  
for(i in 1:10){  
 testIndexes <- which(folds==i,arr.ind=TRUE)  
 testData <- df[testIndexes, ]  
 trainData <- df[-testIndexes, ]  
 nb\_model <- naiveBayes(trainData$target ~ .,data=trainData) # naiveBayes(data\_train$target~.,data=data\_train)  
 nb\_pred[[i]]<-predict(nb\_model,testData,type='raw')  
 nb\_testclass[[i]]<-unlist(apply(round(nb\_pred[[i]]),1,which.max))-1  
 nb\_table[[i]]<-table(testData$target, nb\_testclass[[i]])  
 base\_metric\_nb[[i]]<-caret::confusionMatrix(nb\_table[[i]])  
 base\_metric\_nb\_table\_cv\_10[[i]]<-estimate\_model\_performance(testData$target,nb\_testclass[[i]],paste('NB fold',i,sep =":" ))  
}

### Base Metric for NaiveBayes with Cross Validation folds = 10

base\_metric\_nb

## [[1]]  
## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 8 4  
## 1 1 18  
##   
## Accuracy : 0.8387   
## 95% CI : (0.6627, 0.9455)  
## No Information Rate : 0.7097   
## P-Value [Acc > NIR] : 0.07793   
##   
## Kappa : 0.6437   
##   
## Mcnemar's Test P-Value : 0.37109   
##   
## Sensitivity : 0.8889   
## Specificity : 0.8182   
## Pos Pred Value : 0.6667   
## Neg Pred Value : 0.9474   
## Prevalence : 0.2903   
## Detection Rate : 0.2581   
## Detection Prevalence : 0.3871   
## Balanced Accuracy : 0.8535   
##   
## 'Positive' Class : 0   
##   
##   
## [[2]]  
## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 10 2  
## 1 6 12  
##   
## Accuracy : 0.7333   
## 95% CI : (0.5411, 0.8772)  
## No Information Rate : 0.5333   
## P-Value [Acc > NIR] : 0.02046   
##   
## Kappa : 0.4737   
##   
## Mcnemar's Test P-Value : 0.28884   
##   
## Sensitivity : 0.6250   
## Specificity : 0.8571   
## Pos Pred Value : 0.8333   
## Neg Pred Value : 0.6667   
## Prevalence : 0.5333   
## Detection Rate : 0.3333   
## Detection Prevalence : 0.4000   
## Balanced Accuracy : 0.7411   
##   
## 'Positive' Class : 0   
##   
##   
## [[3]]  
## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 13 2  
## 1 3 12  
##   
## Accuracy : 0.8333   
## 95% CI : (0.6528, 0.9436)  
## No Information Rate : 0.5333   
## P-Value [Acc > NIR] : 0.0005955   
##   
## Kappa : 0.6667   
##   
## Mcnemar's Test P-Value : 1.0000000   
##   
## Sensitivity : 0.8125   
## Specificity : 0.8571   
## Pos Pred Value : 0.8667   
## Neg Pred Value : 0.8000   
## Prevalence : 0.5333   
## Detection Rate : 0.4333   
## Detection Prevalence : 0.5000   
## Balanced Accuracy : 0.8348   
##   
## 'Positive' Class : 0   
##   
##   
## [[4]]  
## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 9 1  
## 1 2 18  
##   
## Accuracy : 0.9   
## 95% CI : (0.7347, 0.9789)  
## No Information Rate : 0.6333   
## P-Value [Acc > NIR] : 0.001066   
##   
## Kappa : 0.7805   
##   
## Mcnemar's Test P-Value : 1.000000   
##   
## Sensitivity : 0.8182   
## Specificity : 0.9474   
## Pos Pred Value : 0.9000   
## Neg Pred Value : 0.9000   
## Prevalence : 0.3667   
## Detection Rate : 0.3000   
## Detection Prevalence : 0.3333   
## Balanced Accuracy : 0.8828   
##   
## 'Positive' Class : 0   
##   
##   
## [[5]]  
## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 9 7  
## 1 1 14  
##   
## Accuracy : 0.7419   
## 95% CI : (0.5539, 0.8814)  
## No Information Rate : 0.6774   
## P-Value [Acc > NIR] : 0.2879   
##   
## Kappa : 0.4897   
##   
## Mcnemar's Test P-Value : 0.0771   
##   
## Sensitivity : 0.9000   
## Specificity : 0.6667   
## Pos Pred Value : 0.5625   
## Neg Pred Value : 0.9333   
## Prevalence : 0.3226   
## Detection Rate : 0.2903   
## Detection Prevalence : 0.5161   
## Balanced Accuracy : 0.7833   
##   
## 'Positive' Class : 0   
##   
##   
## [[6]]  
## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 11 3  
## 1 3 13  
##   
## Accuracy : 0.8   
## 95% CI : (0.6143, 0.9229)  
## No Information Rate : 0.5333   
## P-Value [Acc > NIR] : 0.002316   
##   
## Kappa : 0.5982   
##   
## Mcnemar's Test P-Value : 1.000000   
##   
## Sensitivity : 0.7857   
## Specificity : 0.8125   
## Pos Pred Value : 0.7857   
## Neg Pred Value : 0.8125   
## Prevalence : 0.4667   
## Detection Rate : 0.3667   
## Detection Prevalence : 0.4667   
## Balanced Accuracy : 0.7991   
##   
## 'Positive' Class : 0   
##   
##   
## [[7]]  
## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 12 0  
## 1 3 15  
##   
## Accuracy : 0.9   
## 95% CI : (0.7347, 0.9789)  
## No Information Rate : 0.5   
## P-Value [Acc > NIR] : 4.215e-06   
##   
## Kappa : 0.8   
##   
## Mcnemar's Test P-Value : 0.2482   
##   
## Sensitivity : 0.8000   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 0.8333   
## Prevalence : 0.5000   
## Detection Rate : 0.4000   
## Detection Prevalence : 0.4000   
## Balanced Accuracy : 0.9000   
##   
## 'Positive' Class : 0   
##   
##   
## [[8]]  
## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 12 4  
## 1 4 10  
##   
## Accuracy : 0.7333   
## 95% CI : (0.5411, 0.8772)  
## No Information Rate : 0.5333   
## P-Value [Acc > NIR] : 0.02046   
##   
## Kappa : 0.4643   
##   
## Mcnemar's Test P-Value : 1.00000   
##   
## Sensitivity : 0.7500   
## Specificity : 0.7143   
## Pos Pred Value : 0.7500   
## Neg Pred Value : 0.7143   
## Prevalence : 0.5333   
## Detection Rate : 0.4000   
## Detection Prevalence : 0.5333   
## Balanced Accuracy : 0.7321   
##   
## 'Positive' Class : 0   
##   
##   
## [[9]]  
## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 8 3  
## 1 1 18  
##   
## Accuracy : 0.8667   
## 95% CI : (0.6928, 0.9624)  
## No Information Rate : 0.7   
## P-Value [Acc > NIR] : 0.03015   
##   
## Kappa : 0.7015   
##   
## Mcnemar's Test P-Value : 0.61708   
##   
## Sensitivity : 0.8889   
## Specificity : 0.8571   
## Pos Pred Value : 0.7273   
## Neg Pred Value : 0.9474   
## Prevalence : 0.3000   
## Detection Rate : 0.2667   
## Detection Prevalence : 0.3667   
## Balanced Accuracy : 0.8730   
##   
## 'Positive' Class : 0   
##   
##   
## [[10]]  
## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 14 6  
## 1 1 10  
##   
## Accuracy : 0.7742   
## 95% CI : (0.589, 0.9041)  
## No Information Rate : 0.5161   
## P-Value [Acc > NIR] : 0.002897   
##   
## Kappa : 0.5526   
##   
## Mcnemar's Test P-Value : 0.130570   
##   
## Sensitivity : 0.9333   
## Specificity : 0.6250   
## Pos Pred Value : 0.7000   
## Neg Pred Value : 0.9091   
## Prevalence : 0.4839   
## Detection Rate : 0.4516   
## Detection Prevalence : 0.6452   
## Balanced Accuracy : 0.7792   
##   
## 'Positive' Class : 0   
##

### The Mean of NaiveBayes with Cross Validation folds = 5

rst<-do.call(rbind.data.frame, base\_metric\_nb\_table\_cv\_10)  
base\_metric\_nb\_table\_cv\_10\_mean<-data.frame(cbind(Algo='NB\_CV\_10',AUC=mean(rst$ACCURACY),ACCURACY=mean(rst$ACCURACY),TPR=mean(rst$TPR),FPR=mean(rst$FPR),TNR=mean(rst$TNR),FNR=mean(rst$FNR)))  
base\_metric\_nb\_table\_cv\_10\_mean

## Algo AUC ACCURACY TPR  
## 1 NB\_CV\_10 0.812150537634409 0.812150537634409 0.434613340765515  
## FPR TNR FNR  
## 1 0.18445688083846 0.81554311916154 0.565386659234485

## Logistic Regression

start\_tm <- proc.time()  
lr\_model <- glm(target ~ ., data=data\_train,family = "binomial")  
object.size(lr\_model)

## 399824 bytes

lr\_testpred = predict(lr\_model, newdata=data\_test,type="response")  
lr\_pred <- prediction(as.numeric(lr\_testpred > 0.5),data\_test$target)  
lr\_testclass <- lr\_pred@predictions[[1]]  
lr\_table<-table(data\_test$target, lr\_testclass)  
base\_metric\_lr<-caret::confusionMatrix(lr\_table)  
base\_metric\_lr

## Confusion Matrix and Statistics  
##   
## lr\_testclass  
## 0 1  
## 0 22 9  
## 1 2 27  
##   
## Accuracy : 0.8167   
## 95% CI : (0.6956, 0.9048)  
## No Information Rate : 0.6   
## P-Value [Acc > NIR] : 0.0002826   
##   
## Kappa : 0.6358   
##   
## Mcnemar's Test P-Value : 0.0704404   
##   
## Sensitivity : 0.9167   
## Specificity : 0.7500   
## Pos Pred Value : 0.7097   
## Neg Pred Value : 0.9310   
## Prevalence : 0.4000   
## Detection Rate : 0.3667   
## Detection Prevalence : 0.5167   
## Balanced Accuracy : 0.8333   
##   
## 'Positive' Class : 0   
##

end\_tm<-proc.time()   
print(paste("time taken to run Logistic Regression Model",(end\_tm-start\_tm),sep=":"))

## [1] "time taken to run Logistic Regression Model:0"   
## [2] "time taken to run Logistic Regression Model:0.02"   
## [3] "time taken to run Logistic Regression Model:0.0199999999999996"  
## [4] "time taken to run Logistic Regression Model:NA"   
## [5] "time taken to run Logistic Regression Model:NA"

### Estimate Logistic Regression model test data () performance

base\_metric\_lr\_table\_standalone<-estimate\_model\_performance(data\_test$target,lr\_testclass,'LR')  
base\_metric\_lr\_table\_standalone

## Algo AUC ACCURACY TPR FPR TNR FNR  
## 1 LR 0.820356 0.8166667 0.4489796 0.25 0.75 0.5510204

### Logistic Regression with Cross Validation folds = 5

set.seed(43)  
start\_tm <- proc.time()  
df <- data[sample(nrow(data)),]  
folds <- cut(seq(1,nrow(data)),breaks=5,labels=FALSE)  
lr\_pred <- list()  
lr\_testclass <- list()  
lr\_table <- list()  
base\_metric\_lr <- list()  
base\_metric\_lr\_table\_cv\_5 <- list()  
for(i in 1:5){  
 testIndexes <- which(folds==i,arr.ind=TRUE)  
 testData <- df[testIndexes, ]  
 trainData <- df[-testIndexes, ]  
 lr\_model <- glm(target ~ .,family="binomial",data=trainData)  
 lr\_pred[[i]] <- prediction(as.numeric(predict(lr\_model, newdata=testData,type="response") > 0.5),testData$target)  
 lr\_testclass[[i]] <- lr\_pred[[i]]@predictions[[1]]  
 lr\_table[[i]]<-table(testData$target, lr\_testclass[[i]])  
 base\_metric\_lr[[i]]<-caret::confusionMatrix(lr\_table[[i]])  
 base\_metric\_lr\_table\_cv\_5[[i]]<-estimate\_model\_performance(testData$target,lr\_testclass[[i]],paste('LR fold',i,sep =":" ))  
}  
  
end\_tm<-proc.time()   
  
print(paste("time taken to run Logistic Regression Model with CV with 5 Folds",(end\_tm-start\_tm),sep=":"))

## [1] "time taken to run Logistic Regression Model with CV with 5 Folds:0.1"   
## [2] "time taken to run Logistic Regression Model with CV with 5 Folds:0"   
## [3] "time taken to run Logistic Regression Model with CV with 5 Folds:0.0999999999999996"  
## [4] "time taken to run Logistic Regression Model with CV with 5 Folds:NA"   
## [5] "time taken to run Logistic Regression Model with CV with 5 Folds:NA"

### Base Metric for Logistic Regression with Cross Validation folds = 5

base\_metric\_lr

## [[1]]  
## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 18 6  
## 1 6 31  
##   
## Accuracy : 0.8033   
## 95% CI : (0.6816, 0.894)  
## No Information Rate : 0.6066   
## P-Value [Acc > NIR] : 0.000848   
##   
## Kappa : 0.5878   
##   
## Mcnemar's Test P-Value : 1.000000   
##   
## Sensitivity : 0.7500   
## Specificity : 0.8378   
## Pos Pred Value : 0.7500   
## Neg Pred Value : 0.8378   
## Prevalence : 0.3934   
## Detection Rate : 0.2951   
## Detection Prevalence : 0.3934   
## Balanced Accuracy : 0.7939   
##   
## 'Positive' Class : 0   
##   
##   
## [[2]]  
## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 21 4  
## 1 4 31  
##   
## Accuracy : 0.8667   
## 95% CI : (0.7541, 0.9406)  
## No Information Rate : 0.5833   
## P-Value [Acc > NIR] : 1.964e-06   
##   
## Kappa : 0.7257   
##   
## Mcnemar's Test P-Value : 1   
##   
## Sensitivity : 0.8400   
## Specificity : 0.8857   
## Pos Pred Value : 0.8400   
## Neg Pred Value : 0.8857   
## Prevalence : 0.4167   
## Detection Rate : 0.3500   
## Detection Prevalence : 0.4167   
## Balanced Accuracy : 0.8629   
##   
## 'Positive' Class : 0   
##   
##   
## [[3]]  
## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 19 11  
## 1 1 30  
##   
## Accuracy : 0.8033   
## 95% CI : (0.6816, 0.894)  
## No Information Rate : 0.6721   
## P-Value [Acc > NIR] : 0.017333   
##   
## Kappa : 0.6043   
##   
## Mcnemar's Test P-Value : 0.009375   
##   
## Sensitivity : 0.9500   
## Specificity : 0.7317   
## Pos Pred Value : 0.6333   
## Neg Pred Value : 0.9677   
## Prevalence : 0.3279   
## Detection Rate : 0.3115   
## Detection Prevalence : 0.4918   
## Balanced Accuracy : 0.8409   
##   
## 'Positive' Class : 0   
##   
##   
## [[4]]  
## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 24 4  
## 1 5 27  
##   
## Accuracy : 0.85   
## 95% CI : (0.7343, 0.929)  
## No Information Rate : 0.5167   
## P-Value [Acc > NIR] : 6.136e-08   
##   
## Kappa : 0.6993   
##   
## Mcnemar's Test P-Value : 1   
##   
## Sensitivity : 0.8276   
## Specificity : 0.8710   
## Pos Pred Value : 0.8571   
## Neg Pred Value : 0.8437   
## Prevalence : 0.4833   
## Detection Rate : 0.4000   
## Detection Prevalence : 0.4667   
## Balanced Accuracy : 0.8493   
##   
## 'Positive' Class : 0   
##   
##   
## [[5]]  
## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 22 9  
## 1 2 28  
##   
## Accuracy : 0.8197   
## 95% CI : (0.7002, 0.9064)  
## No Information Rate : 0.6066   
## P-Value [Acc > NIR] : 0.000298   
##   
## Kappa : 0.6406   
##   
## Mcnemar's Test P-Value : 0.070440   
##   
## Sensitivity : 0.9167   
## Specificity : 0.7568   
## Pos Pred Value : 0.7097   
## Neg Pred Value : 0.9333   
## Prevalence : 0.3934   
## Detection Rate : 0.3607   
## Detection Prevalence : 0.5082   
## Balanced Accuracy : 0.8367   
##   
## 'Positive' Class : 0   
##

### The Mean of Logistic Regression with Cross Validation folds = 5

rst<-do.call(rbind.data.frame, base\_metric\_lr\_table\_cv\_5)  
base\_metric\_lr\_table\_cv\_5\_mean<-data.frame(cbind(Algo='LR\_CV\_5',AUC=mean(rst$ACCURACY),ACCURACY=mean(rst$ACCURACY),TPR=mean(rst$TPR),FPR=mean(rst$FPR),TNR=mean(rst$TNR),FNR=mean(rst$FNR)))  
base\_metric\_lr\_table\_cv\_5\_mean

## Algo AUC ACCURACY TPR  
## 1 LR\_CV\_5 0.828579234972678 0.828579234972678 0.41390728599132  
## FPR TNR FNR  
## 1 0.183403212136493 0.816596787863507 0.58609271400868

### Logistic Regression with Cross Validation folds = 10

set.seed(43)  
df <- data[sample(nrow(data)),]  
folds <- cut(seq(1,nrow(data)),breaks=10,labels=FALSE)  
lr\_pred <- list()  
lr\_testclass <- list()  
lr\_table <- list()  
base\_metric\_lr <- list()  
base\_metric\_lr\_table\_cv\_10 <- list()  
for(i in 1:10){  
 testIndexes <- which(folds==i,arr.ind=TRUE)  
 testData <- df[testIndexes, ]  
 trainData <- df[-testIndexes, ]  
 lr\_model <- glm(target ~ .,family="binomial",data=trainData)  
 lr\_pred[[i]] <- prediction(as.numeric(predict(lr\_model, newdata=testData,type="response") > 0.5),testData$target)  
 lr\_testclass[[i]] <- lr\_pred[[i]]@predictions[[1]]  
 lr\_table[[i]]<-table(testData$target, lr\_testclass[[i]])  
 base\_metric\_lr[[i]]<-caret::confusionMatrix(lr\_table[[i]])  
 base\_metric\_lr\_table\_cv\_10[[i]]<-estimate\_model\_performance(testData$target,lr\_testclass[[i]],paste('LR fold',i,sep =":" ))  
   
}

### Base Metric for Logistic Regression with Cross Validation folds = 10

base\_metric\_lr

## [[1]]  
## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 9 3  
## 1 0 19  
##   
## Accuracy : 0.9032   
## 95% CI : (0.7425, 0.9796)  
## No Information Rate : 0.7097   
## P-Value [Acc > NIR] : 0.009641   
##   
## Kappa : 0.7862   
##   
## Mcnemar's Test P-Value : 0.248213   
##   
## Sensitivity : 1.0000   
## Specificity : 0.8636   
## Pos Pred Value : 0.7500   
## Neg Pred Value : 1.0000   
## Prevalence : 0.2903   
## Detection Rate : 0.2903   
## Detection Prevalence : 0.3871   
## Balanced Accuracy : 0.9318   
##   
## 'Positive' Class : 0   
##   
##   
## [[2]]  
## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 8 4  
## 1 6 12  
##   
## Accuracy : 0.6667   
## 95% CI : (0.4719, 0.8271)  
## No Information Rate : 0.5333   
## P-Value [Acc > NIR] : 0.09926   
##   
## Kappa : 0.3243   
##   
## Mcnemar's Test P-Value : 0.75183   
##   
## Sensitivity : 0.5714   
## Specificity : 0.7500   
## Pos Pred Value : 0.6667   
## Neg Pred Value : 0.6667   
## Prevalence : 0.4667   
## Detection Rate : 0.2667   
## Detection Prevalence : 0.4000   
## Balanced Accuracy : 0.6607   
##   
## 'Positive' Class : 0   
##   
##   
## [[3]]  
## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 12 3  
## 1 2 13  
##   
## Accuracy : 0.8333   
## 95% CI : (0.6528, 0.9436)  
## No Information Rate : 0.5333   
## P-Value [Acc > NIR] : 0.0005955   
##   
## Kappa : 0.6667   
##   
## Mcnemar's Test P-Value : 1.0000000   
##   
## Sensitivity : 0.8571   
## Specificity : 0.8125   
## Pos Pred Value : 0.8000   
## Neg Pred Value : 0.8667   
## Prevalence : 0.4667   
## Detection Rate : 0.4000   
## Detection Prevalence : 0.5000   
## Balanced Accuracy : 0.8348   
##   
## 'Positive' Class : 0   
##   
##   
## [[4]]  
## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 8 2  
## 1 2 18  
##   
## Accuracy : 0.8667   
## 95% CI : (0.6928, 0.9624)  
## No Information Rate : 0.6667   
## P-Value [Acc > NIR] : 0.01223   
##   
## Kappa : 0.7   
##   
## Mcnemar's Test P-Value : 1.00000   
##   
## Sensitivity : 0.8000   
## Specificity : 0.9000   
## Pos Pred Value : 0.8000   
## Neg Pred Value : 0.9000   
## Prevalence : 0.3333   
## Detection Rate : 0.2667   
## Detection Prevalence : 0.3333   
## Balanced Accuracy : 0.8500   
##   
## 'Positive' Class : 0   
##   
##   
## [[5]]  
## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 8 8  
## 1 0 15  
##   
## Accuracy : 0.7419   
## 95% CI : (0.5539, 0.8814)  
## No Information Rate : 0.7419   
## P-Value [Acc > NIR] : 0.59359   
##   
## Kappa : 0.4918   
##   
## Mcnemar's Test P-Value : 0.01333   
##   
## Sensitivity : 1.0000   
## Specificity : 0.6522   
## Pos Pred Value : 0.5000   
## Neg Pred Value : 1.0000   
## Prevalence : 0.2581   
## Detection Rate : 0.2581   
## Detection Prevalence : 0.5161   
## Balanced Accuracy : 0.8261   
##   
## 'Positive' Class : 0   
##   
##   
## [[6]]  
## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 11 3  
## 1 0 16  
##   
## Accuracy : 0.9   
## 95% CI : (0.7347, 0.9789)  
## No Information Rate : 0.6333   
## P-Value [Acc > NIR] : 0.001066   
##   
## Kappa : 0.7964   
##   
## Mcnemar's Test P-Value : 0.248213   
##   
## Sensitivity : 1.0000   
## Specificity : 0.8421   
## Pos Pred Value : 0.7857   
## Neg Pred Value : 1.0000   
## Prevalence : 0.3667   
## Detection Rate : 0.3667   
## Detection Prevalence : 0.4667   
## Balanced Accuracy : 0.9211   
##   
## 'Positive' Class : 0   
##   
##   
## [[7]]  
## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 11 1  
## 1 3 15  
##   
## Accuracy : 0.8667   
## 95% CI : (0.6928, 0.9624)  
## No Information Rate : 0.5333   
## P-Value [Acc > NIR] : 0.0001236   
##   
## Kappa : 0.7297   
##   
## Mcnemar's Test P-Value : 0.6170751   
##   
## Sensitivity : 0.7857   
## Specificity : 0.9375   
## Pos Pred Value : 0.9167   
## Neg Pred Value : 0.8333   
## Prevalence : 0.4667   
## Detection Rate : 0.3667   
## Detection Prevalence : 0.4000   
## Balanced Accuracy : 0.8616   
##   
## 'Positive' Class : 0   
##   
##   
## [[8]]  
## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 13 3  
## 1 2 12  
##   
## Accuracy : 0.8333   
## 95% CI : (0.6528, 0.9436)  
## No Information Rate : 0.5   
## P-Value [Acc > NIR] : 0.0001625   
##   
## Kappa : 0.6667   
##   
## Mcnemar's Test P-Value : 1.0000000   
##   
## Sensitivity : 0.8667   
## Specificity : 0.8000   
## Pos Pred Value : 0.8125   
## Neg Pred Value : 0.8571   
## Prevalence : 0.5000   
## Detection Rate : 0.4333   
## Detection Prevalence : 0.5333   
## Balanced Accuracy : 0.8333   
##   
## 'Positive' Class : 0   
##   
##   
## [[9]]  
## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 8 3  
## 1 1 18  
##   
## Accuracy : 0.8667   
## 95% CI : (0.6928, 0.9624)  
## No Information Rate : 0.7   
## P-Value [Acc > NIR] : 0.03015   
##   
## Kappa : 0.7015   
##   
## Mcnemar's Test P-Value : 0.61708   
##   
## Sensitivity : 0.8889   
## Specificity : 0.8571   
## Pos Pred Value : 0.7273   
## Neg Pred Value : 0.9474   
## Prevalence : 0.3000   
## Detection Rate : 0.2667   
## Detection Prevalence : 0.3667   
## Balanced Accuracy : 0.8730   
##   
## 'Positive' Class : 0   
##   
##   
## [[10]]  
## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 14 6  
## 1 2 9  
##   
## Accuracy : 0.7419   
## 95% CI : (0.5539, 0.8814)  
## No Information Rate : 0.5161   
## P-Value [Acc > NIR] : 0.008762   
##   
## Kappa : 0.479   
##   
## Mcnemar's Test P-Value : 0.288844   
##   
## Sensitivity : 0.8750   
## Specificity : 0.6000   
## Pos Pred Value : 0.7000   
## Neg Pred Value : 0.8182   
## Prevalence : 0.5161   
## Detection Rate : 0.4516   
## Detection Prevalence : 0.6452   
## Balanced Accuracy : 0.7375   
##   
## 'Positive' Class : 0   
##

### The Mean of Logistic Regression with Cross Validation folds = 10

rst<-do.call(rbind.data.frame, base\_metric\_lr\_table\_cv\_10)  
base\_metric\_lr\_table\_cv\_10\_mean<-data.frame(cbind(Algo='LR\_CV\_10',AUC=mean(rst$ACCURACY),ACCURACY=mean(rst$ACCURACY),TPR=mean(rst$TPR),FPR=mean(rst$FPR),TNR=mean(rst$TNR),FNR=mean(rst$FNR)))  
base\_metric\_lr\_table\_cv\_10\_mean

## Algo AUC ACCURACY TPR  
## 1 LR\_CV\_10 0.822043010752688 0.822043010752688 0.412381925642795  
## FPR TNR FNR  
## 1 0.198494160301941 0.801505839698059 0.587618074357205

## Compate Metrics

print(paste('NaiveBayes:'))

## [1] "NaiveBayes:"

base\_metric\_nb\_table\_standalone

## Algo AUC ACCURACY TPR FPR TNR FNR  
## 1 NB 0.820356 0.8166667 0.4489796 0.25 0.75 0.5510204

print(paste('NB with cv fold=5:'))

## [1] "NB with cv fold=5:"

base\_metric\_nb\_table\_cv\_5\_mean

## Algo AUC ACCURACY TPR  
## 1 NB\_CV\_5 0.821857923497268 0.821857923497268 0.433998399359744  
## FPR TNR FNR  
## 1 0.170473577349712 0.829526422650288 0.566001600640256

print(paste('NB with cv fold=10:'))

## [1] "NB with cv fold=10:"

base\_metric\_nb\_table\_cv\_10\_mean

## Algo AUC ACCURACY TPR  
## 1 NB\_CV\_10 0.812150537634409 0.812150537634409 0.434613340765515  
## FPR TNR FNR  
## 1 0.18445688083846 0.81554311916154 0.565386659234485

print(paste('Logistic Regression:'))

## [1] "Logistic Regression:"

base\_metric\_lr\_table\_standalone

## Algo AUC ACCURACY TPR FPR TNR FNR  
## 1 LR 0.820356 0.8166667 0.4489796 0.25 0.75 0.5510204

print(paste('LR with cv fold=5:'))

## [1] "LR with cv fold=5:"

base\_metric\_lr\_table\_cv\_5\_mean

## Algo AUC ACCURACY TPR  
## 1 LR\_CV\_5 0.828579234972678 0.828579234972678 0.41390728599132  
## FPR TNR FNR  
## 1 0.183403212136493 0.816596787863507 0.58609271400868

print(paste('LR with cv fold=10:'))

## [1] "LR with cv fold=10:"

base\_metric\_lr\_table\_cv\_10\_mean

## Algo AUC ACCURACY TPR  
## 1 LR\_CV\_10 0.822043010752688 0.822043010752688 0.412381925642795  
## FPR TNR FNR  
## 1 0.198494160301941 0.801505839698059 0.587618074357205

## Bootstrap Methodology - NaiveBayes Model

I’m going to create a function for boostrap purposes first.I’m going to run NaiveBayes model 200 times and store the performance metrics for each data boostrap.

set.seed(43)  
apply\_bootstrap\_data <- function(data, proportion = 0.8, sample\_with\_replacement = TRUE){  
 observation <- round(nrow(data) \* proportion, 0)  
 return(data[sample(nrow(data), observation, replace = sample\_with\_replacement),])  
}

start <- proc.time()  
n\_times <- 200  
for (i in 1:n\_times){  
 sample <- apply\_bootstrap\_data(data\_train)  
 nb\_model <- naiveBayes(sample$target ~ ., data = sample)  
 y\_pred <- predict(nb\_model, data\_test,type='raw') # probability  
 y\_pred\_class<-unlist(apply(round(y\_pred),1,which.max))-1 # class  
 performance <- estimate\_model\_performance(data\_test$target, y\_pred\_class, paste("NB Bootstrap ", i))  
 if(exists("performance\_table\_nb")){  
 performance\_table\_nb <- rbind(performance\_table\_nb, performance)  
 } else {  
 performance\_table\_nb <- performance  
 }  
}  
elapsed\_time <- (proc.time() - start)[[3]]  
elapsed\_time

## [1] 5.85

### NB Boostrap Results Table

performance\_table\_nb

## Algo AUC ACCURACY TPR FPR TNR  
## 1 NB Bootstrap 1 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 2 NB Bootstrap 2 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 3 NB Bootstrap 3 0.7858732 0.7833333 0.4680851 0.2647059 0.7352941  
## 4 NB Bootstrap 4 0.8353726 0.8333333 0.4800000 0.2121212 0.7878788  
## 5 NB Bootstrap 5 0.7697442 0.7666667 0.4565217 0.2857143 0.7142857  
## 6 NB Bootstrap 6 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 7 NB Bootstrap 7 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 8 NB Bootstrap 8 0.8687430 0.8666667 0.4807692 0.1818182 0.8181818  
## 9 NB Bootstrap 9 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 10 NB Bootstrap 10 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 11 NB Bootstrap 11 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 12 NB Bootstrap 12 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 13 NB Bootstrap 13 0.8008899 0.8000000 0.5000000 0.2258065 0.7741935  
## 14 NB Bootstrap 14 0.7697442 0.7666667 0.4565217 0.2857143 0.7142857  
## 15 NB Bootstrap 15 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 16 NB Bootstrap 16 0.8515017 0.8500000 0.4901961 0.1875000 0.8125000  
## 17 NB Bootstrap 17 0.8170189 0.8166667 0.5102041 0.2000000 0.8000000  
## 18 NB Bootstrap 18 0.8008899 0.8000000 0.5000000 0.2258065 0.7741935  
## 19 NB Bootstrap 19 0.8353726 0.8333333 0.4800000 0.2121212 0.7878788  
## 20 NB Bootstrap 20 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 21 NB Bootstrap 21 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 22 NB Bootstrap 22 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 23 NB Bootstrap 23 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 24 NB Bootstrap 24 0.8848721 0.8833333 0.4905660 0.1562500 0.8437500  
## 25 NB Bootstrap 25 0.8181313 0.8166667 0.4897959 0.2187500 0.7812500  
## 26 NB Bootstrap 26 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 27 NB Bootstrap 27 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 28 NB Bootstrap 28 0.8042269 0.8000000 0.4375000 0.2702703 0.7297297  
## 29 NB Bootstrap 29 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 30 NB Bootstrap 30 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 31 NB Bootstrap 31 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 32 NB Bootstrap 32 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 33 NB Bootstrap 33 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 34 NB Bootstrap 34 0.8364850 0.8333333 0.4600000 0.2285714 0.7714286  
## 35 NB Bootstrap 35 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 36 NB Bootstrap 36 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 37 NB Bootstrap 37 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 38 NB Bootstrap 38 0.8387097 0.8333333 0.4200000 0.2564103 0.7435897  
## 39 NB Bootstrap 39 0.8526140 0.8500000 0.4705882 0.2058824 0.7941176  
## 40 NB Bootstrap 40 0.8364850 0.8333333 0.4600000 0.2285714 0.7714286  
## 41 NB Bootstrap 41 0.8181313 0.8166667 0.4897959 0.2187500 0.7812500  
## 42 NB Bootstrap 42 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 43 NB Bootstrap 43 0.7858732 0.7833333 0.4680851 0.2647059 0.7352941  
## 44 NB Bootstrap 44 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 45 NB Bootstrap 45 0.7858732 0.7833333 0.4680851 0.2647059 0.7352941  
## 46 NB Bootstrap 46 0.8181313 0.8166667 0.4897959 0.2187500 0.7812500  
## 47 NB Bootstrap 47 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 48 NB Bootstrap 48 0.8042269 0.8000000 0.4375000 0.2702703 0.7297297  
## 49 NB Bootstrap 49 0.8515017 0.8500000 0.4901961 0.1875000 0.8125000  
## 50 NB Bootstrap 50 0.8214683 0.8166667 0.4285714 0.2631579 0.7368421  
## 51 NB Bootstrap 51 0.8364850 0.8333333 0.4600000 0.2285714 0.7714286  
## 52 NB Bootstrap 52 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 53 NB Bootstrap 53 0.7858732 0.7833333 0.4680851 0.2647059 0.7352941  
## 54 NB Bootstrap 54 0.7847608 0.7833333 0.4893617 0.2500000 0.7500000  
## 55 NB Bootstrap 55 0.8181313 0.8166667 0.4897959 0.2187500 0.7812500  
## 56 NB Bootstrap 56 0.8042269 0.8000000 0.4375000 0.2702703 0.7297297  
## 57 NB Bootstrap 57 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 58 NB Bootstrap 58 0.8687430 0.8666667 0.4807692 0.1818182 0.8181818  
## 59 NB Bootstrap 59 0.8353726 0.8333333 0.4800000 0.2121212 0.7878788  
## 60 NB Bootstrap 60 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 61 NB Bootstrap 61 0.8353726 0.8333333 0.4800000 0.2121212 0.7878788  
## 62 NB Bootstrap 62 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 63 NB Bootstrap 63 0.7697442 0.7666667 0.4565217 0.2857143 0.7142857  
## 64 NB Bootstrap 64 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 65 NB Bootstrap 65 0.8364850 0.8333333 0.4600000 0.2285714 0.7714286  
## 66 NB Bootstrap 66 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 67 NB Bootstrap 67 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 68 NB Bootstrap 68 0.7847608 0.7833333 0.4893617 0.2500000 0.7500000  
## 69 NB Bootstrap 69 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 70 NB Bootstrap 70 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 71 NB Bootstrap 71 0.7880979 0.7833333 0.4255319 0.2894737 0.7105263  
## 72 NB Bootstrap 72 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 73 NB Bootstrap 73 0.8042269 0.8000000 0.4375000 0.2702703 0.7297297  
## 74 NB Bootstrap 74 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 75 NB Bootstrap 75 0.7847608 0.7833333 0.4893617 0.2500000 0.7500000  
## 76 NB Bootstrap 76 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 77 NB Bootstrap 77 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 78 NB Bootstrap 78 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 79 NB Bootstrap 79 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 80 NB Bootstrap 80 0.8170189 0.8166667 0.5102041 0.2000000 0.8000000  
## 81 NB Bootstrap 81 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 82 NB Bootstrap 82 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 83 NB Bootstrap 83 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 84 NB Bootstrap 84 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 85 NB Bootstrap 85 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 86 NB Bootstrap 86 0.7880979 0.7833333 0.4255319 0.2894737 0.7105263  
## 87 NB Bootstrap 87 0.8353726 0.8333333 0.4800000 0.2121212 0.7878788  
## 88 NB Bootstrap 88 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 89 NB Bootstrap 89 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 90 NB Bootstrap 90 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 91 NB Bootstrap 91 0.7858732 0.7833333 0.4680851 0.2647059 0.7352941  
## 92 NB Bootstrap 92 0.8364850 0.8333333 0.4600000 0.2285714 0.7714286  
## 93 NB Bootstrap 93 0.7858732 0.7833333 0.4680851 0.2647059 0.7352941  
## 94 NB Bootstrap 94 0.8364850 0.8333333 0.4600000 0.2285714 0.7714286  
## 95 NB Bootstrap 95 0.8008899 0.8000000 0.5000000 0.2258065 0.7741935  
## 96 NB Bootstrap 96 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 97 NB Bootstrap 97 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 98 NB Bootstrap 98 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 99 NB Bootstrap 99 0.8353726 0.8333333 0.4800000 0.2121212 0.7878788  
## 100 NB Bootstrap 100 0.8515017 0.8500000 0.4901961 0.1875000 0.8125000  
## 101 NB Bootstrap 101 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 102 NB Bootstrap 102 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 103 NB Bootstrap 103 0.8364850 0.8333333 0.4600000 0.2285714 0.7714286  
## 104 NB Bootstrap 104 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 105 NB Bootstrap 105 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 106 NB Bootstrap 106 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 107 NB Bootstrap 107 0.8353726 0.8333333 0.4800000 0.2121212 0.7878788  
## 108 NB Bootstrap 108 0.7869855 0.7833333 0.4468085 0.2777778 0.7222222  
## 109 NB Bootstrap 109 0.7858732 0.7833333 0.4680851 0.2647059 0.7352941  
## 110 NB Bootstrap 110 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 111 NB Bootstrap 111 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 112 NB Bootstrap 112 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 113 NB Bootstrap 113 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 114 NB Bootstrap 114 0.8526140 0.8500000 0.4705882 0.2058824 0.7941176  
## 115 NB Bootstrap 115 0.8503893 0.8500000 0.5098039 0.1666667 0.8333333  
## 116 NB Bootstrap 116 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 117 NB Bootstrap 117 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 118 NB Bootstrap 118 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 119 NB Bootstrap 119 0.8526140 0.8500000 0.4705882 0.2058824 0.7941176  
## 120 NB Bootstrap 120 0.8353726 0.8333333 0.4800000 0.2121212 0.7878788  
## 121 NB Bootstrap 121 0.8375973 0.8333333 0.4400000 0.2432432 0.7567568  
## 122 NB Bootstrap 122 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 123 NB Bootstrap 123 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 124 NB Bootstrap 124 0.7858732 0.7833333 0.4680851 0.2647059 0.7352941  
## 125 NB Bootstrap 125 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 126 NB Bootstrap 126 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 127 NB Bootstrap 127 0.8526140 0.8500000 0.4705882 0.2058824 0.7941176  
## 128 NB Bootstrap 128 0.7880979 0.7833333 0.4255319 0.2894737 0.7105263  
## 129 NB Bootstrap 129 0.8170189 0.8166667 0.5102041 0.2000000 0.8000000  
## 130 NB Bootstrap 130 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 131 NB Bootstrap 131 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 132 NB Bootstrap 132 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 133 NB Bootstrap 133 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 134 NB Bootstrap 134 0.8364850 0.8333333 0.4600000 0.2285714 0.7714286  
## 135 NB Bootstrap 135 0.8214683 0.8166667 0.4285714 0.2631579 0.7368421  
## 136 NB Bootstrap 136 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 137 NB Bootstrap 137 0.8170189 0.8166667 0.5102041 0.2000000 0.8000000  
## 138 NB Bootstrap 138 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 139 NB Bootstrap 139 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 140 NB Bootstrap 140 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 141 NB Bootstrap 141 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 142 NB Bootstrap 142 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 143 NB Bootstrap 143 0.8364850 0.8333333 0.4600000 0.2285714 0.7714286  
## 144 NB Bootstrap 144 0.8181313 0.8166667 0.4897959 0.2187500 0.7812500  
## 145 NB Bootstrap 145 0.8364850 0.8333333 0.4600000 0.2285714 0.7714286  
## 146 NB Bootstrap 146 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 147 NB Bootstrap 147 0.7825362 0.7833333 0.5319149 0.2142857 0.7857143  
## 148 NB Bootstrap 148 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 149 NB Bootstrap 149 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 150 NB Bootstrap 150 0.8008899 0.8000000 0.5000000 0.2258065 0.7741935  
## 151 NB Bootstrap 151 0.8042269 0.8000000 0.4375000 0.2702703 0.7297297  
## 152 NB Bootstrap 152 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 153 NB Bootstrap 153 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 154 NB Bootstrap 154 0.8181313 0.8166667 0.4897959 0.2187500 0.7812500  
## 155 NB Bootstrap 155 0.8364850 0.8333333 0.4600000 0.2285714 0.7714286  
## 156 NB Bootstrap 156 0.8181313 0.8166667 0.4897959 0.2187500 0.7812500  
## 157 NB Bootstrap 157 0.7869855 0.7833333 0.4468085 0.2777778 0.7222222  
## 158 NB Bootstrap 158 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 159 NB Bootstrap 159 0.8515017 0.8500000 0.4901961 0.1875000 0.8125000  
## 160 NB Bootstrap 160 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 161 NB Bootstrap 161 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 162 NB Bootstrap 162 0.8515017 0.8500000 0.4901961 0.1875000 0.8125000  
## 163 NB Bootstrap 163 0.8515017 0.8500000 0.4901961 0.1875000 0.8125000  
## 164 NB Bootstrap 164 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 165 NB Bootstrap 165 0.7686318 0.7666667 0.4782609 0.2727273 0.7272727  
## 166 NB Bootstrap 166 0.8364850 0.8333333 0.4600000 0.2285714 0.7714286  
## 167 NB Bootstrap 167 0.8526140 0.8500000 0.4705882 0.2058824 0.7941176  
## 168 NB Bootstrap 168 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 169 NB Bootstrap 169 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 170 NB Bootstrap 170 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 171 NB Bootstrap 171 0.8515017 0.8500000 0.4901961 0.1875000 0.8125000  
## 172 NB Bootstrap 172 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 173 NB Bootstrap 173 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 174 NB Bootstrap 174 0.8364850 0.8333333 0.4600000 0.2285714 0.7714286  
## 175 NB Bootstrap 175 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 176 NB Bootstrap 176 0.8364850 0.8333333 0.4600000 0.2285714 0.7714286  
## 177 NB Bootstrap 177 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 178 NB Bootstrap 178 0.7858732 0.7833333 0.4680851 0.2647059 0.7352941  
## 179 NB Bootstrap 179 0.7997775 0.8000000 0.5208333 0.2068966 0.7931034  
## 180 NB Bootstrap 180 0.8353726 0.8333333 0.4800000 0.2121212 0.7878788  
## 181 NB Bootstrap 181 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 182 NB Bootstrap 182 0.8181313 0.8166667 0.4897959 0.2187500 0.7812500  
## 183 NB Bootstrap 183 0.8342603 0.8333333 0.5000000 0.1935484 0.8064516  
## 184 NB Bootstrap 184 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 185 NB Bootstrap 185 0.8676307 0.8666667 0.5000000 0.1612903 0.8387097  
## 186 NB Bootstrap 186 0.7858732 0.7833333 0.4680851 0.2647059 0.7352941  
## 187 NB Bootstrap 187 0.7847608 0.7833333 0.4893617 0.2500000 0.7500000  
## 188 NB Bootstrap 188 0.8364850 0.8333333 0.4600000 0.2285714 0.7714286  
## 189 NB Bootstrap 189 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 190 NB Bootstrap 190 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 191 NB Bootstrap 191 0.8181313 0.8166667 0.4897959 0.2187500 0.7812500  
## 192 NB Bootstrap 192 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 193 NB Bootstrap 193 0.8364850 0.8333333 0.4600000 0.2285714 0.7714286  
## 194 NB Bootstrap 194 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 195 NB Bootstrap 195 0.7847608 0.7833333 0.4893617 0.2500000 0.7500000  
## 196 NB Bootstrap 196 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 197 NB Bootstrap 197 0.7858732 0.7833333 0.4680851 0.2647059 0.7352941  
## 198 NB Bootstrap 198 0.7858732 0.7833333 0.4680851 0.2647059 0.7352941  
## 199 NB Bootstrap 199 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 200 NB Bootstrap 200 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## FNR  
## 1 0.5306122  
## 2 0.5510204  
## 3 0.5319149  
## 4 0.5200000  
## 5 0.5434783  
## 6 0.5510204  
## 7 0.5510204  
## 8 0.5192308  
## 9 0.5208333  
## 10 0.5416667  
## 11 0.5416667  
## 12 0.5510204  
## 13 0.5000000  
## 14 0.5434783  
## 15 0.5208333  
## 16 0.5098039  
## 17 0.4897959  
## 18 0.5000000  
## 19 0.5200000  
## 20 0.5416667  
## 21 0.5208333  
## 22 0.5416667  
## 23 0.5510204  
## 24 0.5094340  
## 25 0.5102041  
## 26 0.5510204  
## 27 0.5208333  
## 28 0.5625000  
## 29 0.5306122  
## 30 0.5306122  
## 31 0.5510204  
## 32 0.5510204  
## 33 0.5306122  
## 34 0.5400000  
## 35 0.5208333  
## 36 0.5306122  
## 37 0.5416667  
## 38 0.5800000  
## 39 0.5294118  
## 40 0.5400000  
## 41 0.5102041  
## 42 0.5416667  
## 43 0.5319149  
## 44 0.5306122  
## 45 0.5319149  
## 46 0.5102041  
## 47 0.5416667  
## 48 0.5625000  
## 49 0.5098039  
## 50 0.5714286  
## 51 0.5400000  
## 52 0.5510204  
## 53 0.5319149  
## 54 0.5106383  
## 55 0.5102041  
## 56 0.5625000  
## 57 0.5510204  
## 58 0.5192308  
## 59 0.5200000  
## 60 0.5510204  
## 61 0.5200000  
## 62 0.5208333  
## 63 0.5434783  
## 64 0.5208333  
## 65 0.5400000  
## 66 0.5416667  
## 67 0.5208333  
## 68 0.5106383  
## 69 0.5510204  
## 70 0.5416667  
## 71 0.5744681  
## 72 0.5416667  
## 73 0.5625000  
## 74 0.5306122  
## 75 0.5106383  
## 76 0.5416667  
## 77 0.5306122  
## 78 0.5510204  
## 79 0.5416667  
## 80 0.4897959  
## 81 0.5306122  
## 82 0.5208333  
## 83 0.5416667  
## 84 0.5208333  
## 85 0.5306122  
## 86 0.5744681  
## 87 0.5200000  
## 88 0.5510204  
## 89 0.5208333  
## 90 0.5510204  
## 91 0.5319149  
## 92 0.5400000  
## 93 0.5319149  
## 94 0.5400000  
## 95 0.5000000  
## 96 0.5416667  
## 97 0.5510204  
## 98 0.5510204  
## 99 0.5200000  
## 100 0.5098039  
## 101 0.5416667  
## 102 0.5306122  
## 103 0.5400000  
## 104 0.5416667  
## 105 0.5208333  
## 106 0.5306122  
## 107 0.5200000  
## 108 0.5531915  
## 109 0.5319149  
## 110 0.5208333  
## 111 0.5306122  
## 112 0.5208333  
## 113 0.5416667  
## 114 0.5294118  
## 115 0.4901961  
## 116 0.5510204  
## 117 0.5306122  
## 118 0.5510204  
## 119 0.5294118  
## 120 0.5200000  
## 121 0.5600000  
## 122 0.5510204  
## 123 0.5510204  
## 124 0.5319149  
## 125 0.5416667  
## 126 0.5510204  
## 127 0.5294118  
## 128 0.5744681  
## 129 0.4897959  
## 130 0.5306122  
## 131 0.5416667  
## 132 0.5208333  
## 133 0.5306122  
## 134 0.5400000  
## 135 0.5714286  
## 136 0.5306122  
## 137 0.4897959  
## 138 0.5306122  
## 139 0.5416667  
## 140 0.5416667  
## 141 0.5416667  
## 142 0.5208333  
## 143 0.5400000  
## 144 0.5102041  
## 145 0.5400000  
## 146 0.5306122  
## 147 0.4680851  
## 148 0.5510204  
## 149 0.5416667  
## 150 0.5000000  
## 151 0.5625000  
## 152 0.5306122  
## 153 0.5510204  
## 154 0.5102041  
## 155 0.5400000  
## 156 0.5102041  
## 157 0.5531915  
## 158 0.5510204  
## 159 0.5098039  
## 160 0.5306122  
## 161 0.5208333  
## 162 0.5098039  
## 163 0.5098039  
## 164 0.5416667  
## 165 0.5217391  
## 166 0.5400000  
## 167 0.5294118  
## 168 0.5416667  
## 169 0.5306122  
## 170 0.5306122  
## 171 0.5098039  
## 172 0.5208333  
## 173 0.5306122  
## 174 0.5400000  
## 175 0.5208333  
## 176 0.5400000  
## 177 0.5306122  
## 178 0.5319149  
## 179 0.4791667  
## 180 0.5200000  
## 181 0.5416667  
## 182 0.5102041  
## 183 0.5000000  
## 184 0.5306122  
## 185 0.5000000  
## 186 0.5319149  
## 187 0.5106383  
## 188 0.5400000  
## 189 0.5510204  
## 190 0.5416667  
## 191 0.5102041  
## 192 0.5208333  
## 193 0.5400000  
## 194 0.5510204  
## 195 0.5106383  
## 196 0.5416667  
## 197 0.5319149  
## 198 0.5319149  
## 199 0.5416667  
## 200 0.5416667

### The Mean of Boostrap NB model

rst<-performance\_table\_nb  
performance\_table\_nbboostrap\_mean<-data.frame(cbind(Algo='NB\_Bosstrap',AUC=mean(rst$ACCURACY),ACCURACY=mean(rst$ACCURACY),TPR=mean(rst$TPR),FPR=mean(rst$FPR),TNR=mean(rst$TNR),FNR=mean(rst$FNR)))  
performance\_table\_nbboostrap\_mean

## Algo AUC ACCURACY TPR  
## 1 NB\_Bosstrap 0.812416666666667 0.812416666666667 0.467871627039947  
## FPR TNR FNR  
## 1 0.239153754658734 0.760846245341266 0.532128372960053

## Bootstrap Methodology - Logistic Regression Model

I’m going to create a function for boostrap purposes first.I’m going to run Logistic Regression model 200 times and store the performance metrics for each data boostrap.

set.seed(43)  
apply\_bootstrap\_data <- function(data, proportion = 0.8, sample\_with\_replacement = TRUE){  
 observation <- round(nrow(data) \* proportion, 0)  
 return(data[sample(nrow(data), observation, replace = sample\_with\_replacement),])  
}

start <- proc.time()  
n\_times <- 200  
for (i in 1:n\_times){  
 sample <- apply\_bootstrap\_data(data\_train)  
 lr\_model <- glm(target ~ ., data=sample,family = "binomial")  
 lr\_testpred <- predict(lr\_model, data\_test,type='response') # probability  
 lr\_pred <- prediction(as.numeric(lr\_testpred > 0.5,1,0),data\_test$target)  
 y\_pred\_class<-lr\_pred@predictions[[1]] # class  
 performance <- estimate\_model\_performance(data\_test$target, y\_pred\_class, paste("LR Bootstrap", i))  
 if(exists("performance\_table\_lr")){  
 performance\_table\_lr <- rbind(performance\_table\_lr, performance)  
 } else {  
 performance\_table\_lr <- performance  
 }  
}  
elapsed\_time <- (proc.time() - start)[[3]]  
elapsed\_time

## [1] 1.96

### LR Boostrap Results Table

performance\_table\_lr

## Algo AUC ACCURACY TPR FPR TNR  
## 1 LR Bootstrap 1 0.7697442 0.7666667 0.4565217 0.2857143 0.7142857  
## 2 LR Bootstrap 2 0.7858732 0.7833333 0.4680851 0.2647059 0.7352941  
## 3 LR Bootstrap 3 0.8353726 0.8333333 0.4800000 0.2121212 0.7878788  
## 4 LR Bootstrap 4 0.8353726 0.8333333 0.4800000 0.2121212 0.7878788  
## 5 LR Bootstrap 5 0.8364850 0.8333333 0.4600000 0.2285714 0.7714286  
## 6 LR Bootstrap 6 0.8181313 0.8166667 0.4897959 0.2187500 0.7812500  
## 7 LR Bootstrap 7 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 8 LR Bootstrap 8 0.7869855 0.7833333 0.4468085 0.2777778 0.7222222  
## 9 LR Bootstrap 9 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 10 LR Bootstrap 10 0.8526140 0.8500000 0.4705882 0.2058824 0.7941176  
## 11 LR Bootstrap 11 0.8042269 0.8000000 0.4375000 0.2702703 0.7297297  
## 12 LR Bootstrap 12 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 13 LR Bootstrap 13 0.7686318 0.7666667 0.4782609 0.2727273 0.7272727  
## 14 LR Bootstrap 14 0.7686318 0.7666667 0.4782609 0.2727273 0.7272727  
## 15 LR Bootstrap 15 0.8353726 0.8333333 0.4800000 0.2121212 0.7878788  
## 16 LR Bootstrap 16 0.8364850 0.8333333 0.4600000 0.2285714 0.7714286  
## 17 LR Bootstrap 17 0.8848721 0.8833333 0.4905660 0.1562500 0.8437500  
## 18 LR Bootstrap 18 0.7847608 0.7833333 0.4893617 0.2500000 0.7500000  
## 19 LR Bootstrap 19 0.8503893 0.8500000 0.5098039 0.1666667 0.8333333  
## 20 LR Bootstrap 20 0.8353726 0.8333333 0.4800000 0.2121212 0.7878788  
## 21 LR Bootstrap 21 0.7708565 0.7666667 0.4347826 0.2972973 0.7027027  
## 22 LR Bootstrap 22 0.7536151 0.7500000 0.4444444 0.3055556 0.6944444  
## 23 LR Bootstrap 23 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 24 LR Bootstrap 24 0.8859844 0.8833333 0.4716981 0.1764706 0.8235294  
## 25 LR Bootstrap 25 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 26 LR Bootstrap 26 0.8053393 0.8000000 0.4166667 0.2820513 0.7179487  
## 27 LR Bootstrap 27 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 28 LR Bootstrap 28 0.8687430 0.8666667 0.4807692 0.1818182 0.8181818  
## 29 LR Bootstrap 29 0.8353726 0.8333333 0.4800000 0.2121212 0.7878788  
## 30 LR Bootstrap 30 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 31 LR Bootstrap 31 0.8353726 0.8333333 0.4800000 0.2121212 0.7878788  
## 32 LR Bootstrap 32 0.8364850 0.8333333 0.4600000 0.2285714 0.7714286  
## 33 LR Bootstrap 33 0.7352614 0.7333333 0.4772727 0.3030303 0.6969697  
## 34 LR Bootstrap 34 0.8364850 0.8333333 0.4600000 0.2285714 0.7714286  
## 35 LR Bootstrap 35 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 36 LR Bootstrap 36 0.8364850 0.8333333 0.4600000 0.2285714 0.7714286  
## 37 LR Bootstrap 37 0.8364850 0.8333333 0.4600000 0.2285714 0.7714286  
## 38 LR Bootstrap 38 0.8375973 0.8333333 0.4400000 0.2432432 0.7567568  
## 39 LR Bootstrap 39 0.8364850 0.8333333 0.4600000 0.2285714 0.7714286  
## 40 LR Bootstrap 40 0.8353726 0.8333333 0.4800000 0.2121212 0.7878788  
## 41 LR Bootstrap 41 0.7997775 0.8000000 0.5208333 0.2068966 0.7931034  
## 42 LR Bootstrap 42 0.7847608 0.7833333 0.4893617 0.2500000 0.7500000  
## 43 LR Bootstrap 43 0.8526140 0.8500000 0.4705882 0.2058824 0.7941176  
## 44 LR Bootstrap 44 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 45 LR Bootstrap 45 0.7858732 0.7833333 0.4680851 0.2647059 0.7352941  
## 46 LR Bootstrap 46 0.8503893 0.8500000 0.5098039 0.1666667 0.8333333  
## 47 LR Bootstrap 47 0.7869855 0.7833333 0.4468085 0.2777778 0.7222222  
## 48 LR Bootstrap 48 0.7536151 0.7500000 0.4444444 0.3055556 0.6944444  
## 49 LR Bootstrap 49 0.7513904 0.7500000 0.4888889 0.2812500 0.7187500  
## 50 LR Bootstrap 50 0.8042269 0.8000000 0.4375000 0.2702703 0.7297297  
## 51 LR Bootstrap 51 0.8364850 0.8333333 0.4600000 0.2285714 0.7714286  
## 52 LR Bootstrap 52 0.7858732 0.7833333 0.4680851 0.2647059 0.7352941  
## 53 LR Bootstrap 53 0.8526140 0.8500000 0.4705882 0.2058824 0.7941176  
## 54 LR Bootstrap 54 0.7018910 0.7000000 0.4761905 0.3333333 0.6666667  
## 55 LR Bootstrap 55 0.8492770 0.8500000 0.5294118 0.1428571 0.8571429  
## 56 LR Bootstrap 56 0.7847608 0.7833333 0.4893617 0.2500000 0.7500000  
## 57 LR Bootstrap 57 0.7880979 0.7833333 0.4255319 0.2894737 0.7105263  
## 58 LR Bootstrap 58 0.7686318 0.7666667 0.4782609 0.2727273 0.7272727  
## 59 LR Bootstrap 59 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 60 LR Bootstrap 60 0.8364850 0.8333333 0.4600000 0.2285714 0.7714286  
## 61 LR Bootstrap 61 0.7525028 0.7500000 0.4666667 0.2941176 0.7058824  
## 62 LR Bootstrap 62 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 63 LR Bootstrap 63 0.7858732 0.7833333 0.4680851 0.2647059 0.7352941  
## 64 LR Bootstrap 64 0.7513904 0.7500000 0.4888889 0.2812500 0.7187500  
## 65 LR Bootstrap 65 0.8364850 0.8333333 0.4600000 0.2285714 0.7714286  
## 66 LR Bootstrap 66 0.7836485 0.7833333 0.5106383 0.2333333 0.7666667  
## 67 LR Bootstrap 67 0.7697442 0.7666667 0.4565217 0.2857143 0.7142857  
## 68 LR Bootstrap 68 0.7502781 0.7500000 0.5111111 0.2666667 0.7333333  
## 69 LR Bootstrap 69 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 70 LR Bootstrap 70 0.7858732 0.7833333 0.4680851 0.2647059 0.7352941  
## 71 LR Bootstrap 71 0.7880979 0.7833333 0.4255319 0.2894737 0.7105263  
## 72 LR Bootstrap 72 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 73 LR Bootstrap 73 0.7869855 0.7833333 0.4468085 0.2777778 0.7222222  
## 74 LR Bootstrap 74 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 75 LR Bootstrap 75 0.7180200 0.7166667 0.4883721 0.3125000 0.6875000  
## 76 LR Bootstrap 76 0.7708565 0.7666667 0.4347826 0.2972973 0.7027027  
## 77 LR Bootstrap 77 0.7997775 0.8000000 0.5208333 0.2068966 0.7931034  
## 78 LR Bootstrap 78 0.7869855 0.7833333 0.4468085 0.2777778 0.7222222  
## 79 LR Bootstrap 79 0.8042269 0.8000000 0.4375000 0.2702703 0.7297297  
## 80 LR Bootstrap 80 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 81 LR Bootstrap 81 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 82 LR Bootstrap 82 0.7675195 0.7666667 0.5000000 0.2580645 0.7419355  
## 83 LR Bootstrap 83 0.8342603 0.8333333 0.5000000 0.1935484 0.8064516  
## 84 LR Bootstrap 84 0.8353726 0.8333333 0.4800000 0.2121212 0.7878788  
## 85 LR Bootstrap 85 0.7847608 0.7833333 0.4893617 0.2500000 0.7500000  
## 86 LR Bootstrap 86 0.7363737 0.7333333 0.4545455 0.3142857 0.6857143  
## 87 LR Bootstrap 87 0.8008899 0.8000000 0.5000000 0.2258065 0.7741935  
## 88 LR Bootstrap 88 0.7880979 0.7833333 0.4255319 0.2894737 0.7105263  
## 89 LR Bootstrap 89 0.7858732 0.7833333 0.4680851 0.2647059 0.7352941  
## 90 LR Bootstrap 90 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 91 LR Bootstrap 91 0.8364850 0.8333333 0.4600000 0.2285714 0.7714286  
## 92 LR Bootstrap 92 0.8353726 0.8333333 0.4800000 0.2121212 0.7878788  
## 93 LR Bootstrap 93 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 94 LR Bootstrap 94 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 95 LR Bootstrap 95 0.7675195 0.7666667 0.5000000 0.2580645 0.7419355  
## 96 LR Bootstrap 96 0.8676307 0.8666667 0.5000000 0.1612903 0.8387097  
## 97 LR Bootstrap 97 0.7525028 0.7500000 0.4666667 0.2941176 0.7058824  
## 98 LR Bootstrap 98 0.7213571 0.7166667 0.4186047 0.3421053 0.6578947  
## 99 LR Bootstrap 99 0.8364850 0.8333333 0.4600000 0.2285714 0.7714286  
## 100 LR Bootstrap 100 0.8181313 0.8166667 0.4897959 0.2187500 0.7812500  
## 101 LR Bootstrap 101 0.7847608 0.7833333 0.4893617 0.2500000 0.7500000  
## 102 LR Bootstrap 102 0.7513904 0.7500000 0.4888889 0.2812500 0.7187500  
## 103 LR Bootstrap 103 0.8515017 0.8500000 0.4901961 0.1875000 0.8125000  
## 104 LR Bootstrap 104 0.8042269 0.8000000 0.4375000 0.2702703 0.7297297  
## 105 LR Bootstrap 105 0.7836485 0.7833333 0.5106383 0.2333333 0.7666667  
## 106 LR Bootstrap 106 0.8687430 0.8666667 0.4807692 0.1818182 0.8181818  
## 107 LR Bootstrap 107 0.8364850 0.8333333 0.4600000 0.2285714 0.7714286  
## 108 LR Bootstrap 108 0.8676307 0.8666667 0.5000000 0.1612903 0.8387097  
## 109 LR Bootstrap 109 0.7836485 0.7833333 0.5106383 0.2333333 0.7666667  
## 110 LR Bootstrap 110 0.7836485 0.7833333 0.5106383 0.2333333 0.7666667  
## 111 LR Bootstrap 111 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 112 LR Bootstrap 112 0.8859844 0.8833333 0.4716981 0.1764706 0.8235294  
## 113 LR Bootstrap 113 0.8364850 0.8333333 0.4600000 0.2285714 0.7714286  
## 114 LR Bootstrap 114 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 115 LR Bootstrap 115 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 116 LR Bootstrap 116 0.7869855 0.7833333 0.4468085 0.2777778 0.7222222  
## 117 LR Bootstrap 117 0.8342603 0.8333333 0.5000000 0.1935484 0.8064516  
## 118 LR Bootstrap 118 0.8214683 0.8166667 0.4285714 0.2631579 0.7368421  
## 119 LR Bootstrap 119 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 120 LR Bootstrap 120 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 121 LR Bootstrap 121 0.7547275 0.7500000 0.4222222 0.3157895 0.6842105  
## 122 LR Bootstrap 122 0.7869855 0.7833333 0.4468085 0.2777778 0.7222222  
## 123 LR Bootstrap 123 0.8008899 0.8000000 0.5000000 0.2258065 0.7741935  
## 124 LR Bootstrap 124 0.7547275 0.7500000 0.4222222 0.3157895 0.6842105  
## 125 LR Bootstrap 125 0.7525028 0.7500000 0.4666667 0.2941176 0.7058824  
## 126 LR Bootstrap 126 0.7869855 0.7833333 0.4468085 0.2777778 0.7222222  
## 127 LR Bootstrap 127 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 128 LR Bootstrap 128 0.7708565 0.7666667 0.4347826 0.2972973 0.7027027  
## 129 LR Bootstrap 129 0.8008899 0.8000000 0.5000000 0.2258065 0.7741935  
## 130 LR Bootstrap 130 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 131 LR Bootstrap 131 0.7697442 0.7666667 0.4565217 0.2857143 0.7142857  
## 132 LR Bootstrap 132 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 133 LR Bootstrap 133 0.7858732 0.7833333 0.4680851 0.2647059 0.7352941  
## 134 LR Bootstrap 134 0.7847608 0.7833333 0.4893617 0.2500000 0.7500000  
## 135 LR Bootstrap 135 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 136 LR Bootstrap 136 0.7869855 0.7833333 0.4468085 0.2777778 0.7222222  
## 137 LR Bootstrap 137 0.7997775 0.8000000 0.5208333 0.2068966 0.7931034  
## 138 LR Bootstrap 138 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 139 LR Bootstrap 139 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 140 LR Bootstrap 140 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 141 LR Bootstrap 141 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 142 LR Bootstrap 142 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 143 LR Bootstrap 143 0.8515017 0.8500000 0.4901961 0.1875000 0.8125000  
## 144 LR Bootstrap 144 0.8848721 0.8833333 0.4905660 0.1562500 0.8437500  
## 145 LR Bootstrap 145 0.7697442 0.7666667 0.4565217 0.2857143 0.7142857  
## 146 LR Bootstrap 146 0.8042269 0.8000000 0.4375000 0.2702703 0.7297297  
## 147 LR Bootstrap 147 0.7814238 0.7833333 0.5531915 0.1923077 0.8076923  
## 148 LR Bootstrap 148 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 149 LR Bootstrap 149 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 150 LR Bootstrap 150 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 151 LR Bootstrap 151 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 152 LR Bootstrap 152 0.7858732 0.7833333 0.4680851 0.2647059 0.7352941  
## 153 LR Bootstrap 153 0.8526140 0.8500000 0.4705882 0.2058824 0.7941176  
## 154 LR Bootstrap 154 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 155 LR Bootstrap 155 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 156 LR Bootstrap 156 0.8526140 0.8500000 0.4705882 0.2058824 0.7941176  
## 157 LR Bootstrap 157 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 158 LR Bootstrap 158 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 159 LR Bootstrap 159 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 160 LR Bootstrap 160 0.8526140 0.8500000 0.4705882 0.2058824 0.7941176  
## 161 LR Bootstrap 161 0.7513904 0.7500000 0.4888889 0.2812500 0.7187500  
## 162 LR Bootstrap 162 0.7675195 0.7666667 0.5000000 0.2580645 0.7419355  
## 163 LR Bootstrap 163 0.8676307 0.8666667 0.5000000 0.1612903 0.8387097  
## 164 LR Bootstrap 164 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 165 LR Bootstrap 165 0.8203560 0.8166667 0.4489796 0.2500000 0.7500000  
## 166 LR Bootstrap 166 0.7880979 0.7833333 0.4255319 0.2894737 0.7105263  
## 167 LR Bootstrap 167 0.8526140 0.8500000 0.4705882 0.2058824 0.7941176  
## 168 LR Bootstrap 168 0.8181313 0.8166667 0.4897959 0.2187500 0.7812500  
## 169 LR Bootstrap 169 0.8008899 0.8000000 0.5000000 0.2258065 0.7741935  
## 170 LR Bootstrap 170 0.8342603 0.8333333 0.5000000 0.1935484 0.8064516  
## 171 LR Bootstrap 171 0.8353726 0.8333333 0.4800000 0.2121212 0.7878788  
## 172 LR Bootstrap 172 0.8515017 0.8500000 0.4901961 0.1875000 0.8125000  
## 173 LR Bootstrap 173 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 174 LR Bootstrap 174 0.8526140 0.8500000 0.4705882 0.2058824 0.7941176  
## 175 LR Bootstrap 175 0.7836485 0.7833333 0.5106383 0.2333333 0.7666667  
## 176 LR Bootstrap 176 0.8848721 0.8833333 0.4905660 0.1562500 0.8437500  
## 177 LR Bootstrap 177 0.7525028 0.7500000 0.4666667 0.2941176 0.7058824  
## 178 LR Bootstrap 178 0.7686318 0.7666667 0.4782609 0.2727273 0.7272727  
## 179 LR Bootstrap 179 0.8170189 0.8166667 0.5102041 0.2000000 0.8000000  
## 180 LR Bootstrap 180 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 181 LR Bootstrap 181 0.8020022 0.8000000 0.4791667 0.2424242 0.7575758  
## 182 LR Bootstrap 182 0.8181313 0.8166667 0.4897959 0.2187500 0.7812500  
## 183 LR Bootstrap 183 0.8481646 0.8500000 0.5490196 0.1153846 0.8846154  
## 184 LR Bootstrap 184 0.8687430 0.8666667 0.4807692 0.1818182 0.8181818  
## 185 LR Bootstrap 185 0.8526140 0.8500000 0.4705882 0.2058824 0.7941176  
## 186 LR Bootstrap 186 0.8008899 0.8000000 0.5000000 0.2258065 0.7741935  
## 187 LR Bootstrap 187 0.7847608 0.7833333 0.4893617 0.2500000 0.7500000  
## 188 LR Bootstrap 188 0.8526140 0.8500000 0.4705882 0.2058824 0.7941176  
## 189 LR Bootstrap 189 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 190 LR Bootstrap 190 0.7708565 0.7666667 0.4347826 0.2972973 0.7027027  
## 191 LR Bootstrap 191 0.7686318 0.7666667 0.4782609 0.2727273 0.7272727  
## 192 LR Bootstrap 192 0.7374861 0.7333333 0.4318182 0.3243243 0.6756757  
## 193 LR Bootstrap 193 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 194 LR Bootstrap 194 0.8031146 0.8000000 0.4583333 0.2571429 0.7428571  
## 195 LR Bootstrap 195 0.8353726 0.8333333 0.4800000 0.2121212 0.7878788  
## 196 LR Bootstrap 196 0.7697442 0.7666667 0.4565217 0.2857143 0.7142857  
## 197 LR Bootstrap 197 0.8042269 0.8000000 0.4375000 0.2702703 0.7297297  
## 198 LR Bootstrap 198 0.8192436 0.8166667 0.4693878 0.2352941 0.7647059  
## 199 LR Bootstrap 199 0.7525028 0.7500000 0.4666667 0.2941176 0.7058824  
## 200 LR Bootstrap 200 0.7525028 0.7500000 0.4666667 0.2941176 0.7058824  
## FNR  
## 1 0.5434783  
## 2 0.5319149  
## 3 0.5200000  
## 4 0.5200000  
## 5 0.5400000  
## 6 0.5102041  
## 7 0.5416667  
## 8 0.5531915  
## 9 0.5510204  
## 10 0.5294118  
## 11 0.5625000  
## 12 0.5416667  
## 13 0.5217391  
## 14 0.5217391  
## 15 0.5200000  
## 16 0.5400000  
## 17 0.5094340  
## 18 0.5106383  
## 19 0.4901961  
## 20 0.5200000  
## 21 0.5652174  
## 22 0.5555556  
## 23 0.5510204  
## 24 0.5283019  
## 25 0.5306122  
## 26 0.5833333  
## 27 0.5208333  
## 28 0.5192308  
## 29 0.5200000  
## 30 0.5306122  
## 31 0.5200000  
## 32 0.5400000  
## 33 0.5227273  
## 34 0.5400000  
## 35 0.5306122  
## 36 0.5400000  
## 37 0.5400000  
## 38 0.5600000  
## 39 0.5400000  
## 40 0.5200000  
## 41 0.4791667  
## 42 0.5106383  
## 43 0.5294118  
## 44 0.5306122  
## 45 0.5319149  
## 46 0.4901961  
## 47 0.5531915  
## 48 0.5555556  
## 49 0.5111111  
## 50 0.5625000  
## 51 0.5400000  
## 52 0.5319149  
## 53 0.5294118  
## 54 0.5238095  
## 55 0.4705882  
## 56 0.5106383  
## 57 0.5744681  
## 58 0.5217391  
## 59 0.5416667  
## 60 0.5400000  
## 61 0.5333333  
## 62 0.5416667  
## 63 0.5319149  
## 64 0.5111111  
## 65 0.5400000  
## 66 0.4893617  
## 67 0.5434783  
## 68 0.4888889  
## 69 0.5510204  
## 70 0.5319149  
## 71 0.5744681  
## 72 0.5208333  
## 73 0.5531915  
## 74 0.5416667  
## 75 0.5116279  
## 76 0.5652174  
## 77 0.4791667  
## 78 0.5531915  
## 79 0.5625000  
## 80 0.5208333  
## 81 0.5208333  
## 82 0.5000000  
## 83 0.5000000  
## 84 0.5200000  
## 85 0.5106383  
## 86 0.5454545  
## 87 0.5000000  
## 88 0.5744681  
## 89 0.5319149  
## 90 0.5416667  
## 91 0.5400000  
## 92 0.5200000  
## 93 0.5416667  
## 94 0.5208333  
## 95 0.5000000  
## 96 0.5000000  
## 97 0.5333333  
## 98 0.5813953  
## 99 0.5400000  
## 100 0.5102041  
## 101 0.5106383  
## 102 0.5111111  
## 103 0.5098039  
## 104 0.5625000  
## 105 0.4893617  
## 106 0.5192308  
## 107 0.5400000  
## 108 0.5000000  
## 109 0.4893617  
## 110 0.4893617  
## 111 0.5306122  
## 112 0.5283019  
## 113 0.5400000  
## 114 0.5416667  
## 115 0.5510204  
## 116 0.5531915  
## 117 0.5000000  
## 118 0.5714286  
## 119 0.5510204  
## 120 0.5306122  
## 121 0.5777778  
## 122 0.5531915  
## 123 0.5000000  
## 124 0.5777778  
## 125 0.5333333  
## 126 0.5531915  
## 127 0.5208333  
## 128 0.5652174  
## 129 0.5000000  
## 130 0.5306122  
## 131 0.5434783  
## 132 0.5416667  
## 133 0.5319149  
## 134 0.5106383  
## 135 0.5416667  
## 136 0.5531915  
## 137 0.4791667  
## 138 0.5306122  
## 139 0.5510204  
## 140 0.5208333  
## 141 0.5208333  
## 142 0.5306122  
## 143 0.5098039  
## 144 0.5094340  
## 145 0.5434783  
## 146 0.5625000  
## 147 0.4468085  
## 148 0.5510204  
## 149 0.5208333  
## 150 0.5416667  
## 151 0.5416667  
## 152 0.5319149  
## 153 0.5294118  
## 154 0.5306122  
## 155 0.5416667  
## 156 0.5294118  
## 157 0.5306122  
## 158 0.5416667  
## 159 0.5416667  
## 160 0.5294118  
## 161 0.5111111  
## 162 0.5000000  
## 163 0.5000000  
## 164 0.5208333  
## 165 0.5510204  
## 166 0.5744681  
## 167 0.5294118  
## 168 0.5102041  
## 169 0.5000000  
## 170 0.5000000  
## 171 0.5200000  
## 172 0.5098039  
## 173 0.5208333  
## 174 0.5294118  
## 175 0.4893617  
## 176 0.5094340  
## 177 0.5333333  
## 178 0.5217391  
## 179 0.4897959  
## 180 0.5306122  
## 181 0.5208333  
## 182 0.5102041  
## 183 0.4509804  
## 184 0.5192308  
## 185 0.5294118  
## 186 0.5000000  
## 187 0.5106383  
## 188 0.5294118  
## 189 0.5306122  
## 190 0.5652174  
## 191 0.5217391  
## 192 0.5681818  
## 193 0.5306122  
## 194 0.5416667  
## 195 0.5200000  
## 196 0.5434783  
## 197 0.5625000  
## 198 0.5306122  
## 199 0.5333333  
## 200 0.5333333

### The Mean of Boostrap LR model

rst<-performance\_table\_lr  
performance\_table\_lrboostrap\_mean<-data.frame(cbind(Algo='LR\_Boostrap',AUC=mean(rst$ACCURACY),ACCURACY=mean(rst$ACCURACY),TPR=mean(rst$TPR),FPR=mean(rst$FPR),TNR=mean(rst$TNR),FNR=mean(rst$FNR)))  
# Average  
performance\_table\_lrboostrap\_mean

## Algo AUC ACCURACY TPR  
## 1 LR\_Boostrap 0.803916666666667 0.803916666666667 0.471115313161927  
## FPR TNR FNR  
## 1 0.243282243002635 0.756717756997365 0.528884686838073

## Summary Performance Results

#putting results in dataFrame  
data.frame(rbind(base\_metric\_nb\_table\_standalone,base\_metric\_nb\_table\_cv\_5\_mean,base\_metric\_nb\_table\_cv\_10\_mean,performance\_table\_nbboostrap\_mean,base\_metric\_lr\_table\_standalone,base\_metric\_lr\_table\_cv\_5\_mean,base\_metric\_lr\_table\_cv\_10\_mean,performance\_table\_lrboostrap\_mean))

## Algo AUC ACCURACY TPR  
## 1 NB 0.82035595105673 0.816666666666667 0.448979591836735  
## 2 NB\_CV\_5 0.821857923497268 0.821857923497268 0.433998399359744  
## 3 NB\_CV\_10 0.812150537634409 0.812150537634409 0.434613340765515  
## 4 NB\_Bosstrap 0.812416666666667 0.812416666666667 0.467871627039947  
## 5 LR 0.82035595105673 0.816666666666667 0.448979591836735  
## 6 LR\_CV\_5 0.828579234972678 0.828579234972678 0.41390728599132  
## 7 LR\_CV\_10 0.822043010752688 0.822043010752688 0.412381925642795  
## 8 LR\_Boostrap 0.803916666666667 0.803916666666667 0.471115313161927  
## FPR TNR FNR  
## 1 0.25 0.75 0.551020408163265  
## 2 0.170473577349712 0.829526422650288 0.566001600640256  
## 3 0.18445688083846 0.81554311916154 0.565386659234485  
## 4 0.239153754658734 0.760846245341266 0.532128372960053  
## 5 0.25 0.75 0.551020408163265  
## 6 0.183403212136493 0.816596787863507 0.58609271400868  
## 7 0.198494160301941 0.801505839698059 0.587618074357205  
## 8 0.243282243002635 0.756717756997365 0.528884686838073

# PART B

For the same dataset, set seed (43) split 80/20. Using randomForest grow three different forests varuing the number of trees atleast three times. Start with seeding and fresh split for each forest. Note down as best as you can development (engineering) cost as well as computing cost(elapsed time) for each run. And compare these results with the experiment in Part A. Submit a pdf and executable script in python or R.

data$cp <- as.factor(data$cp)  
data$fbs <- as.factor(data$fbs)  
data$exang <- as.factor(data$exang)  
data$slope <- as.factor(data$slope)  
data$ca <- as.factor(data$ca)  
data$sex <- as.factor(data$sex)  
data$restecg <- as.factor(data$restecg)  
data$thal <- as.factor(data$thal)  
data$target <- as.factor(data$target)  
# do a 80/20 split   
set.seed(43)  
split\_df <- sample(seq\_len(nrow(data)), size = floor(0.8 \* nrow(data)))  
train\_heart <- data[ split\_df,]  
test\_heart <- data[-split\_df,]

### Random Forest - 10 Trees

start <- proc.time()  
rf\_10\_trees <- train(form = target ~ .,  
 data = train\_heart,  
 method = 'rf',  
 trControl = trainControl(),  
 ntree = 10)  
rf\_10\_trees

## Random Forest   
##   
## 242 samples  
## 13 predictor  
## 2 classes: '0', '1'   
##   
## No pre-processing  
## Resampling: Bootstrapped (25 reps)   
## Summary of sample sizes: 242, 242, 242, 242, 242, 242, ...   
## Resampling results across tuning parameters:  
##   
## mtry Accuracy Kappa   
## 2 0.7941716 0.5799767  
## 12 0.7682864 0.5301247  
## 22 0.7676479 0.5267493  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 2.

elapsed\_time <- (proc.time() - start)[[3]]  
elapsed\_time

## [1] 1.02

### Random Forest - 10 Trees Performance

pred<-predict(rf\_10\_trees, subset(test\_heart, select = -c(target)))  
rst\_class<-as.factor(pred)  
model\_cm <-confusionMatrix(rst\_class,test\_heart$target)  
rst\_rf\_10<-estimate\_model\_performance(rst\_class,test\_heart$target,'Random Forest - 10 Trees')  
rst\_rf\_10

## Algo AUC ACCURACY TPR FPR TNR FNR  
## 1 Random Forest - 10 Trees 0.8225108 0.8196721 0.48 0.1333333 0.8666667 0.52

### Random Forest - 30 Trees

start <- proc.time()  
rf\_30\_trees <- train(form = target ~ .,  
 data = train\_heart,  
 method = 'rf',  
 trControl = trainControl(),  
 ntree = 30)  
rf\_30\_trees

## Random Forest   
##   
## 242 samples  
## 13 predictor  
## 2 classes: '0', '1'   
##   
## No pre-processing  
## Resampling: Bootstrapped (25 reps)   
## Summary of sample sizes: 242, 242, 242, 242, 242, 242, ...   
## Resampling results across tuning parameters:  
##   
## mtry Accuracy Kappa   
## 2 0.7915910 0.5770511  
## 12 0.7703821 0.5349698  
## 22 0.7588745 0.5124723  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 2.

elapsed\_time <- (proc.time() - start)[[3]]  
elapsed\_time

## [1] 1.38

### Random Forest - 30 Trees Performance

pred<-predict(rf\_30\_trees, subset(test\_heart, select = -c(target)))  
rst\_class<-as.factor(pred)  
model\_cm <-confusionMatrix(rst\_class,test\_heart$target)  
rst\_rf\_30<-estimate\_model\_performance(rst\_class,test\_heart$target,'Random Forest - 30 Trees')  
rst\_rf\_30

## Algo AUC ACCURACY TPR FPR TNR  
## 1 Random Forest - 30 Trees 0.7895022 0.7868852 0.4791667 0.1666667 0.8333333  
## FNR  
## 1 0.5208333

### Random Forest - 90 Trees

start <- proc.time()  
rf\_90\_trees <- train(form = target ~ .,  
 data = train\_heart,  
 method = 'rf',  
 trControl = trainControl(),  
 ntree = 90)  
rf\_90\_trees

## Random Forest   
##   
## 242 samples  
## 13 predictor  
## 2 classes: '0', '1'   
##   
## No pre-processing  
## Resampling: Bootstrapped (25 reps)   
## Summary of sample sizes: 242, 242, 242, 242, 242, 242, ...   
## Resampling results across tuning parameters:  
##   
## mtry Accuracy Kappa   
## 2 0.8041547 0.5988027  
## 12 0.7751630 0.5405130  
## 22 0.7771215 0.5449373  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 2.

elapsed\_time <- (proc.time() - start)[[3]]  
elapsed\_time

## [1] 2.76

### Random Forest - 90 Trees Performance

pred<-predict(rf\_90\_trees, subset(test\_heart, select = -c(target)))  
rst\_class<-as.factor(pred)  
model\_cm <-confusionMatrix(rst\_class,test\_heart$target)  
rst\_rf\_90<-estimate\_model\_performance(rst\_class,test\_heart$target,'Random Forest - 90 Trees')  
rst\_rf\_90

## Algo AUC ACCURACY TPR FPR TNR  
## 1 Random Forest - 90 Trees 0.8598398 0.8360656 0.4313725 0.03333333 0.9666667  
## FNR  
## 1 0.5686275

### Combine Random Forest Results

data.frame(rbind(rst\_rf\_10, rst\_rf\_30, rst\_rf\_90))

## Algo AUC ACCURACY TPR FPR TNR  
## 1 Random Forest - 10 Trees 0.8225108 0.8196721 0.4800000 0.13333333 0.8666667  
## 2 Random Forest - 30 Trees 0.7895022 0.7868852 0.4791667 0.16666667 0.8333333  
## 3 Random Forest - 90 Trees 0.8598398 0.8360656 0.4313725 0.03333333 0.9666667  
## FNR  
## 1 0.5200000  
## 2 0.5208333  
## 3 0.5686275

## Part C

Include a summary of your findings. Which of the two methods bootstrap vs cv do you recommend to your customer? And why? Be elaborate. Including computing costs, engineering costs and model performance. Did you incorporate Pareto’s maxim or the Razor and how did these two heuristics influence your decision?

**Answer:** I would use cross validation methodlogies over bootstrapp methods, I can see that it was less computationally expensive and cross-validation resulted in better Accuracy than boostrapping methods. All four Logistic models created high accuracy, and AUC. However we dont see huge differences in Accuracy results between CV=5 and CV=10.The Logistic Regression with 10-fold CV model does not add much accuracy or stability to 5-fold CV model.The Occam’s razor suggests that the simpler model (the 5-fold CV) should be used for Logistic Regression Models. NaiveBayes models also performed well compare to average results.We do see decrease in accuracy changing cross-validation from 5-folds to 10 folds.The Occam’s razor suggests that the simpler model (the 5-fold CV) should be used since there is no additional increase in accuracy. Random Forest model, increasing ntrees from 30 to 90 actually incread the accuracy. I would use randomforest with 90 trees.

final\_results <- data.frame(rbind(base\_metric\_nb\_table\_standalone,base\_metric\_nb\_table\_cv\_5\_mean,base\_metric\_nb\_table\_cv\_10\_mean,performance\_table\_nbboostrap\_mean,base\_metric\_lr\_table\_standalone,base\_metric\_lr\_table\_cv\_5\_mean,base\_metric\_lr\_table\_cv\_10\_mean,performance\_table\_lrboostrap\_mean,rst\_rf\_10,rst\_rf\_30,rst\_rf\_90))  
final\_results

## Algo AUC ACCURACY  
## 1 NB 0.82035595105673 0.816666666666667  
## 2 NB\_CV\_5 0.821857923497268 0.821857923497268  
## 3 NB\_CV\_10 0.812150537634409 0.812150537634409  
## 4 NB\_Bosstrap 0.812416666666667 0.812416666666667  
## 5 LR 0.82035595105673 0.816666666666667  
## 6 LR\_CV\_5 0.828579234972678 0.828579234972678  
## 7 LR\_CV\_10 0.822043010752688 0.822043010752688  
## 8 LR\_Boostrap 0.803916666666667 0.803916666666667  
## 9 Random Forest - 10 Trees 0.822510822510823 0.819672131147541  
## 10 Random Forest - 30 Trees 0.789502164502164 0.786885245901639  
## 11 Random Forest - 90 Trees 0.859839816933638 0.836065573770492  
## TPR FPR TNR FNR  
## 1 0.448979591836735 0.25 0.75 0.551020408163265  
## 2 0.433998399359744 0.170473577349712 0.829526422650288 0.566001600640256  
## 3 0.434613340765515 0.18445688083846 0.81554311916154 0.565386659234485  
## 4 0.467871627039947 0.239153754658734 0.760846245341266 0.532128372960053  
## 5 0.448979591836735 0.25 0.75 0.551020408163265  
## 6 0.41390728599132 0.183403212136493 0.816596787863507 0.58609271400868  
## 7 0.412381925642795 0.198494160301941 0.801505839698059 0.587618074357205  
## 8 0.471115313161927 0.243282243002635 0.756717756997365 0.528884686838073  
## 9 0.48 0.133333333333333 0.866666666666667 0.52  
## 10 0.479166666666667 0.166666666666667 0.833333333333333 0.520833333333333  
## 11 0.431372549019608 0.0333333333333333 0.966666666666667 0.568627450980392