# HW_6.3.R

arnab

2023-02-23

```
library(readr)
library(readsparse)
library(readr)
library(dplyr)
library(stargazer)
library(caret)
library(pROC)
library(ggplot2)
library(gridExtra)
library(tensor)

library(adabag)
library(caret)

packages = c("parallel","doParallel","doSNOW")
invisible(xfun::pkg_attach(packages))
train_X<- read_table("D:/OneDrive - Florida State University/MyFSU_OneDrive/FSU Course Work/5635/Dataset
                     col_names = FALSE)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double(),
##   X501 = col_logical()
## )
## i Use `spec()` for the full column specifications.
```

```
train_Y<- read_table("D:/OneDrive - Florida State University/MyFSU_OneDrive/FSU Course Work/5635/Dataset
                     col_names = FALSE)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   X1 = col_double()
## )
```

```
test_X <- read_table("D:/OneDrive - Florida State University/MyFSU_OneDrive/FSU Course Work/5635/Dataset
                     col_names = FALSE)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double(),
##   X501 = col_logical()
```

```
## )
## i Use `spec()` for the full column specifications.
```

```
test_Y <- read_table("D:/OneDrive - Florida State University/MyFSU_OneDrive/FSU Course Work/5635/Dataset
                     col_names = FALSE)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   X1 = col_double()
## )
```

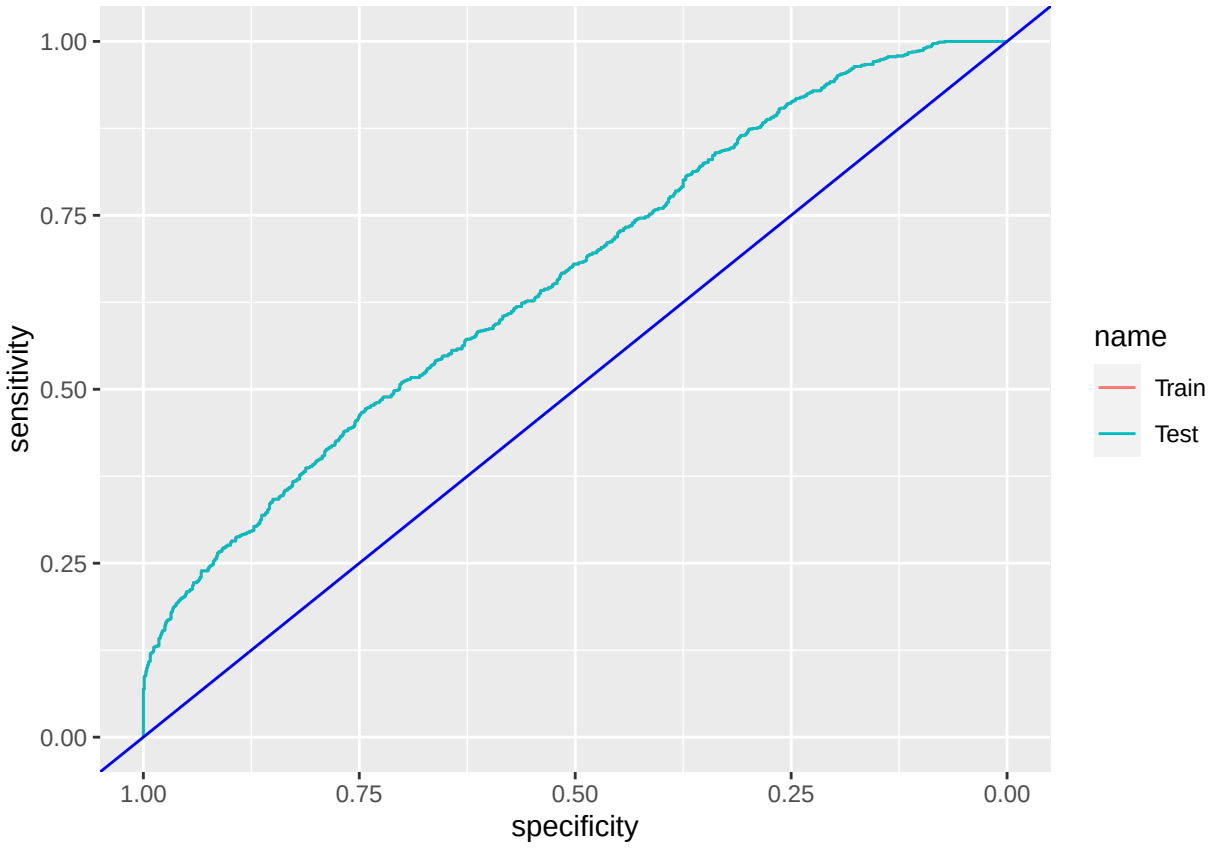
```
train_X=train_X[,-501]
test_X=test_X[,-501]

x_mean=as.numeric(colMeans(train_X))
x_sd =as.numeric(apply(train_X,2,sd))

X = rbind(scale(train_X,center=x_mean,scale=x_sd),
          scale(test_X,center=x_mean,scale=x_sd))
X = X[, colSums(is.na(X)) == 0]
length(which(is.na(X) == TRUE))
```

```
## [1] 0
```

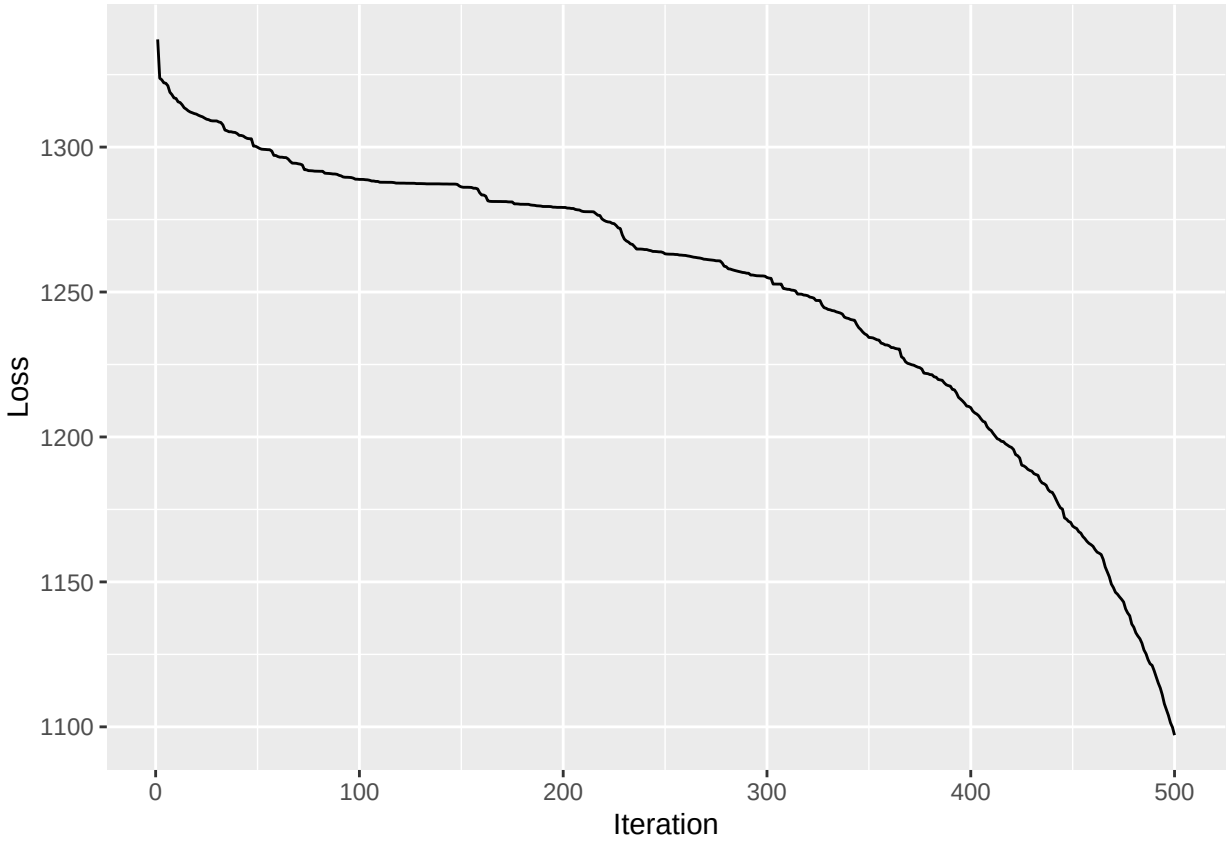
```
# setting up train data
X_train = X[1:2000, ]
X_train = cbind(X0 = rep(1, nrow(X_train)), X_train)
# dim(X_train)
data_train = list(y = as.matrix(train_Y), x = as.matrix(X_train))
# str(data_train)
#setting up test data
X_test = X[2001:2600, ]
# dim(X_test)
data_test = list(y = as.matrix(test_Y), x = as.matrix(X_test))
# str(data_test)
LBoost = function(n_iter=100,train=data_train,test=data_test)
{
  n=1
  output=list()
  # store actual response
  Y_train = train$y
  Y_test = test$y
  # changing data labels for algorithm
  y1 = ifelse(train$y==1,1,0)
  # initialize weak learner
  h_new_train = rep(0,length(train$y))
  h_new_test = rep(0,length(test$y))
  # initial dataset
  X_new_train = train$x
  X_new_test = test$x
  iteration = array()
  loss = array()
  while(n <= n_iter)
  {
    X_train = X_new_train
```

```
    X_test = X_new_test
    h_train = h_new_train
    h_test = h_new_test
    #calculating probablity
    p_train=exp(h_train)/(1+exp(h_train))
    # classifier weight
    w=p_train*(1-p_train)
    # weighted error
    z=(y1-p_train)/w
    # finding best weak learner
    l = apply(X_train,2,function(x,y,w) sum(log(1+exp(- y*predict(lm(y~x-1,weights = w))))),z,w)
    step = which(l==min(l))
    model = lm(z~X_train[,step]-1,weights = w)
    # cbind(model$coefficients*data.frame(X_train[,step]),h_new_train)
    # cbind(model$coefficients*data.frame(X_test[,step])*w,h_new_test)
    h_new_train = h_train+predict(model,weights = w)
    h_new_test = h_test+model$coefficients*data.frame(X_test[,step])
    # calculating Loss
    loss[n] = sum(log(1+exp(- Y_train*h_new_train)))
    # re-valuate data
    X_new_train = X_train[,-step]
    X_new_test = X_test[,-step]
    # iteration
    iteration[n] = n
    # if(n == n_iter){break}
    n=n+1

  }

  # Training Data
  #  Y_train=ifelse(Y_train==1,1,0)
  roc_train=roc(as.numeric(Y_train),as.numeric(h_new_train))
  threshold_train = as.numeric(coords(roc_train, "best", ret = "threshold",drop=TRUE)[1])
  y_hat_train = as.factor(ifelse(h_new_train > threshold_train, 1, -1))
  levels(y_hat_train) = c("-1", "1")
  output$Train_miss = 1 - as.numeric(confusionMatrix(as.factor(Y_train), as.factor(y_hat_train))$byClass
  # Test Data
  #  Y_test=ifelse(Y_test==1,1,0)
  roc_test=roc(as.numeric(Y_test),as.numeric(unlist(h_new_test)))
  threshold_test = as.numeric(coords(roc_test, "best", ret = "threshold",drop=TRUE)[1])
  y_hat_test = as.factor(ifelse(h_new_test > threshold_test, 1, -1))
  levels(y_hat_test) = c("-1", "1")
  output$Test_miss = 1 - as.numeric(confusionMatrix(as.factor(Y_test), as.factor(y_hat_test))$byClass['
  #Loss vs Iteration
  d=data.frame(L=loss,I=iteration)
  LP=ggplot(d,aes(x=I))+geom_line(aes(y=L))+xlab('Iteration')+ylab('Loss')
  output$loss.plot = LP
  #Roc plot test vs train
  RP= ggroc(list(Train = roc_train, Test = roc_test ))+geom_abline(slope=1,intercept = 1,color = "blue"
  output$roc.plot = RP
  return(output)
}
# B1=LBoost(5,data_train,data_test)
```

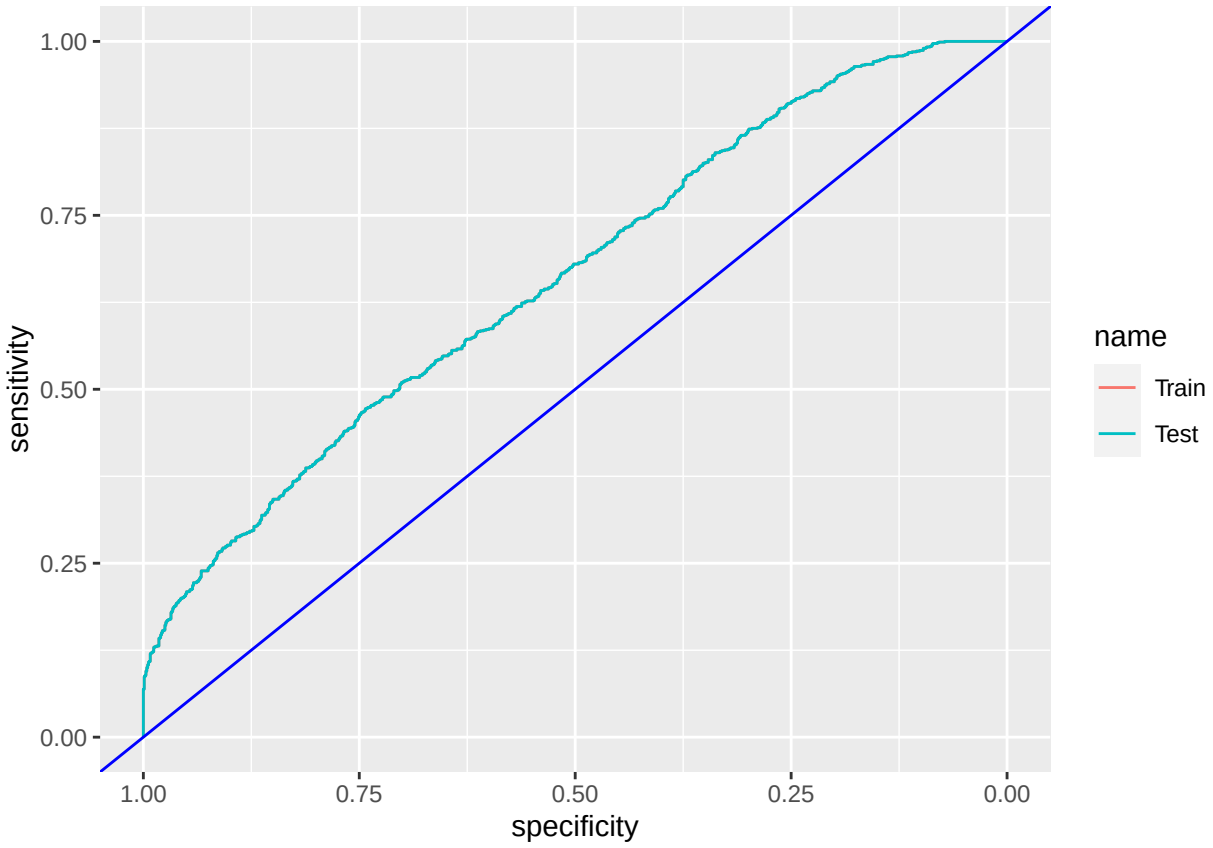
```
#
# B1

boost_iter = c(10,30,100,300,500)
# boost_iter = c(2,3,4,5,6)

rm(Final_LBoost)
Final_LBoost = function(boost_iters=boost_iter,data_train,data_test,n_cores=9)
{
  my.cluster = makeCluster(n_cores)
  registerDoParallel(my.cluster)
  invisible(clusterEvalQ(my.cluster,{library(dplyr)
    library(stargazer)
    library(caret)
    library(pROC)
    library(ggplot2)
    library(gridExtra)   }))
  clusterExport(my.cluster,c("LBoost","data_train","data_test"),envir = .GlobalEnv)
  result = parLapply(my.cluster,boost_iters, LBoost,data_train,data_train)
  stopCluster(my.cluster)
  output = list()
  output$loss_4_500 = result[[5]]$loss.plot
  output$roc_4_100 = result[[3]]$roc.plot
  train_miss = array()
  test_miss = array()
  for(i in 1:length(boost_iters))
  {
    train_miss[i] = result[[i]]$Train_miss
    test_miss[i] = result[[i]]$Test_miss
  }
  D=data.frame(Iteration=boost_iters,Miss_Train = train_miss,Miss_Test = test_miss)
  output$Result = D
  output$Misclassification.plot = ggplot(D,aes(x=D[,1]))+
    geom_line(aes(y=D[,2],color="Train"))+
#    geom_line(aes(y=D[,3],color="Test"))+
    ylab('Misclassification Error')+ xlab('Number of Feature')
  return(output)
}
R1=Final_LBoost(boost_iter,data_train,data_test,n_cores=9)

R1
```


1.00
0.75
0.50
0.25
0.00
sensitivity
1.00
0.75
0.50
0.25
0.00
specificity
name
Train
Test

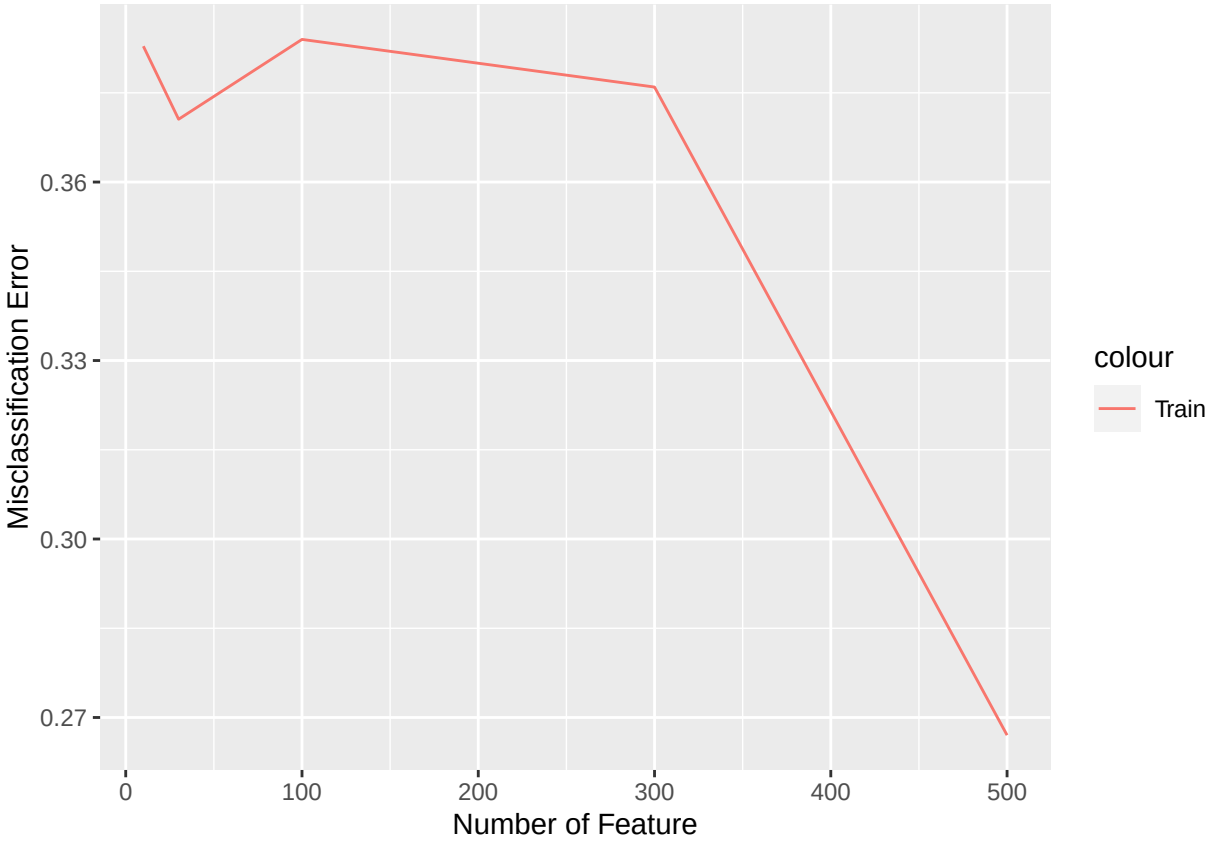
```
## $loss_4_500
```




```
##
## $roc_4_100
```




```
##
## $Result
##   Iteration Miss_Train Miss_Test
## 1        10  0.3828329 0.3828329
## 2        30  0.3705575 0.3705575
## 3       100  0.3839793 0.3839793
## 4       300  0.3759679 0.3759679
## 5       500  0.2670284 0.2670284
##
## $Misclassification.plot
```


Misclassification Error
0.36
0.33
0.30
0.27
0
100
200
300
400
500
Number of Feature
colour
Train