Homework_6

Arnab Aich

2023-02-22

- Loading required Libraries

```
gc()
```

```
##          used (Mb) gc trigger (Mb) max used (Mb)
## Ncells 478102 25.6    1043884 55.8   644242 34.5
## Vcells 855907  6.6    8388608 64.0  1635710 12.5
```

```
packages = c("parallel","doParallel","doSNOW","readr","readsparse","dplyr","stargazer","caret","pROC","g
invisible(xfun::pkg_attach(packages))
```

Defining necessary functions

LogitBoost with regression weak classifier

```
WL=function(x,y,w) {
  model = lm(y~x-1,weights = w)
  pred = predict(model)
  loss = sum(log(1+exp(- as.vector(y) * pred)))
  return(as.numeric(loss))
           }

LBoost = function(n_iter=480,train=data_train,test=data_test,n_cores=11)
{
  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$y))
  # initial dataset
  X_new_train =cbind(rep(0,length(train$y)),train$x)
  X_new_test = cbind(rep(0,length(test$y)),test$x)
  iteration = array()
  loss = array()
  step = array()
  my.cluster1=makeCluster(n_cores)
  registerDoParallel(my.cluster1)
  clusterExport(my.cluster1,c("data_train","data_test","WL"),envir = .GlobalEnv)
```

```
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 = parApply(my.cluster1,data.frame(X_train),2,WL,z,w)
  if(is.na(mean(l)) == 'TRUE'){break}

  step[n] = as.numeric(which(l==min(l)))
  model = lm(z~X_train[,step[n]]-1,weights = w)

  h_new_train = h_train + predict(model,data.frame(X_train[,step[n]])) # str(h_test)
  if(is.na(model$coefficients)=='TRUE') h_theta = 0
  else h_theta = model$coefficients
  h_new_test = h_test + h_theta*as.vector(X_test[,step[n]]) # model$coefficients
  if(is.na(mean(h_new_test))=='TRUE'){break}
  # calculating Loss
  loss[n] = sum(log(1+exp(- as.vector(Y_train)* as.vector(h_new_train))))
  # updating dataset
  X_new_train = X_train[,-step[n]]
  X_new_test = X_test[,-step[n]]
  # iteration
  iteration[n] = n
  n=n+1

}
stopCluster(my.cluster1)
output$Step =step
# Training Data

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$Train_miss = 1 - as.numeric(confusionMatrix(as.factor(Y_train), as.factor(y_hat_train))$byClas
# Test Data

roc_test=roc(as.numeric(unlist(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)
}

boost_iter = c(10,30,100,300,500)
Final_LBoost = function(boost_iters=boost_iter,data_train,data_test,n_cores=9)
{
  result = lapply(boost_iters, LBoost,data_train,data_test,n_cores)
  output = list()
  output$loss_500= result[[5]]$loss.plot
  output$roc_100 = result[[3]]$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=as.vector(boost_iters)))+
    geom_line(aes(y=as.vector(train_miss),color="Train"))+
    geom_line(aes(y=as.vector(test_miss),color="Test"))+
    ylab('Misclassification Error')+ xlab('Number of Feature')
  return(output)
}
```

Dexter Data

- Import Dataset

```
train <-read.sparse("D:/OneDrive - Florida State University/MyFSU_OneDrive/FSU Course Work/5635/Datasets

train_y <- as.matrix(read_csv("D:/OneDrive - Florida State University/MyFSU_OneDrive/FSU Course Work/563
                              col_names = FALSE))
test <-read.sparse("D:/OneDrive - Florida State University/MyFSU_OneDrive/FSU Course Work/5635/Datasets/
test_y <- as.matrix(read_csv("D:/OneDrive - Florida State University/MyFSU_OneDrive/FSU Course Work/5635
                             col_names = FALSE))
```

- Setting Up data

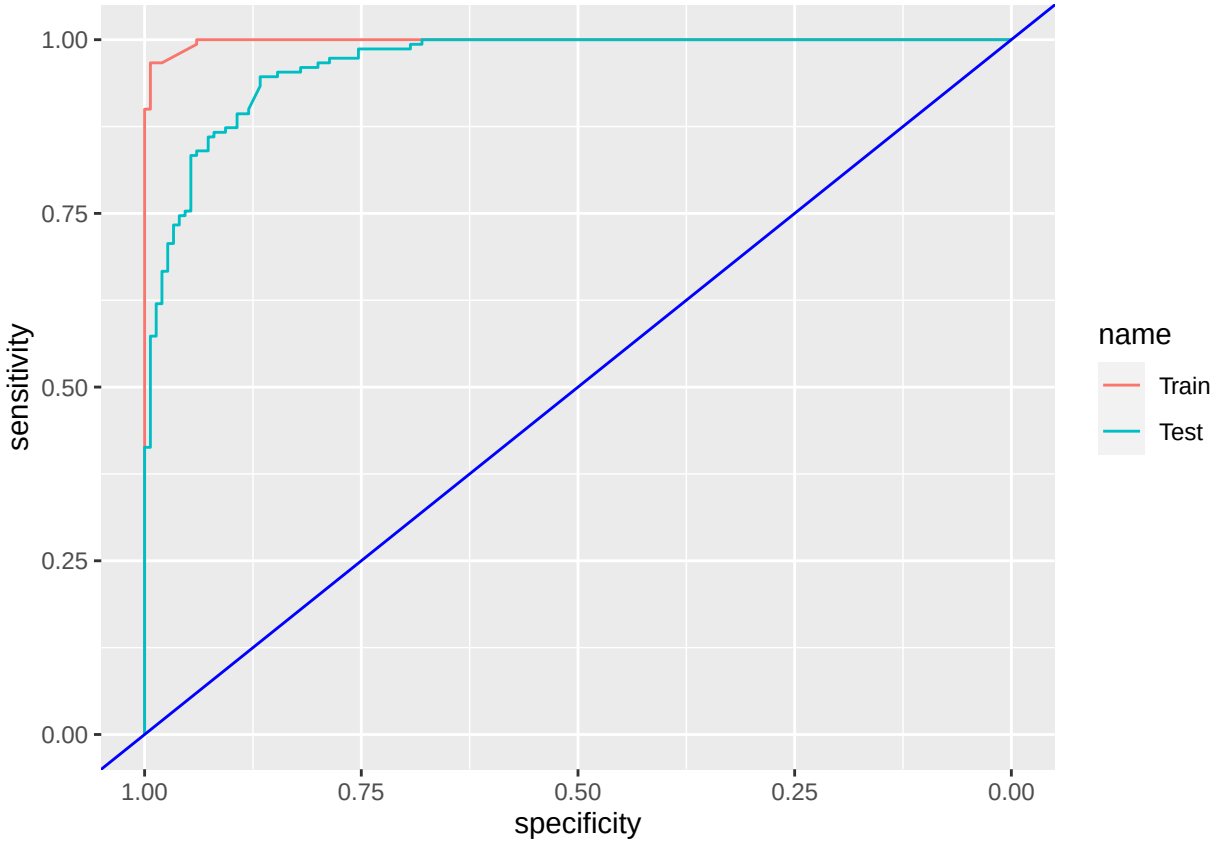
```
x_mean=as.numeric(colMeans(as.matrix(train$X)))
x_sd =as.numeric(apply(as.matrix(train$X),2,sd))

train$x <- as.matrix(scale(train$X,x_mean,x_sd))
test$x <- as.matrix(scale(test$X,x_mean,x_sd))

X = rbind(train$x, test$x)
```

```
X = X[, colSums(is.na(X)) == 0]
data_train = list(y=train_y,x=as.matrix(X[1:300,]))
data_test = list(y=test_y,x=as.matrix(X[301:600,]))
```

```
R=Final_LBoost(boost_iter,data_train,data_test,n_cores=10)
```

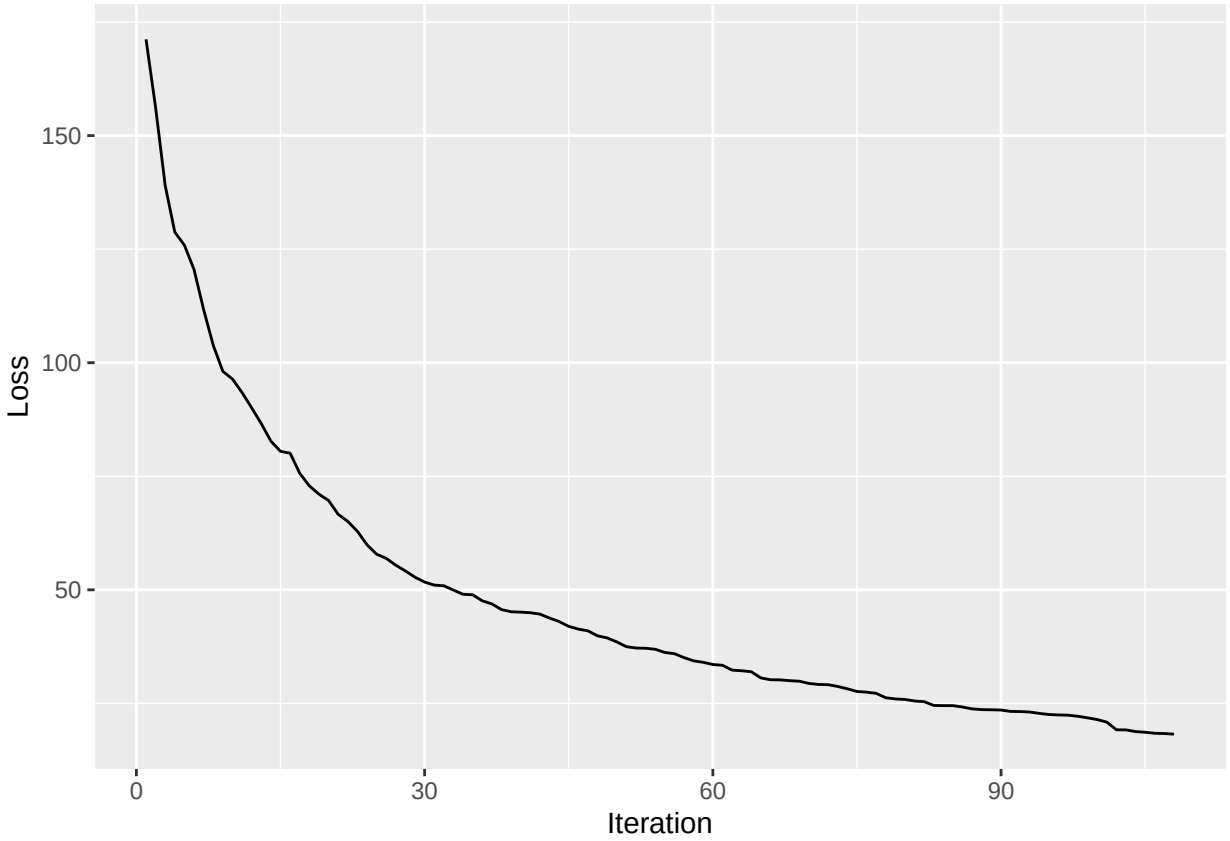
Train and Test ROC curve for 100 Iteration

```
R$roc_100
```



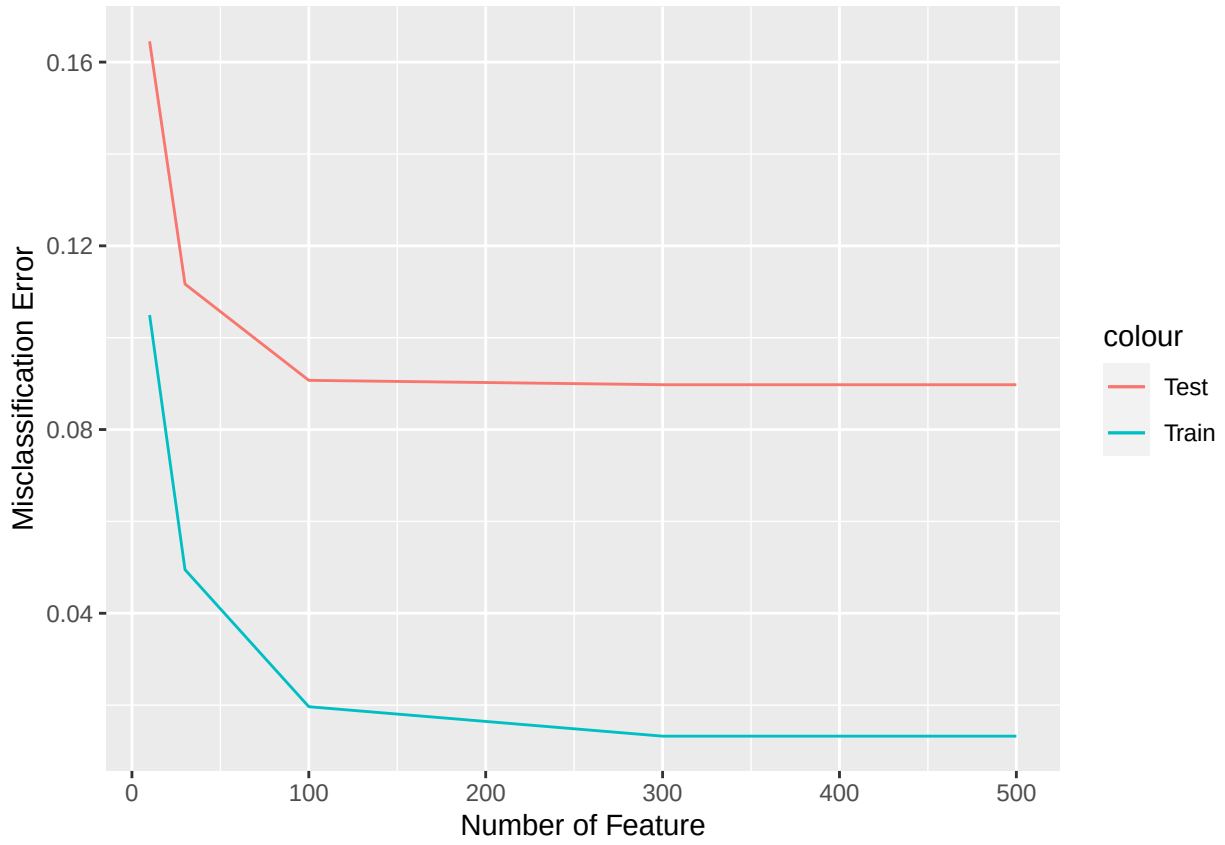
Training Loss vs 500 Iteration

```
R$loss_500
```



Train and Test Misclassification plot

```
R$Misclassification.plot
```



Train and Test Misclassification probability table

```
R$Result
```

```
  Iteration Miss_Train  Miss_Test
1        10 0.10492549 0.16451959
2        30 0.04949944 0.11167890
3       100 0.01965842 0.09071390
4       300 0.01324680 0.08975968
5       500 0.01324680 0.08975968
```

Madelon Data

- Import Dataset

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

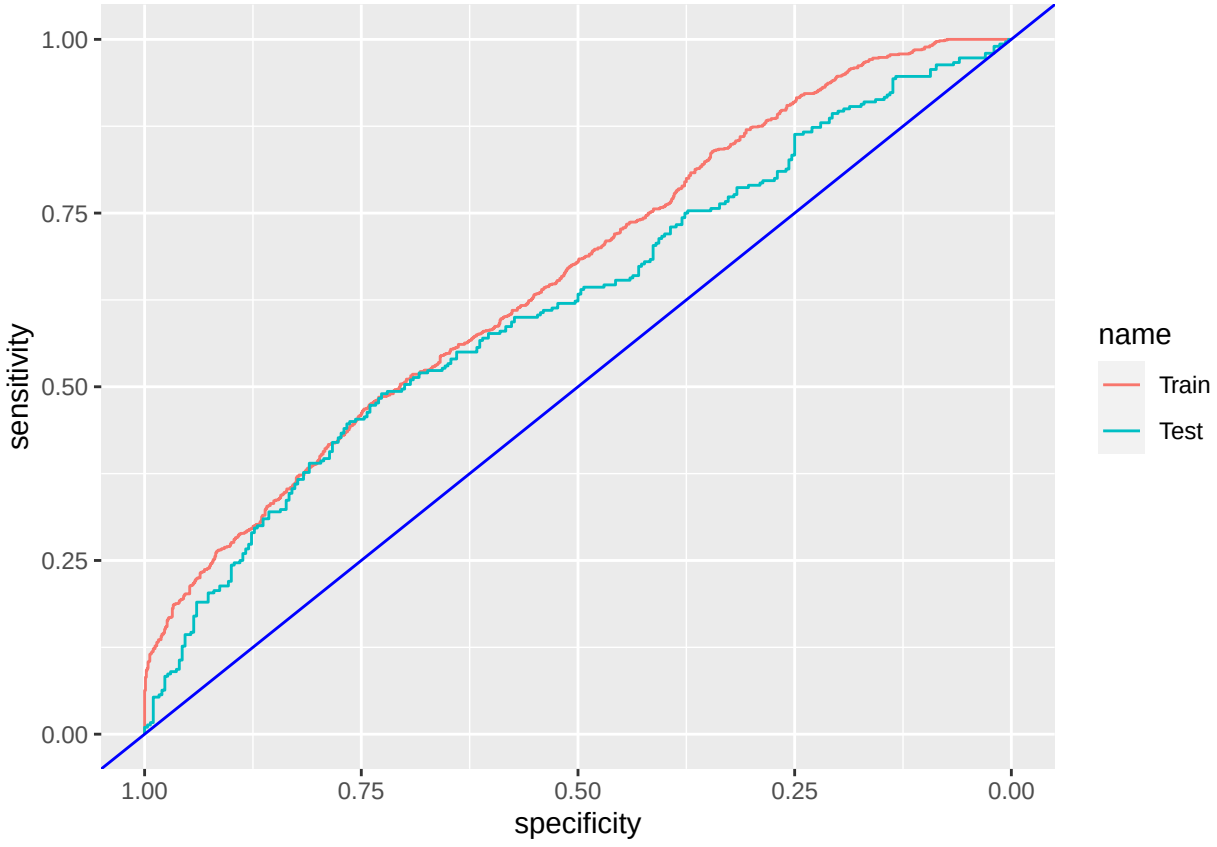
- Setting Up data

```
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]

X_train = X[1:2000, ]
data_train= list(y = as.matrix(train_Y), x = as.matrix(X_train))
X_test = X[2001:2600, ]
data_test = list(y = as.matrix(test_Y), x = as.matrix(X_test))
```

```
R=Final_LBoost(boost_iter,data_train,data_test,n_cores=10)
```

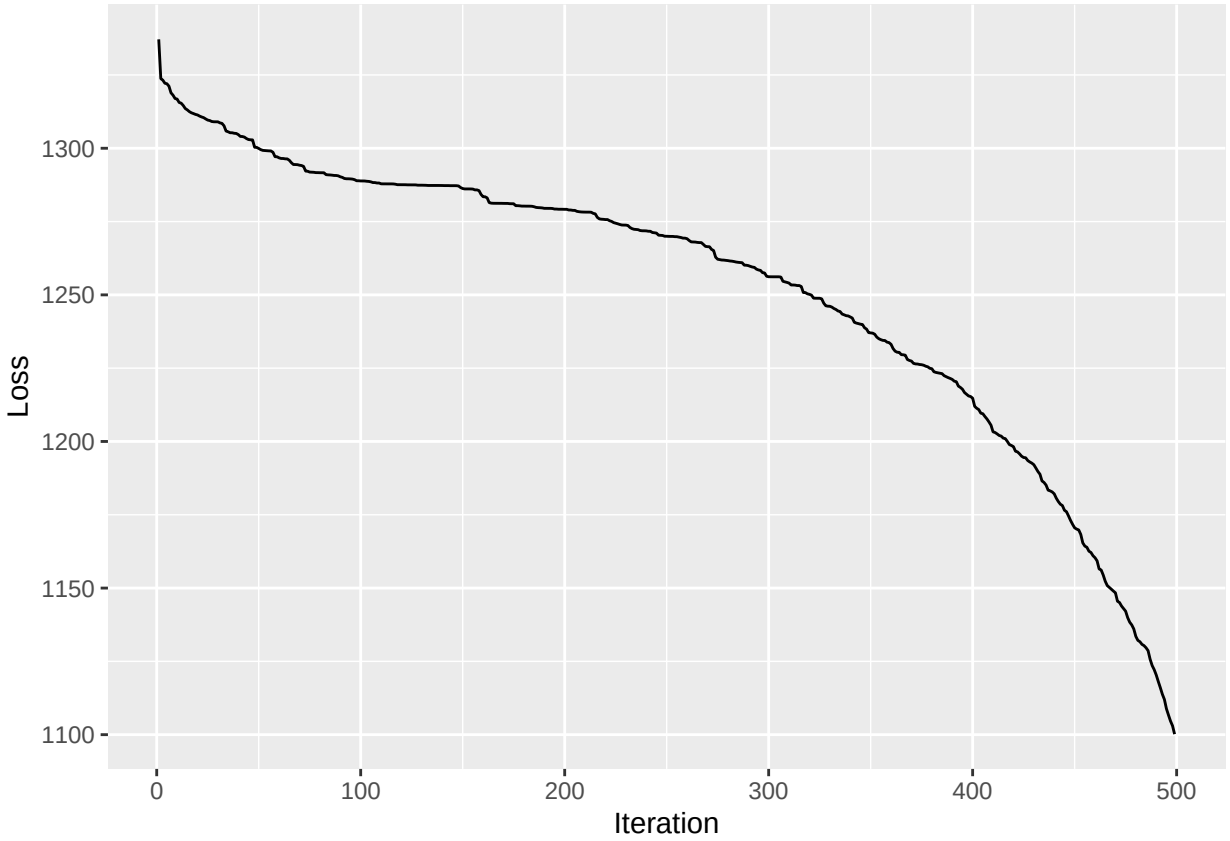
Train and Test ROC curve for 100 Iteration

```
R$roc_100
```



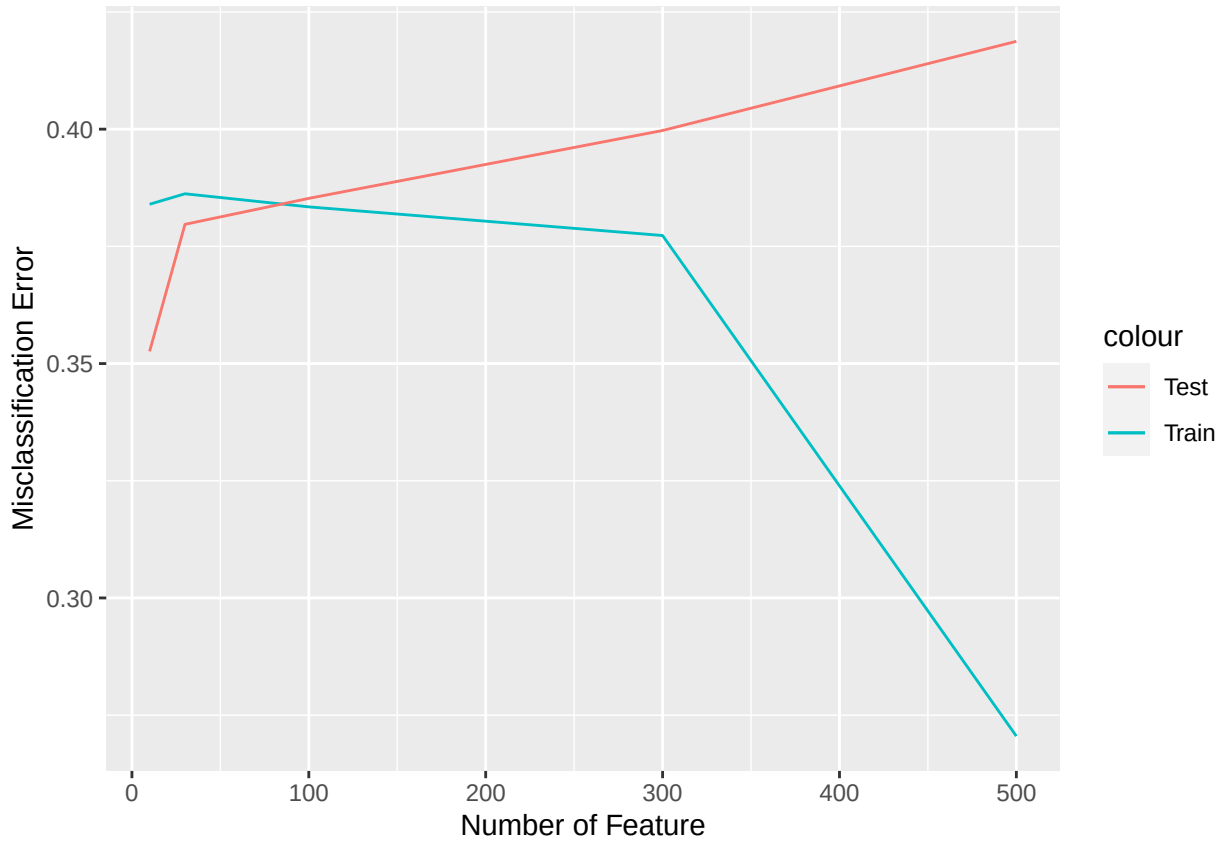
Training Loss vs 500 Iteration

```
R$loss_500
```



Train and Test Misclassification plot

```
R$Misclassification.plot
```



Train and Test Misclassification probability table

```
R$Result
```

```
  Iteration Miss_Train Miss_Test
1        10  0.3839864 0.3526040
2        30  0.3862169 0.3796945
3       100  0.3834257 0.3852388
4       300  0.3773134 0.3997147
5       500  0.2705179 0.4187329
```

Gisette Data

- Import Dataset

```
train_X <-
  read_table(
    "D:/OneDrive - Florida State University/MyFSU_OneDrive/FSU Course Work/5635/Datasets/Gisette/gisett
    col_names = FALSE
  )
test_X <-
  read_table(
    "D:/OneDrive - Florida State University/MyFSU_OneDrive/FSU Course Work/5635/Datasets/Gisette/gisett
    col_names = FALSE
  )
train_Y <-
```

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

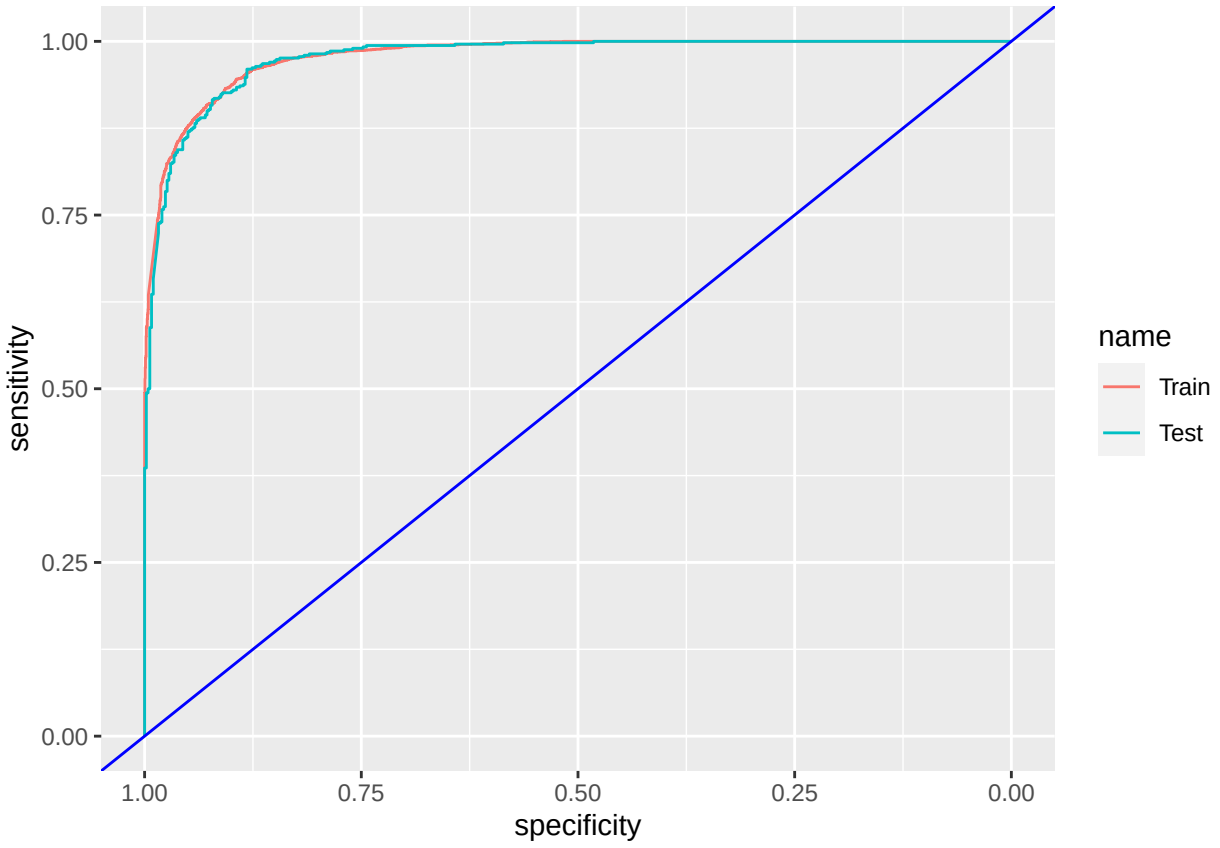
- Setting Up data

```
x_mean=as.numeric(colMeans(train_X[,-5001]))
x_sd =as.numeric(apply(train_X[,-5001],2,sd))
X = rbind(scale(train_X[, -5001],center=x_mean,scale=x_sd),
          scale(test_X[, -5001],center=x_mean,scale=x_sd))
X = X[, colSums(is.na(X)) == 0]
X_train = X[1:6000, ]
data_train = list(y = as.matrix(train_Y), x = as.matrix(X_train))
X_test = X[6001:7000, ]
data_test = list(y = as.matrix(test_Y), x = as.matrix(X_test))
```

```
R=Final_LBoost(boost_iter,data_train,data_test,n_cores=10)
```

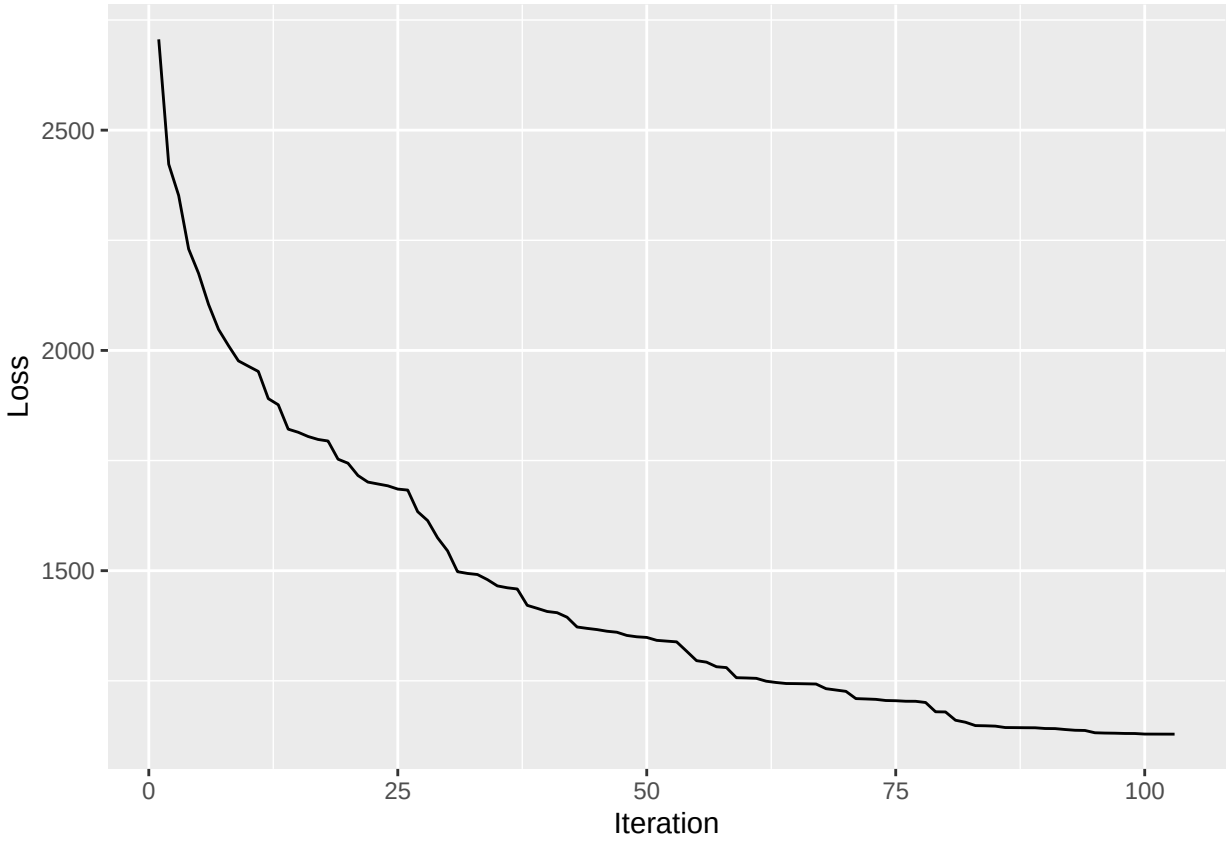
Train and Test ROC curve for 100 Iteration

```
R$roc_100
```



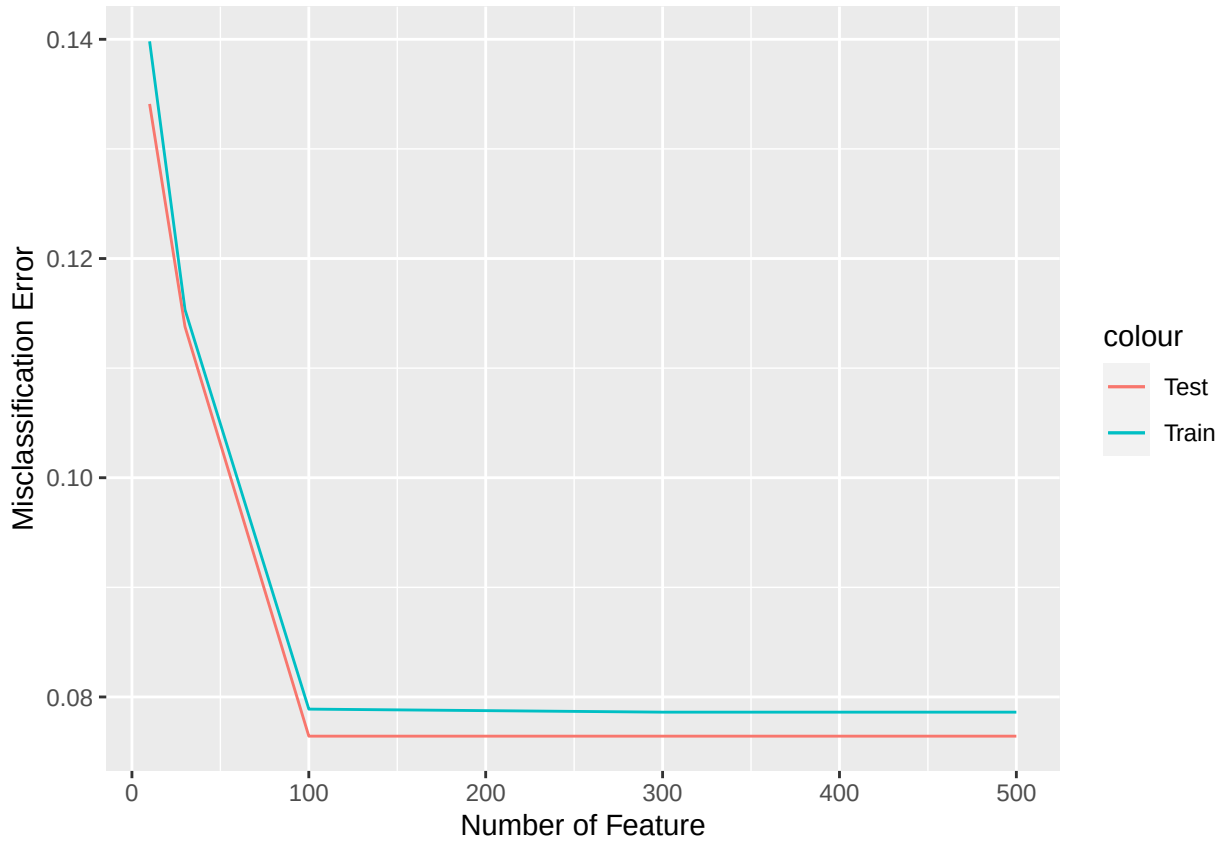
Training Loss vs 500 Iteration

```
R$loss_500
```



Train and Test Misclassification plot

```
R$Misclassification.plot
```



Train and Test Misclassification probability table

```
R$Result
```

```
  Iteration Miss_Train  Miss_Test
1        10 0.13981322 0.13410319
2        30 0.11533265 0.11377754
3       100 0.07889033 0.07642296
4       300 0.07861320 0.07642296
5       500 0.07861320 0.07642296
```