ASSIGNMENT 2 SML

Détagnon Bernic GBAGUIDI

2024-01-18

```r
library(audio)
```

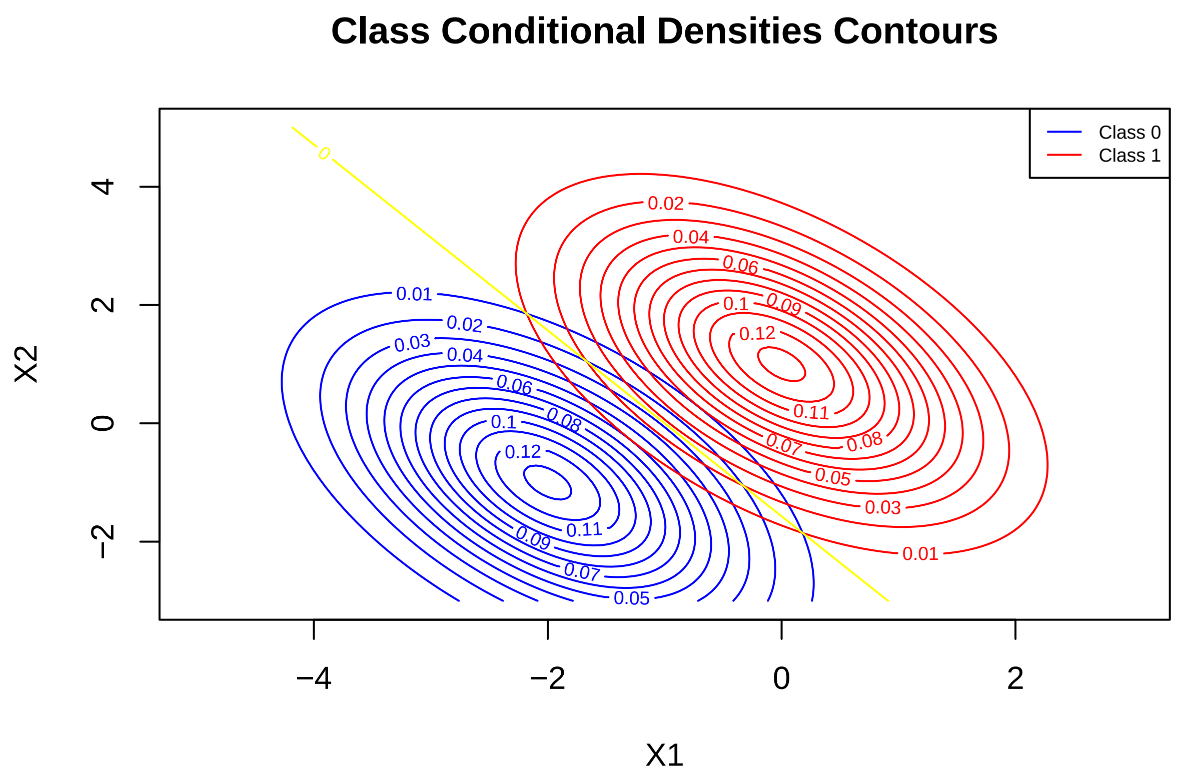
```r
    library(e1071)
    library(class)
    library(MASS)
    library(kernlab)
    library(mlbench)
    library(reshape2)
    library(ROCR)

    library(ada)
    library(adabag)
    library(ipred)
    library(survival)
    library(rchallenge)
    library(PerformanceAnalytics)
    library(knitr)
    library(acepack)
    library(caret)
    library(HSAUR2)
    library(corrplot)
```

Exercice 2:When can we compute the Bayes' Risk?

1- Let sketch the contour of the two classes.

```r
library(mvtnorm)



mu0 <- c(-2, -1)
mu1 <- c(0, 1)
Sigma <- matrix(c(1, -3/4, -3/4, 2), nrow = 2, ncol = 2)

#Create a grid of points
x <- seq(-5, 3, length.out = 150)
y <- seq(-3, 5, length.out = 150)
grid <- expand.grid(x = x, y = y)

# Evaluate the PDFs for each point
pdf0 <- dmvnorm(grid, mean = mu0, sigma = Sigma)
pdf1 <- dmvnorm(grid, mean = mu1, sigma = Sigma)
```

```r
# Reshape the PDFs for contour plotting
pdf0matrix <- matrix(pdf0, nrow = length(x), ncol = length(y))
pdf1matrix <- matrix(pdf1, nrow = length(x), ncol = length(y))

# Plot the contours
contour(x, y, pdf0matrix, main = "Class Conditional Densities Contours", xlab = "X1", ylab = "X2", col =
contour(x, y, pdf1matrix, add = TRUE, col = "red")

# Add legend
legend("topright", legend = c("Class 0", "Class 1"), col = c("blue", "red"), lty = 1,cex = 0.6)

contour(x, y, pdf0matrix - pdf1matrix, levels = 0, col = "yellow", add = TRUE)
```



Exercice 3:Detecting and Recognizing Speaker Accent

1- Let consider the dataset The datasets accent-raw-data-1.csv. 1) Let comment on the peculiarities of the dataset from a point of view dimensionality.

```r
dataa <- read.csv("accent-raw-data-1.csv")
```

```r
x <- as.matrix(dataa[,-1])
y <- dataa[,1]
dim(x)
```

```
## [1]   329 39680
```

```r
table(y)
```

```
## y
##  ES  FR  GE  IT  UK  US
##  29  30  30  30  45 165
```

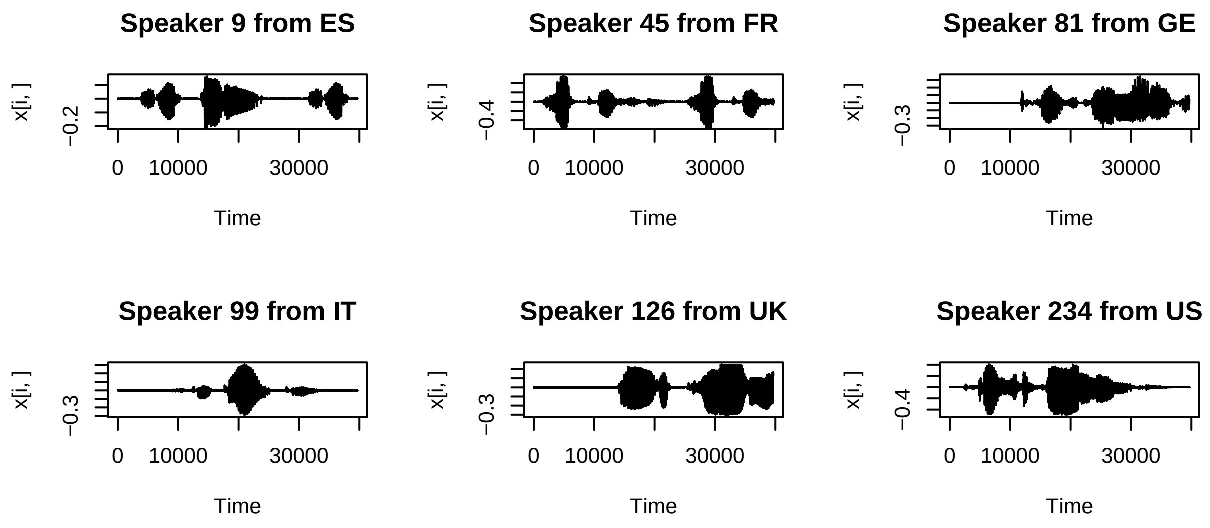
The data set has 326 observations(speakers) and 39680 variables that represent the dimensionality on the input space. The response is categorical and has 6 levels : " ES", "FR" , "GE", "IT", "UK", US.

We can notice that the number of variables is very large than the number of observations, we are in presence of ultra high dimension.

The ration n/p is then small than 5 and this is bad in the context of setting dimentionality. This is due to the fact that each observation is an audio recorded.

2) Let plot the soundtracks for theses speakers.

```r
par(mfrow=c(3,3))
for (i in c(9, 45, 81, 99, 126, 234)){
ts.plot(x=x[i,],main=paste("Speaker",i,"from",y[i]))
}
```



3) Let comment comparatively on the features of the plotted soundtracks.

We can remark that it is the graph of the soundtrack of each country. In each graph the variation or frequence is different . This is the explaination of the fact that for each country the accent is not the same.

Let hear from the speaker

```r
play(x[9,])
play(x[45,])
```

```r
play(x[81,])
```

```r
play(x[99,])
```

```r
play(x[126,])
```

```r
play(x[234,])
```

4) Let comment on the use of Classification Trees as learning machines for classifying the speaking accent using this data.

Firstly , in the context of tree learning machine we are going to choose the variable that's separate the most the response variable, but we can remark that the number of variables is very large. This mean that it won't be easy to explore the data and to know if it is good to apply the tree learning machine. Further , even if we decide that to apply this machine we will lose the best treasure of tree that's it **interpertability**.

But if we are going to use tree learning machine , in order to reduce the number of variables, we can use the compression methods like PCA .

5) Let comment on the use of kNearest Neighbors as learning machines for classifying the speaking accent using this data.

KNearest Neighbors has a very simple principe of learning, and this lead to a large range of data set where it can perform well. On this data set we can use the Knearest Neighbors . The only fear will be the time complexity . But KNN is reputated to cost in time complexity because it must compute the distance for each observations . So here the fact that KNN will take time will not the reason to notn be good for this data set. Each time for any data set we are able to find the best K that classify the response no matter the time .Another machine can be better on prediction on this data set but KNN will be between the best candidate.

2- Let consider now the dataset accent-mfcc-data-1.csv along with the binary classification task of US Born versus Non-US Born speakers. You are to compare the following methods of classification: (1) kNearest Neighbors (2) Trees (3) Support Vector Machines.

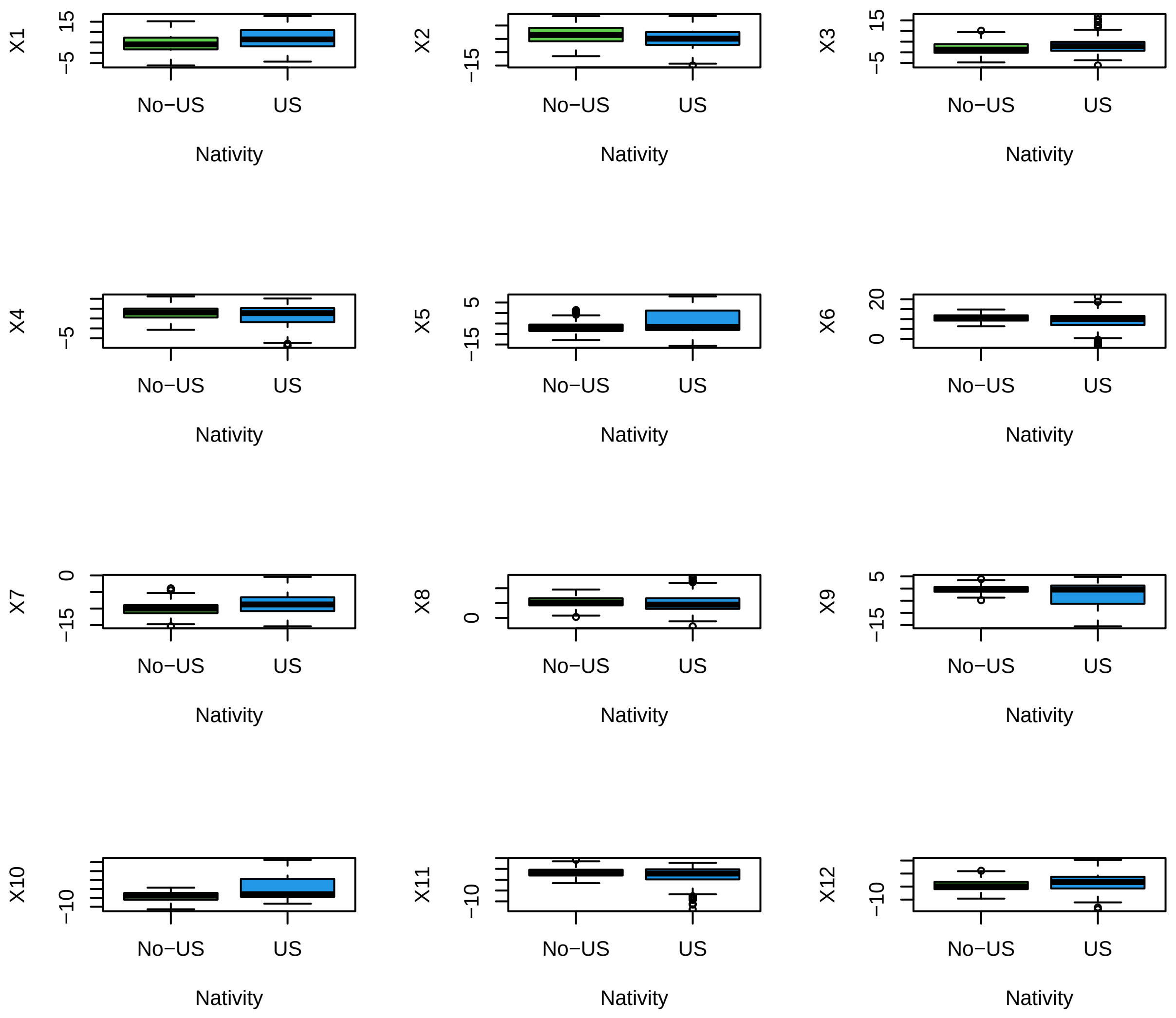
Exploration of data

```r
dataaa <- read.csv("accent-mfcc-data-1.csv")
```

```r
xx <- dataaa[,-1]
yy <- dataaa[,1]
```

```r
#### recode the respose in binary

yy <- ifelse(yy=="US","US","No-US")
```

```r
par(mfrow=c(3,3))
for (i in 1:ncol(xx)){


  boxplot(xx[,i]~yy, xlab='Nativity',
              ylab=colnames(xx)[i],col=3:4)
}
```

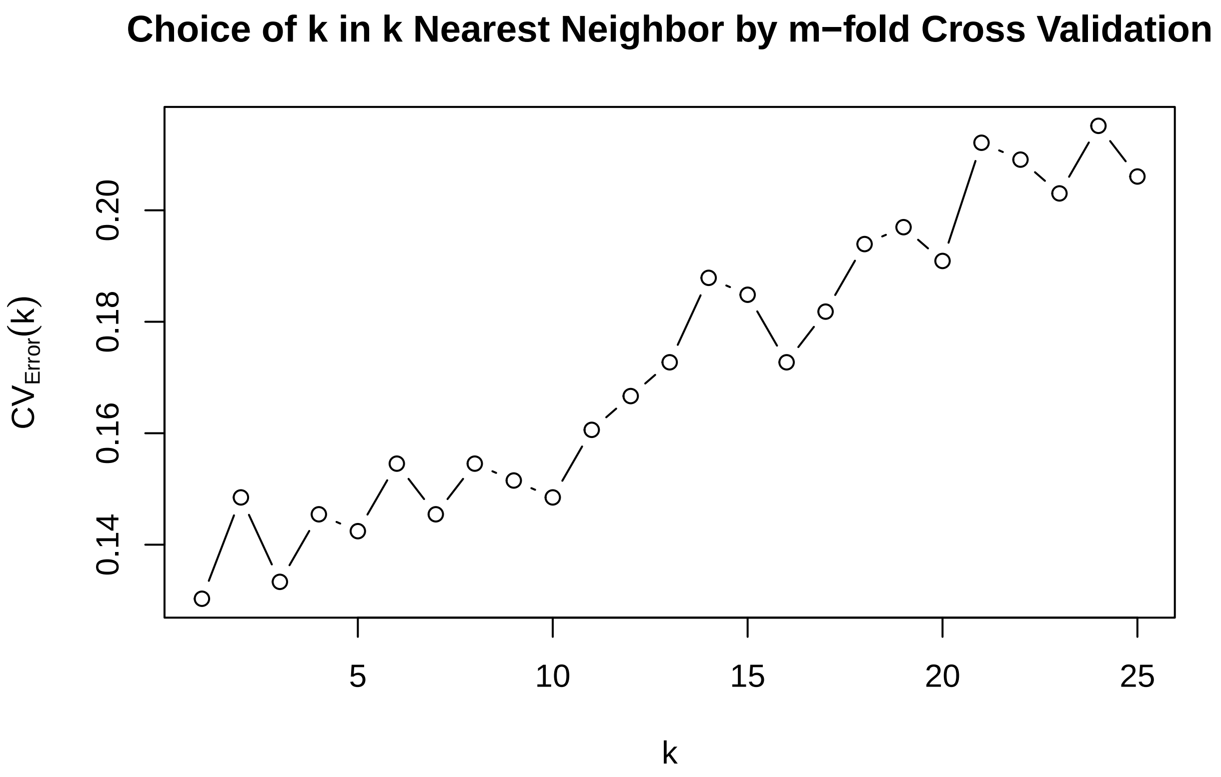

X1
15
−5
X2
−15
X3
15
−5
X4
−5
X5
5
−15
X6
20
0
X7
0
−15
X8
0
X9
5
−15
X10
−10
X11
−10
X12
−10
No−US
US
Nativity

1)Let generate separate confusion matrices for each of the three methods

For the knn

Here there are not a specifique K, so we are going to choose it doing cross validation .

```r
set.seed(20000604)
xtr <- xx
   ytr <- yy

   vK <- seq(1, 25, by=1)
   nK <- length(vK)
   cv.error <-numeric(nK)
   nc <- nrow(xtr)
   c   <- 10

   S   <- sample(sample(nc))
   m   <- ceiling(nc/c)

   held.out.set <- matrix(0, nrow=c, ncol=m)

   for(ic in 1:(c-1))
   {
    held.out.set[ic,] <- S[((ic-1)*m + 1):(ic*m)]
   }

   held.out.set[c, 1:(nc-(c-1)*m)] <- S[((c-1)*m + 1):nc]
# Running the cross validation itself

  for(j in 1:nK)
  {
    for(i in 1:c)
    {
      out <-  held.out.set[i,]
      yhatc<- knn(xtr[-out,], xtr[out,],ytr[-out],  k=vK[j])
      cv.error[j]<-cv.error[j] + (length(out)-sum(diag(table(ytr[out],yhatc))))/length(out)
    }
    cv.error[j]<-cv.error[j]/c
  }


# Plot the cross validation curve

  plot(vK, cv.error, xlab='k', ylab=expression(CV[Error](k)),
      main='Choice of k in k Nearest Neighbor by m-fold Cross Validation')
  lines(vK, cv.error, type='c')
```



```
idm <- which(cv.error==min(cv.error))
K=vK[idm]
K
```

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

We can then conclude that the best K according the cross validation we did is the K above.

Confusion matrix on the train set

```
### Confusion matrice one the trainning set.

yknn <- knn(xx,xx,yy,k = K,prob = TRUE)
table(yy,yknn)
```

```
##         yknn
## yy       No-US  US
##   No-US    164   0
##   US         0 165
```

For the tree classification

The exploration did before can help to decide which variable separate the best the response.

```
### Confusion matrice on the train data

ytreeModel <- rpart(yy~.,data = dataaa[,-1])
ytree <- predict(ytreeModel,xx,type="class")
table(yy,ytree)
```

```
##        ytree
## yy      No-US  US
##   No-US   154  10
##   US       23 142
```

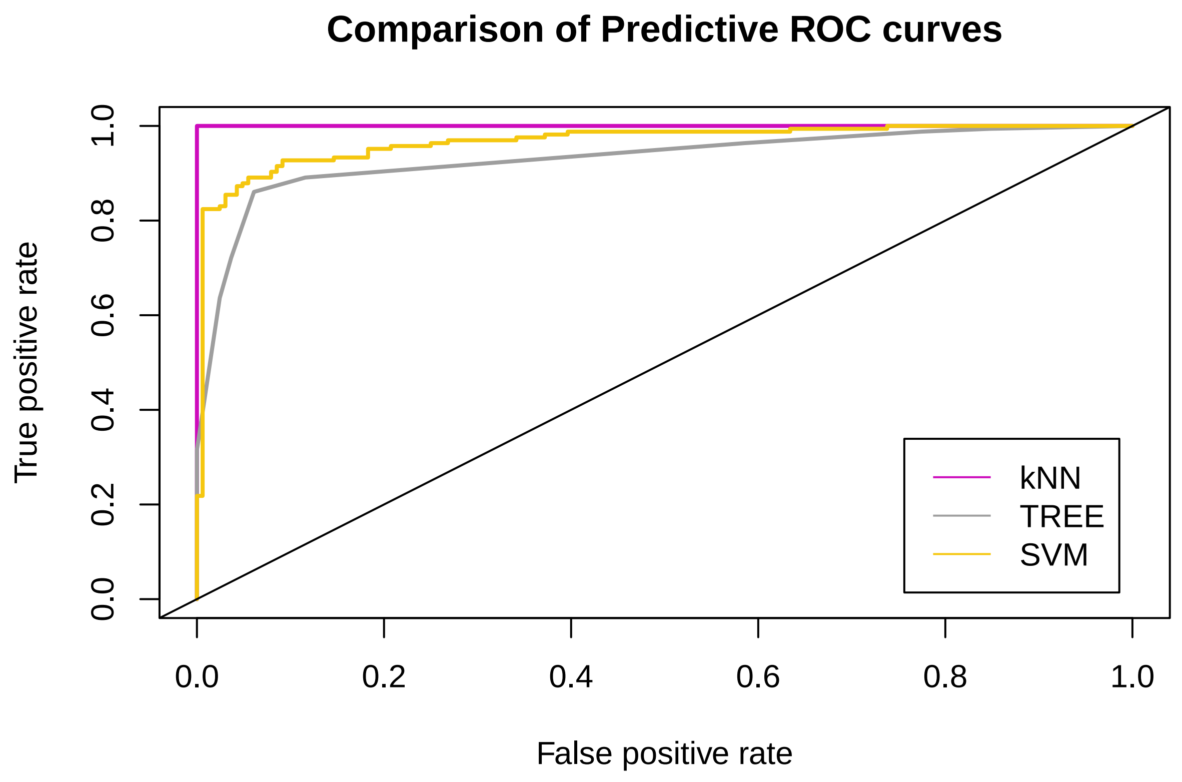
For Support vector machines

```
#### Confusion matrice on the train set
ysvmModel <- ksvm(as.factor(yy)~.,data = xx, kernel='rbfdot', type='C-svc', prob.model=TRUE)
ysvm <- predict(ysvmModel,xx)
table(yy,ysvm)
```

```
##        ysvm
## yy      No-US  US
##   No-US   159   5
##   US       27 138
```

2) Let plot a comparative roc curve for all three models

```
  prob      <- attr(yknn, 'prob')
  prob      <- 2*ifelse(yknn == "No-US", 1-prob, prob) - 1
  predkNN <- prediction(prob, yy)
  perf.kNN <- performance(predkNN, measure='tpr', x.measure='fpr')



  prob      <- predict(ytreeModel, xx, type='prob')[,2]
  predtree <- prediction(prob, yy)
  perf.tree <- performance(predtree, measure='tpr', x.measure='fpr')



  prob     <- predict(ysvmModel, xx, type='probabilities')[,2]
  predsvm <- prediction(prob, yy)
  perf.svm <- performance(predsvm, measure='tpr', x.measure='fpr')

    plot(perf.kNN, col=6, lwd= 2, lty=1, main=paste('Comparison of Predictive ROC curves'))
    plot(perf.tree, col=8, lwd= 2, lty=1, add=TRUE)
    plot(perf.svm, col=7, lwd= 2, lty=1, add=TRUE)
      abline(a=0,b=1)
    legend('bottomright', inset=0.05, c( 'kNN','TREE','SVM'),
         col=c(6,8,7), lty=1)
```



3) Let use a 60% - 40% Training-Test set split to generate comparative boxplots of the test error based on 100 replications.

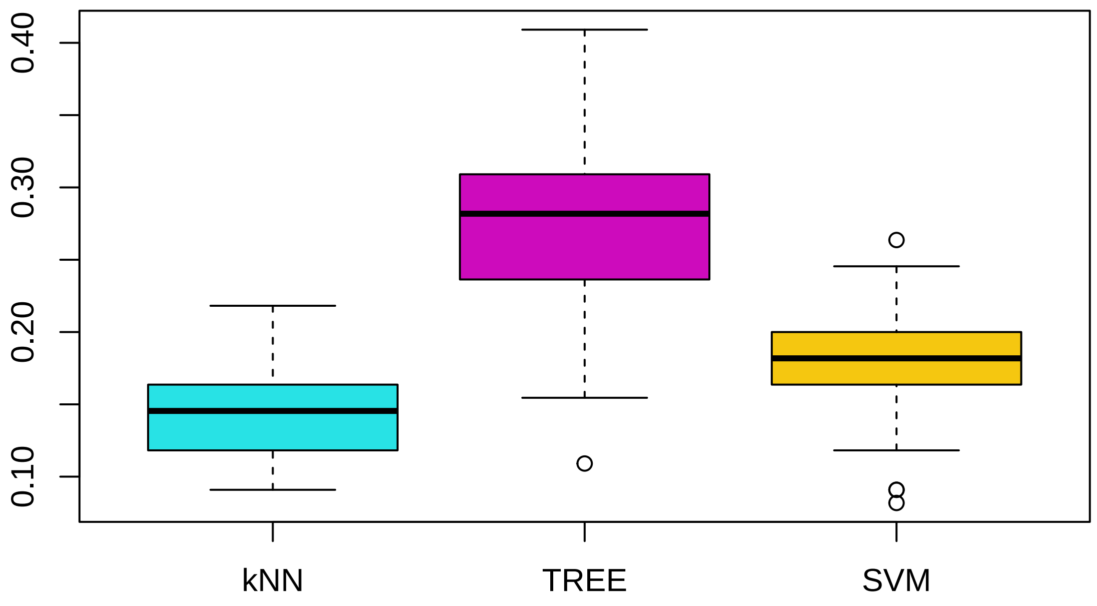
Here the function for stratified sampling

```
stratified.holdout <- function(y, ptr)
  {
      n               <- length(y)
      labels          <- unique(y)
      id.tr <- id.te <- NULL



      y <- sample(sample(sample(y)))

      for(j in 1:length(labels))
      {
        sj    <- which(y==labels[j])
        nj    <- length(sj)

        id.tr <- c(id.tr, (sample(sample(sample(sj))))[1:round(nj*ptr)])
      }

      id.te  <- (1:n) [-id.tr]

 return(list(idx1=id.tr,idx2=id.te))
```

```
  }
```

```
set.seed (19671210)
  n <- nrow(xx)
  epsilon <- 1/3
  nte     <- round(n*epsilon)
  ntr     <- n - nte



  R <- 100
   test.err <- matrix(0, nrow=R, ncol=3)

  for(r in 1:R)
  {
    # Split the data

    hold <- stratified.holdout(as.factor(yy), 1-epsilon)
    id.tr <- hold$idx1
    id.te <- hold$idx2
    ntr   <- length(id.tr)
    nte   <- length(id.te)

    y.te         <- yy[id.te]


   # Nearest Neighbors Learning Machine

    y.te.hat       <- knn(xx[id.tr,], xx[id.te,], yy[id.tr], k=K, prob=TRUE)
    ind.err.te     <- ifelse(y.te!=y.te.hat,1,0)
    test.err[r,1]  <- mean(ind.err.te)


    # Classification Trees

    tree.mod       <- rpart(as.factor(yy[id.tr])~., data=xx[id.tr, ])
    y.te.hat       <- predict(tree.mod, xx[id.te, ], type='class')
    ind.err.te     <- ifelse(y.te!=y.te.hat,1,0)
    test.err[r,2]  <- mean(ind.err.te)


    # Support Vector Machines

    svm.mod        <- ksvm(as.factor(yy[id.tr])~., data=xx[id.tr, ], kernel='rbfdot', type='C-svc', prob
    y.te.hat       <- predict(svm.mod, xx[id.te, ])
    ind.err.te     <- ifelse(y.te!=y.te.hat,1,0)
    test.err[r,3]  <- mean(ind.err.te)
  }


  test <- data.frame(test.err)

  Method<-c('kNN','TREE','SVM')
  colnames(test) <- Method
```

```
boxplot(test,col = 5:7)
```



4) Let comment on the predictive performance.

From the boxplot above , the machine that make the less errors is KNN , the next is the SVM learning machine and the last one is the tree learning machine. So in terme of performance the KNN machine is the best one to make a prediction between the three.

5) Let reconsider the confusion matrix of the best method and comment on the similarity between speaking accents.

```
### Confusion matrix on the test set
 y.te.hat       <- knn(xx[id.tr,], xx[id.te,], yy[id.tr], k=K, prob=TRUE)

  table(y.te,y.te.hat)
```

```
##        y.te.hat
## y.te    No-US US
##   No-US    51  6
##   US        7 46
```

Based on this confusion matrix , the False positive rate is 0.11 then we can say that 11% of the NON-US born speaker's accent is similar to US born speaker.

Futhermore, the False negative rate is 0.13, then we can say that 13% of the US born speaker accent are similar to NON-US born speaker accent .

Another Measure: Sensivity=$\frac{TP}{TP+FN}$ = 0.867

Specificity$=\dfrac{TN}{TN+FP}=0.894$