Project 3 Group 3

Nishanth Gandhidoss, Raghavendran Shankar

4 March 2017

Question 1

(24 points) Write a function that will evaluate the predictions of a classifier. That is, given a vector of predicted values, Y_{pred}, and actual class values, Y_{true}, can report the following quantities:

- number of true positives (TP),
- number of false positives (FP),
- number of true negatives (TN),
- number of false negatives (FN),
- true positive rate (TPR),
- true negative rate (TNR),
- sensitivity (Sens.),
- specificity (Spec.),
- precision (Prec.),
- recall (Rec.),
- accuracy (Acc.), and
- error rate (Err)

```
# Creting S4 object for the predcition evaluation
setClass("prediction", representation(predictions = "list", labels = "list",
cutoffs = "list",
                                      fp = "list", tp = "list", tn = "list",
fn = "list",
                                      n.pos = "list", n.neg = "list",
n.pos.pred = "list",
                                      n.neg.pred = "list", tpr = "numeric",
fpr = "numeric",
                                      tnr = "numeric", fnr = "numeric",
sensitivity = "numeric",
                                      specificity = "numeric", precision =
"numeric",
                                      recall = "numeric", accuracy =
"numeric", error_rate = "numeric"))

# Function to Evaluate the prediction
# Returns the necessary measures depeing
# upon the passed argument
predictions <- function (predictions, labels, label.ordering = NULL) {
    # Check if data frame
```

```r
    if(is.data.frame(predictions)) {
        names(predictions) <- c()
        predictions <- as.list(predictions)
    }
    # Check if it is matrix
    else if(is.matrix(predictions)) {
        predictions <- as.list(data.frame(predictions))
        names(predictions) <- c()
    }
    # Check if it is Vector
    else if(is.vector(predictions) && !is.list(predictions)) {
        predictions <- list(predictions)
    }
    # Stop if prediction is not of type list
    else if(!is.list(predictions)) {
        stop("Format of predictions is invalid.")
    }

    # Assign the labels
    if(is.data.frame(labels)) {
        names(labels) <- c()
        labels <- as.list(labels)
    }
    else if(is.matrix(labels)) {
        labels <- as.list(data.frame(labels))
        names(labels) <- c()
    }
    else if((is.vector(labels) || is.ordered(labels) || is.factor(labels)) &&
!is.list(labels)) {
        labels <- list(labels)
    }
    # Stop if labels is not of type list
    else if(!is.list(labels)) {
        stop("Format of labels is invalid.")
    }
    # Length check for single and multiple evaluations
    if(length(predictions) != length(labels))
        stop(paste("Number of cross-validation runs must be equal for
predictions and labels."))
    if(!all(sapply(predictions, length) == sapply(labels, length)))
        stop(paste("Number of predictions in each run must be equal to the
number of labels for each run."))

    for(i in 1:length(predictions)) {
        finite.bool <- is.finite(predictions[[i]])
        predictions[[i]] <- predictions[[i]][finite.bool]
        labels[[i]] <- labels[[i]][finite.bool]
    }

    label.format = ""
    # Assign label fotmat for each type depending upon certain conditions
    if(all(sapply(labels, is.factor)) && !any(sapply(labels, is.ordered))) {
        label.format <- "factor"
    }
    else if(all(sapply(labels, is.ordered))) {
        label.format <- "ordered"
    }
    else if(all(sapply(labels, is.character)) || all(sapply(labels,
is.numeric)) ||
            all(sapply(labels, is.logical))) {
        label.format <- "normal"
    }
    else{
        stop(paste("Inconsistent label data type across different cross-
validation runs."))
    }
    if(!all(sapply(labels, levels) == levels(labels[[1]]))) {
        stop(paste("Inconsistent factor levels across different cross-
validation runs."))
    }

    # Depending on the label format level assign ment is made
    levels <- c()
    if(label.format == "ordered") {
        if(!is.null(label.ordering)) {
            stop(paste("'labels' is already ordered. No additional
'label.ordering' must be supplied."))
        }
        else{
            levels <- levels(labels[[1]])
        }
    }
    else{
        if(is.null(label.ordering)) {
            if(label.format == "factor")
                levels <- sort(levels(labels[[1]]))
            else
                levels <- sort(unique(unlist(labels)))
        }
        else{
            if(!setequal(unique(unlist(labels)), label.ordering)) {
                stop("Label ordering does not match class labels.")
            }
            levels <- label.ordering
        }
        for(i in 1:length(labels)) {
            if(is.factor(labels))
                labels[[i]] <- ordered(as.character(labels[[i]]), levels =
levels)
```

```r
            else labels[[i]] <- ordered(labels[[i]], levels = levels)
        }
    }
    if(length(levels) != 2) {
        stop("Number of classes is not equal to 2.")
    }
    if(!is.numeric(unlist(predictions))) {
        stop("Currently, only continuous predictions are supported.")
    }

    # Intailizing empty list
    cutoffs <- list()
    fp <- list()
    tp <- list()
    fn <- list()
    tn <- list()
    n.pos <- list()
    n.neg <- list()
    n.pos.pred <- list()
    n.neg.pred <- list()

    # Loop to perform the caluclation for evaluation metrics using the labels
    for(i in 1:length(predictions)) {
        n.pos <- c(n.pos, sum(labels[[i]] == levels[2]))
        n.neg <- c(n.neg, sum(labels[[i]] == levels[1]))
        ans <- .compute.unnormalized.roc.curve(predictions[[i]],
            labels[[i]])
        cutoffs <- c(cutoffs, list(ans$cutoffs))
        fp <- c(fp, list(ans$fp))
        tp <- c(tp, list(ans$tp))
        fn <- c(fn, list(n.pos[[i]] - tp[[i]]))
        tn <- c(tn, list(n.neg[[i]] - fp[[i]]))
        n.pos.pred <- c(n.pos.pred, list(tp[[i]] + fp[[i]]))
        n.neg.pred <- c(n.neg.pred, list(tn[[i]] + fn[[i]]))
    }
    tpr <- tp[[1]] / max(tp[[1]])
    fpr <- fp[[1]] / max(fp[[1]])
    tnr <- tn[[1]] / max(tn[[1]])
    fnr <- fn[[1]] / max(fn[[1]])
    sensitivity <- tp[[1]] / (tp[[1]] + fn[[1]])
    specificity <- tn[[1]] / (fp[[1]] + tn[[1]])
    precision <- tp[[1]] / (tp[[1]] + fp[[1]])
    accuracy <- (tp[[1]] + tn[[1]]) / (tp[[1]] + fp[[1]] + fn[[1]] + tn[[1]])
    error_rate <- 1 - accuracy

    return(new("prediction", predictions = predictions, labels = labels,
        cutoffs = cutoffs, fp = fp, tp = tp, fn = fn, tn = tn,
        n.pos = n.pos, n.neg = n.neg, n.pos.pred = n.pos.pred,
        n.neg.pred = n.neg.pred, tpr = tpr, fpr = fpr, tnr = tnr, fnr = fnr,
        sensitivity = sensitivity, specificity = specificity, precision =
precision,
        recall = sensitivity, accuracy = accuracy, error_rate = error_rate))
}
```

Question 2

(20 points) For the following data set, you will compute the true positive rate, false positive rate, and accuracy (with the function you wrote for Prob. 1) using every predicted output as a possible threshold.

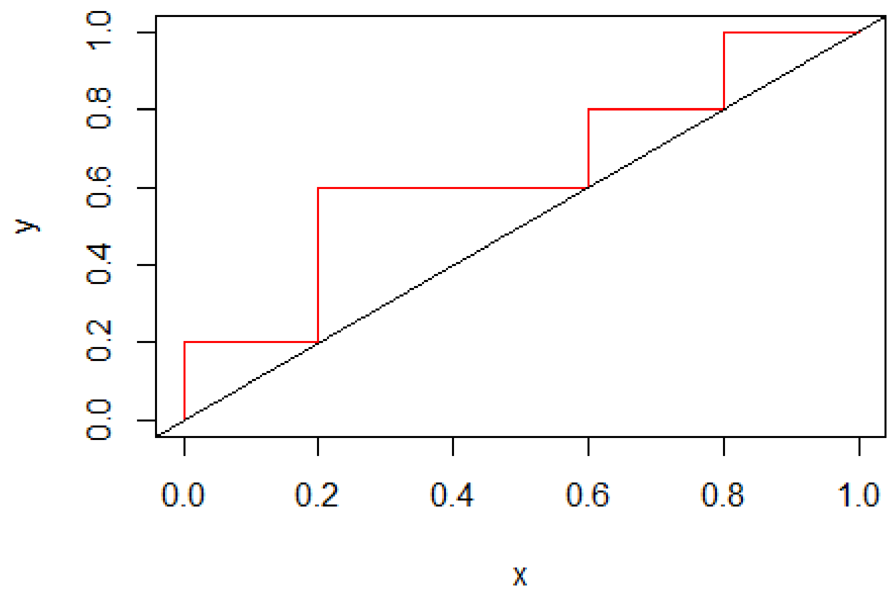
Sample	Y_{true}	Y_{Pred}
1	1	0.98
2	0	0.92
3	1	0.85
4	1	0.77
5	0	0.71
6	0	0.64
7	1	0.50
8	0	0.39
9	1	0.34
10	0	0.31

```
# Assigning the given data to varibles
y_true <- c(1, 0, 1, 1, 0 ,0, 1, 0, 1, 0)
y_pred <- c(0.98, 0.92, 0.85, 0.77, 0.71, 0.64, 0.50, 0.39, 0.34, 0.31)

# Calling the prediction function
pred <- predictions(y_pred, y_true)

## Found more than one class "prediction" in cache; using the first, from
namespace 'ROCR'

# Creating dataframe for the required result
result_q2 <- data.frame(true.positve.rate = pred@tpr[-1], false.positive.rate
= pred@fpr[-1],
                        accuracy = pred@accuracy[-1])

# Printing the reults
result_q2

##    true.positve.rate false.positive.rate accuracy
## 1                0.2                 0.0      0.6
## 2                0.2                 0.2      0.5
## 3                0.4                 0.2      0.6
## 4                0.6                 0.2      0.7
## 5                0.6                 0.4      0.6
## 6                0.6                 0.6      0.5
## 7                0.8                 0.6      0.6
## 8                    0.8                   0.8      0.5
## 9                    1.0                   0.8      0.6
## 10                   1.0                   1.0      0.5
```

Question 3

(10 points) Use the results from Prob. 2 to plot the ROC curve for the data. Note, plot this curve using the standard plotting tools rather than any special library/package available in R or Matlab.

```
# Setting (0, 0) coordinate up infront to
# make the startin point error corrections
x<- c(0, result_q2$false.positive.rate)
y <- c(0, result_q2$true.positve.rate)

# Plotting the ROC curve for the data
plot(x, y, type = "l", col = "red")
abline(a=0, b=1)
```



Question 4

(12 points) Data Mining Book, 8.14.

Suppose that we want to select between two prediction models, M_1 and M_2. We have performed 10 rounds of 10-fold cross-validation on each model, where the same data partitioning in round i is used for both M_1 and M_2. The error rates obtained for M_1 are 30.5, 32.2, 20.7, 20.6, 31.0, 41.0, 27.7, 26.0, 21.5, 26.0. The error rates for M_2 are 22.4, 14.5, 22.4, 19.6, 20.7, 20.4, 22.1, 19.4, 16.2, 35.0. Comment on whether one model is significantly better than the other considering a significance level of 1%.

```
paste0("Hypothesis testing can be done to see if there are significant
difference in error rates. Since same data partitioning is used for both M1
and M2 i.e same test set, the paired observation hypothesis testing to
compare means is
H0: me1 - me2 = 0
H1: me1 - me2 != 0 where me1 is the mean error of M1 and me2 is the mean
error of M2")

## [1] "Hypothesis testing can be done to see if there are significant
difference in error rates. Since same data partitioning is used for both M1
and M2 i.e same test set, the paired observation hypothesis testing to
compare means is\nH0: me1 - me2 = 0\nH1: me1 - me2 != 0 where me1 is the mean
error of M1 and me2 is the mean error of M2"

M1 = c(30.5, 32.2, 20.7, 20.6, 31.0, 41.0, 27.7, 26.0, 21.5, 26.0)
M2 = c(22.4, 14.5, 22.4, 19.6, 20.7, 20.4, 22.1, 19.4, 16.2, 35.0)

t.test(M1,M2,paired = TRUE,conf.level = 0.99)

##
##  Paired t-test
##
## data:  M1 and M2
## t = 2.3444, df = 9, p-value = 0.0437
## alternative hypothesis: true difference in means is not equal to 0
## 99 percent confidence interval:
##  -2.490986 15.390986
## sample estimates:
## mean of the differences
##                    6.45

paste0("Since p-value is 0.0437 which is greater than the significance level
of 0.01, we fail to reject the null hypothesis i.e the two models are not
significantly different.")

## [1] "Since p-value is 0.0437 which is greater than the significance level
of 0.01, we fail to reject the null hypothesis i.e the two models are not
significantly different."
```

Question 5

Suppose you are building a decision tree on a data set with three classes A;B;C. At the current position in the tree you have the following samples available:

$$N = \begin{pmatrix} A & 100 \\ B & 50 \\ C & 60 \end{pmatrix}$$

You are examining two ways to split the data. The first splits the data as,

$$N\ 1{,}1 \ = \begin{pmatrix} A & 62 \\ B & 8 \\ C & 0 \end{pmatrix}$$

$$N\ 1{,}2 \ = \begin{pmatrix} A & 38 \\ B & 42 \\ C & 60 \end{pmatrix}$$

The second splits the data as,

$$N\ 2{,}1 \ = \begin{pmatrix} A & 65 \\ B & 20 \\ C & 0 \end{pmatrix}$$

$$N\ 2{,}2 \ = \begin{pmatrix} A & 21 \\ B & 19 \\ C & 20 \end{pmatrix}$$

$$N\ 2{,}3 \ = \begin{pmatrix} A & 14 \\ B & 11 \\ C & 40 \end{pmatrix}$$

Question 5a

(a) (18 points) Compute the gain in GINI index for the two splits. Show the form of the calculations, not just the final numbers.

```
# Calculate GINI for overall data
GINI_N <- 1- (100/(100+50+60))^2 - (50/(100+50+60))^2 - (60/(100+50+60))^2
paste0("The overall GINI Index is ",GINI_N)

## [1] "The overall GINI Index is 0.634920634920635"

# Calculate GINI Index for first split
GINI_N_11 <- 1- (62/(62+8+0))^2 - (8/(62+8+0))^2 - (0)
paste0("The Gini index for the first split on first node is ",GINI_N_11)

## [1] "The Gini index for the first split on first node is
0.202448979591837"

GINI_N_12 <- 1- (38/(38+42+60))^2 - (42/(38+42+60))^2 - (60/(38+42+60))^2
paste0("The Gini index for the first split on second node is ",GINI_N_12)

## [1] "The Gini index for the first split on second node is
0.65265306122449"

#Calculate GINI Index for second split
GINI_N_21 <- 1-(65/(65+20+0))^2 - (20/(65+20+0))^2 - (0)
paste0("The Gini index for the second split on first node is ",GINI_N_21)

## [1] "The Gini index for the second split on first node is
0.359861591695502"

GINI_N_22 <- 1-(21/(21+19+20))^2 - (19/(21+19+20))^2 -(20/(21+19+20))^2
paste0("The Gini index for the second split on second node is ",GINI_N_22)
```

```
## [1] "The Gini index for the second split on second node is 
0.666111111111111"

GINI_N_23 <- 1-(14/(14+11+40))^2 - (11/(14+11+40))^2 -(40/(14+11+40))^2
paste0("The Gini index for the second split on third node is ",GINI_N_23)

## [1] "The Gini index for the second split on third node is 
0.546272189349112"

# Calculate Gain for first split
GINI_GAIN_FIRST <- GINI_N - (62+8+0)/210 * GINI_N_11 - (38+42+60)/210 * 
GINI_N_12
paste0("The gain from the gini for first split is ",GINI_GAIN_FIRST)

## [1] "The gain from the gini for first split is 0.13233560090703"

GINI_GAIN_SECOND <- GINI_N - (65+20+0)/210 * GINI_N_21 - (21+19+20)/210 * 
GINI_N_22 - (14+11+40)/210 * GINI_N_23
paste0("The gain from the gini for second split is ",GINI_GAIN_SECOND)

## [1] "The gain from the gini for second split is 0.129860662213603"
```

Question 5b

(4 points) Which node would be preferred to include next in the decision tree?

The gain from gini for first split is higher and hence first split will be considered to be included in the decision tree

Question 5c

(18 points) Compute the information gain (based on entropy) for the two splits. Show the form of the calculations, not just the final numbers.

```
Entropy_N <- - (100/210) * log(100/210) - (50/210) * log(50/210) - (60/210) * 
log(60/210)
paste0("The entropy for the root node is", Entropy_N)

## [1] "The entropy for the root node is1.05292256593869"

Entropy_N_11 <- - (62/70) * log(62/70) - (8/70) * log(8/70) - 0
paste0("The entropy for the first split first node is", Entropy_N_11)

## [1] "The entropy for the first split first node is0.355382896246011"

Entropy_N_12 <- - (38/140) * log(38/140) - (42/140) * log(42/140) - (60/140) 
* log(60/140)
paste0("The entropy for the first split second node is", Entropy_N_12)

## [1] "The entropy for the first split second node is1.07827762424623"

Entropy_N_21 <- -(65/85)*log(65/85) - (20/85)*log(20/85) - 0
paste0("The entropy for the second split first node is", Entropy_N_21)
```

```
## [1] "The entropy for the second split first node is0.545594573969184"

Entropy_N_22 <- -(21/60)*log(21/60) - (19/60)*log(19/60) - (20/60)*log(20/60)
paste0("The entropy for the second split second node is", Entropy_N_22)

## [1] "The entropy for the second split second node is1.09777860776487"

Entropy_N_23 <- -(14/65)*log(14/65) - (11/65)*log(11/65) - (40/65)*log(40/65)
paste0("The entropy for the second split third node is", Entropy_N_23)

## [1] "The entropy for the second split third node is0.930097596357896"

Entropy_GAIN_FIRST <- Entropy_N - (62+8+0)/210 * Entropy_N_11 -
(38+42+60)/210 * Entropy_N_12
paste0("The gain from entropy for first split",Entropy_GAIN_FIRST)

## [1] "The gain from entropy for first split0.21560985102587"

Entropy_GAIN_SECOND <- Entropy_N - (65+20+0)/210 * Entropy_N_21 -
(21+19+20)/210 * Entropy_N_22 - (14+11+40)/210 * Entropy_N_23
paste0("The gain from entropy for second split",Entropy_GAIN_SECOND)

## [1] "The gain from entropy for second split0.230548284907572"
```

Question 5d

(4 points) Which node would be preferred to include next in the decision tree?

The gain from entropy for second split is higher and hence second split will be considered to be included in the decision tree

Question 6

Classification of Age based on Social Media Usage

For this problem, you want to classify the age of an individual ("High School" or "Adult") basic on their social media app usage. The data was collected via a survey, with respondents consisting of the Women in Computing Sciences Summer Youth Program participants and female faculty at Michigan Tech. The data is available at syp-s16-data.csv. There are 60 responses with 26 Adults and 34 HS respondents.

Question 6a

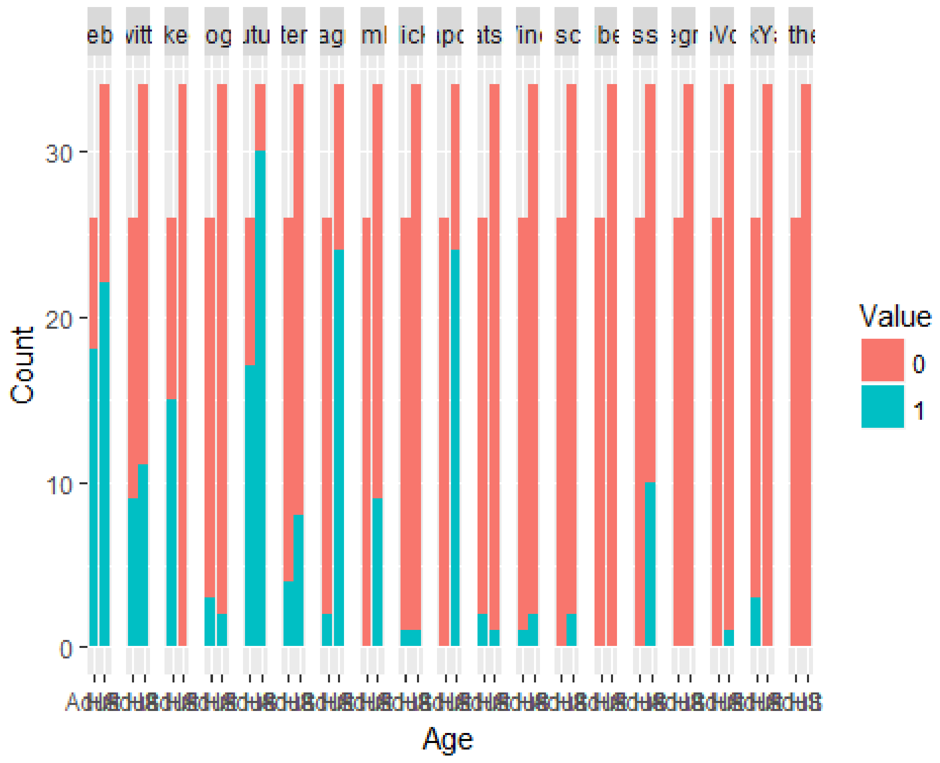
(12 points) Create a small multiples plot showing the number of respondents who use or do not use each app grouped by age. Consider making grouped bar plots.

```
# Importing data
syp_16_data <-  read.csv('data/syp-16-data.csv')
head(syp_16_data)

##   Facebook Twitter LinkedIn Google. Youtube Pinterest Instagram Tumblr
## 1        1       1        1       0       1         0         0      0
## 2        1       0        0       0       1         0         0      0
## 3        1       1        1       0       1         0         0      0
## 4        0       0        0       0       0         0         0      0
## 5        1       1        1       0       1         0         0      0
## 6        1       0        1       0       1         1         1      0
##   Flickr Snapchat WhatsApp Vine Periscope Viber KikMessenger Telegram
## 1      0        0        0    0         0     0            0        0
## 2      0        0        0    0         0     0            0        0
## 3      0        0        0    0         0     0            0        0
## 4      0        0        0    0         0     0            0        0
## 5      0        0        1    0         0     0            0        0
## 6      0        0        0    0         0     0            0        0
##   ooVoo YikYak Other Adult
## 1     0      0     0 Adult
## 2     0      0     0 Adult
## 3     0      0     0 Adult
## 4     0      0     0 Adult
## 5     0      0     0 Adult
## 6     0      0     0 Adult
```

```
# Converting data from wide to long format
data1 <- gather(data = syp_16_data,key = Network,value =
vals,c(Facebook,Twitter,LinkedIn,Google.,Youtube,Pinterest,Instagram,Tumblr,F
lickr,Snapchat,WhatsApp,Vine,Periscope,Viber,KikMessenger,Telegram,ooVoo,YikY
ak,Other),factor_key = TRUE)
# Finding the frequency
data1 <- data.frame(table(data1$Adult,data1$vals,data1$Network))
colnames(data1) <- c("Age","Value","Network","Count")
head(data1)
```

```
##     Age Value  Network Count
## 1 Adult     0 Facebook     8
## 2    HS     0 Facebook    12
## 3 Adult     1 Facebook    18
## 4    HS     1 Facebook    22
## 5 Adult     0  Twitter    17
## 6    HS     0  Twitter    23
```

```
# Creating grouped barplots
ggplot(data = data1,aes(x = Age,y = Count))+geom_bar(aes(fill = Value),stat =
"identity",position = "stack")+facet_grid(.~Network)
```



Question 6b

(4 points) From the figures in (a), which of the apps would you expect to find at the root of a deicision tree? that is, which app (variable) would be best to separate the two classes of responses?

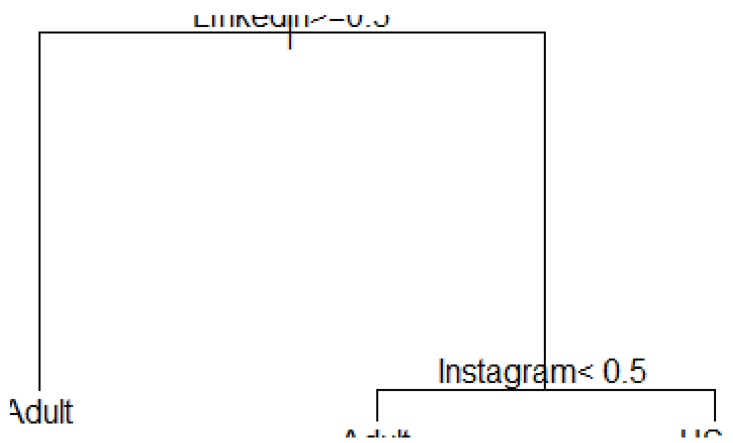
The intuition is Facebook which could separate the two classes (Adults and HS).

Question 6c

(10 points) Construct a decision tree using this data set. Does the variable at the root of the tree match the intuition from part (b)?

```r
# Splitting the dataset as 75/25
set.seed(123)
split <- sample.split(syp_16_data$Adult,SplitRatio = 3/4)
training_set <- subset(syp_16_data,split == TRUE)
test_set <- subset(syp_16_data,split == FALSE)

# Fitting decision tree to training set
dtree <- rpart(formula = Adult~.,data = training_set)

# Predicting the output for test set
y_pred <- predict(object = dtree,newdata = test_set,type = 'class')

# Creating confusion matrix
```

```
cm <- confusionMatrix(test_set[,20],y_pred)
cm

## Confusion Matrix and Statistics
##
##           Reference
## Prediction Adult HS
##      Adult     6  0
##      HS        3  5
##
##                Accuracy : 0.7857
##                  95% CI : (0.492, 0.9534)
##     No Information Rate : 0.6429
##     P-Value [Acc > NIR] : 0.2044
##
##                   Kappa : 0.5882
##  Mcnemar's Test P-Value : 0.2482
##
##             Sensitivity : 0.6667
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.6250
##              Prevalence : 0.6429
##          Detection Rate : 0.4286
##    Detection Prevalence : 0.4286
##       Balanced Accuracy : 0.8333
##
##        'Positive' Class : Adult
##

# Plotting Decision tree
plot(dtree)
text(dtree)
```



The intuition was facebook but the variable at the root of the tree is LinkedIn.

Question 7

Classification of Spam: Trees

For this problem, you will work to classify e-mail messages as spam or not. The data set to be used is given with the project (note, this is not the same spam data set that is available from UCI ML repository).

Question 7a

Load in the spam data. You should not include the following columns in the classification task: isuid, id, domain, spampct, category, and cappct.

```
spam <- read.csv("data/spam.csv")
spam <- data.frame(spam[,-c(1,2,7,11,19,20)])
head(spam)

##   day.of.week time.of.day size.kb box local digits name special credit
## 1         Thu           0       7  no    no      0 name       1     no
## 2         Thu           0       2  no    no      0 name       5     no
## 3         Thu          14       3  no   yes      0 name       2     no
## 4         Thu           3       3 yes    no      0 name       0     no
## 5         Thu           3       4  no    no      0 name       2     no
## 6         Thu           4       4  no    no      0 name       1     no
##   sucker porn chain username large.text spam
## 1      no    no     no        no          no    no
## 2      no    no     no        no          no   yes
## 3      no    no    yes        no          no    no
## 4      no    no     no        no          no    no
## 5      no    no     no        no          no    no
## 6      no    no     no        no          no    no
```

Question 7b

(4 points) Split the data into a training and test set with an 80/20 split of the data.

```
# Splitting data set by 80/20
set.seed(123)
split <- sample.split(spam$spam,SplitRatio = 0.80)
training_set <- subset(spam,split == TRUE)
test_set <- subset(spam,split == FALSE)
```

Question 7c

(8 points) Construct a classification tree to predict spam on the training data.

For R users, consider using the rpart library. Try using the default parameters of the software package. Understand how the decision tree is represented in R.

For Matlab users, consider using the Statistics Toolbox. This toolbox has several functions available for machine learning methods including classification trees.

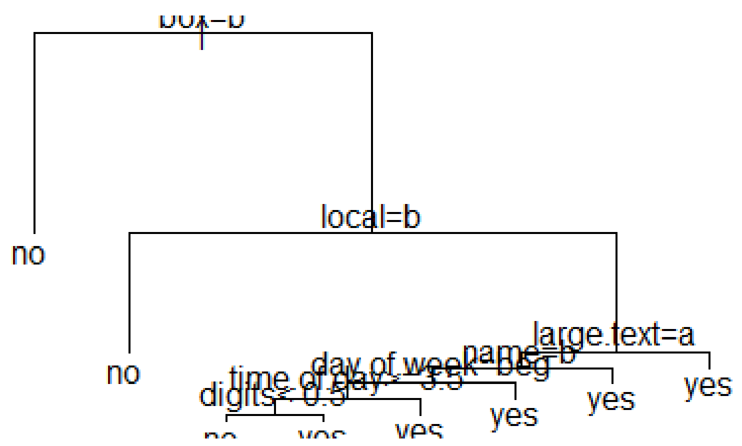
```
# Fitting the decision tree to training set
dtree <- rpart(formula = spam ~.,data = training_set)

# Predicting the output for test set
pred_test <- predict(object = dtree,newdata = test_set,type = 'class')

# Creating confusion matrix
cm <- confusionMatrix(test_set[,15],pred_test)
cm

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
##        no  276  16
##        yes  19 123
##
##                Accuracy : 0.9194
##                  95% CI : (0.8896, 0.9432)
##     No Information Rate : 0.6797
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8158
##  Mcnemar's Test P-Value : 0.7353
```

```
##
##             Sensitivity : 0.9356
##             Specificity : 0.8849
##          Pos Pred Value : 0.9452
##          Neg Pred Value : 0.8662
##              Prevalence : 0.6797
##          Detection Rate : 0.6359
##    Detection Prevalence : 0.6728
##       Balanced Accuracy : 0.9102
##
##        'Positive' Class : no
##
```

Question 7d

(6 points) Describe the tree that is constructed (print or plot the tree). How many terminal leaves does the tree have? What is the total number of nodes in the tree?

```
# Plotting Decision tree
plot(dtree)
text(dtree)
```

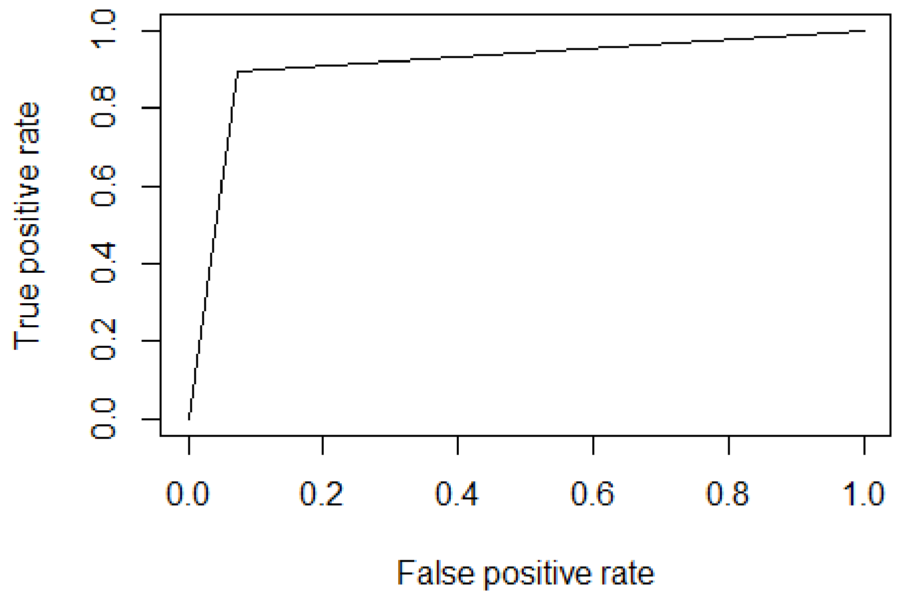


There are 8 terminal leaves and 7 total nodes.

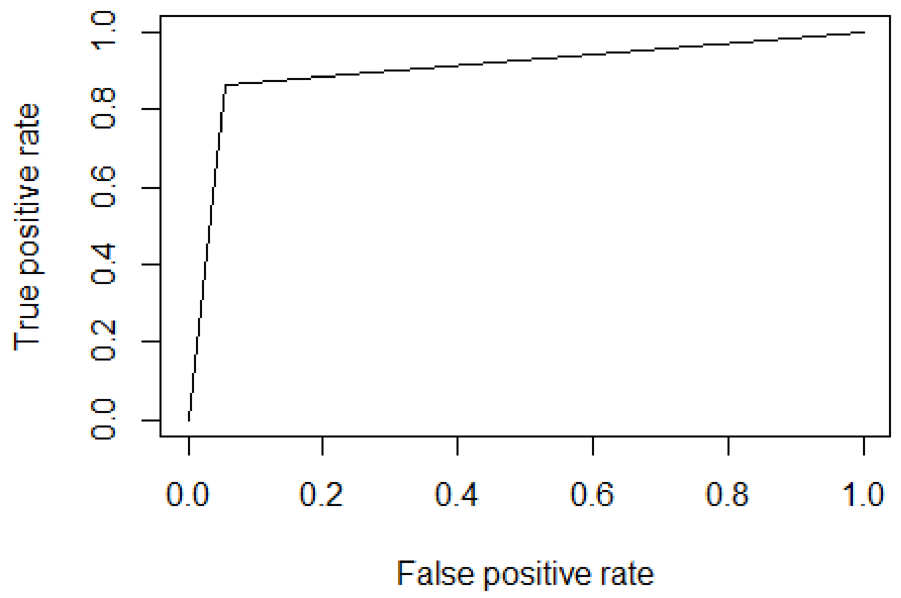
Question 7e

(6 points) Estimate the performance of the decision tree on the training set and the testing set. Report accuracy, error rate, and AUC using a threshold of 0.5.

```
# ROC Curve for training set
pred_training <- predict(object = dtree,newdata = training_set,type =
'class')
nb.pred <- prediction(as.numeric(pred_training),
as.numeric(training_set$spam))
nb.roc.perf <- performance(nb.pred, "tpr", "fpr")
plot(nb.roc.perf,xlab="False positive rate",ylab="True positive rate")
```



```
pROC::auc(as.numeric(training_set$spam),as.numeric(pred_training),thresh =
0.5)

## Area under the curve: 0.9104

# ROC Curve for test set
nb.pred <- prediction(as.numeric(pred_test), as.numeric(test_set$spam))
nb.roc.perf <- performance(nb.pred, "tpr", "fpr")
plot(nb.roc.perf,xlab="False positive rate",ylab="True positive rate")
```



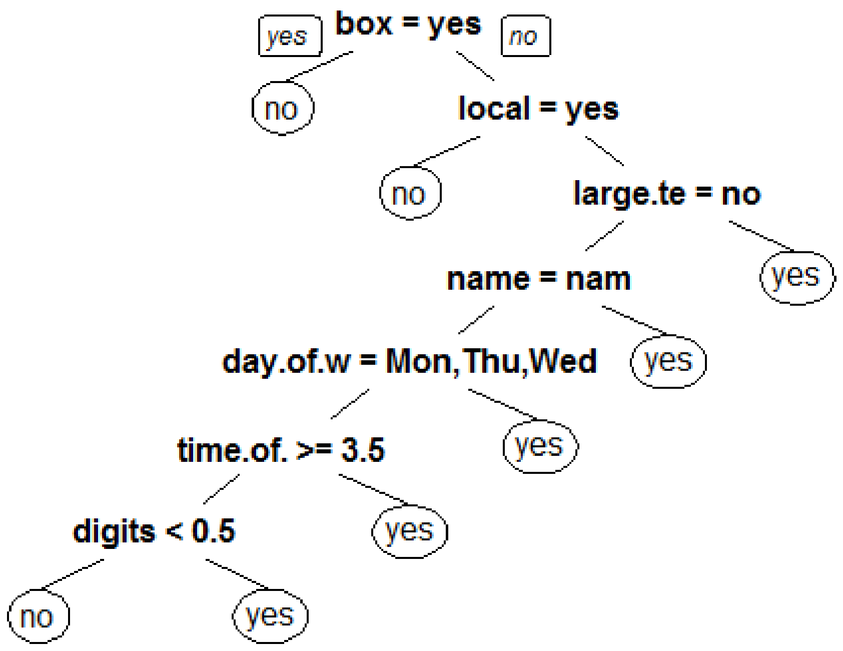
```
pROC::auc(as.numeric(test_set$spam),as.numeric(pred_test),thresh = 0.5)

## Area under the curve: 0.9057

# Confusion Matrix
cm <- confusionMatrix(test_set[,15],pred_test)
cm

## Confusion Matrix and Statistics
##
##             Reference
## Prediction   no yes
##        no   276  16
##        yes   19 123
##
##                Accuracy : 0.9194
##                  95% CI : (0.8896, 0.9432)
##     No Information Rate : 0.6797
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8158
##  Mcnemar's Test P-Value : 0.7353
##
##             Sensitivity : 0.9356
##             Specificity : 0.8849
##          Pos Pred Value : 0.9452
##          Neg Pred Value : 0.8662
##               Prevalence : 0.6797
##           Detection Rate : 0.6359
##     Detection Prevalence : 0.6728
##        Balanced Accuracy : 0.9102
##
##         'Positive' Class : no
##
```

The accuracy is 0.9194 and the error rate is 0.0806.

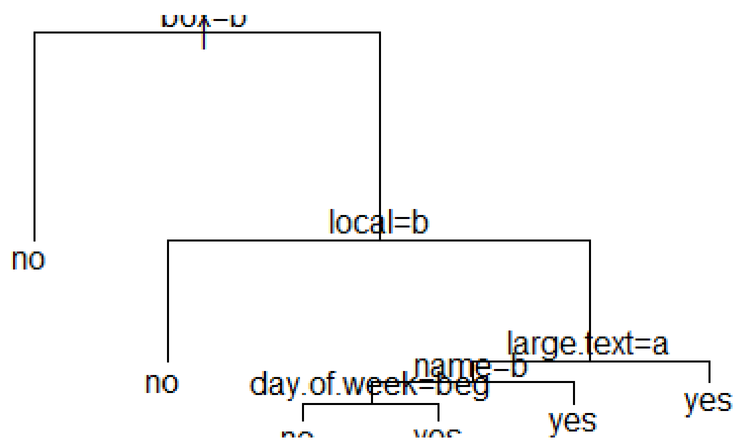
Question 7f

(8 points) Try pruning the tree, explore 2 other sized tree and report the classification performance in either case.

```
pruned <-
prune(dtree,cp=dtree$cptable[which.min(dtree$cptable[,"xerror"]),"CP"])
prp(pruned)
```



```
# First sized tree
dtree1 <- rpart(formula = spam ~.,data = training_set,control =
rpart.control(minsplit = 240))
pred_test1 <- predict(object = dtree1,newdata = test_set,type = 'class')
plot(dtree1)
text(dtree1)
```

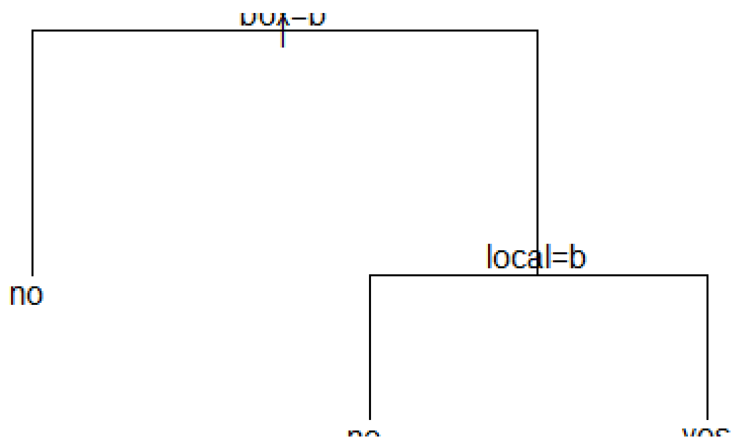


```
cm1 <- confusionMatrix(test_set[,15],pred_test1)
cm1

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
##        no  280  12
##        yes  23 119
##
##                Accuracy : 0.9194
##                  95% CI : (0.8896, 0.9432)
##     No Information Rate : 0.6982
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8131
##  Mcnemar's Test P-Value : 0.09097
##
##             Sensitivity : 0.9241
##             Specificity : 0.9084
##          Pos Pred Value : 0.9589
##          Neg Pred Value : 0.8380
##              Prevalence : 0.6982
##          Detection Rate : 0.6452
##    Detection Prevalence : 0.6728
##       Balanced Accuracy : 0.9162
##
```

```
##        'Positive' Class : no
##
```

The Accuracy remains 0.9194 but the prediction values in confusion matrix changed.

```
# Second sized tree
dtree2 <- rpart(formula = spam ~.,data = training_set,control =
rpart.control(minsplit = 360))
pred_test2 <- predict(object = dtree2,newdata = test_set,type = 'class')
plot(dtree2)
text(dtree2)
```



```
cm2 <- confusionMatrix(test_set[,15],pred_test2)
cm2

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
##        no  261  31
##        yes   6 136
##
##                Accuracy : 0.9147
##                  95% CI : (0.8844, 0.9393)
##     No Information Rate : 0.6152
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                   Kappa : 0.8147
##  Mcnemar's Test P-Value : 7.961e-05
##
##             Sensitivity : 0.9775
##             Specificity : 0.8144
##          Pos Pred Value : 0.8938
##          Neg Pred Value : 0.9577
##              Prevalence : 0.6152
##          Detection Rate : 0.6014
##    Detection Prevalence : 0.6728
##       Balanced Accuracy : 0.8959
##
##        'Positive' Class : no
##
```

The accuracy changes from 0.9194 to 0.9147

Question 8

Classification of Music Popularity

For this problem, you will work to classify a song's popularity. Specifically, you will develop methods to predict whether a song will make the Top10 of Billboard's Hot 100 Chart. The data set consists of song from the Top10 of Billboard's Hot 100 Chart from 1990-2010 along with a sampling of other songs that did not make the list1.

The variables included in the data set include several description of the song and artist (including song title and id numbers), the year the song was released. Additionally, several variables describe the song attributes: time signature, loudness, tempo, key, energy pitch, and timbre (measured of different sections of the song). The last variable is binary indicated whether the song was in the Top10 or not.

You will use the variables of the song attributes (excluding the variables involving confidence in that attribute, e.g., key confidence) to predict whether the song will be popular or not.

Question 8a

Load in the music data. You should not use the artist or song title and IDs in the prediction along with the confidence variables.

```r
# Loading the music data
music_real <- read.csv("data/music.csv")

# Removing IDs, artist, song title variables out of the data
music <-  music_real[, -c(2, 3, 4, 5, 6 , 9, 11)]

# Checking for NA values in the data
sapply(music, function(x) sum(is.na(x)))
```

```
##                      year timesignature_confidence                 loudness
##                         0                        0                        0
##          tempo_confidence           key_confidence                   energy
##                         0                        0                        0
##                     pitch            timbre_0_min             timbre_0_max
##                         0                        0                        0
##              timbre_1_min             timbre_1_max             timbre_2_min
##                         0                        0                        0
##              timbre_2_max             timbre_3_min             timbre_3_max
##                         0                        0                        0
##              timbre_4_min             timbre_4_max             timbre_5_min
##                         0                        0                        0
##              timbre_5_max             timbre_6_min             timbre_6_max
##                         0                        0                        0
##              timbre_7_min             timbre_7_max             timbre_8_min
##                         0                        0                        0
##              timbre_8_max             timbre_9_min             timbre_9_max
##                         0                        0                        0
##             timbre_10_min            timbre_10_max            timbre_11_min
##                         0                        0                        0
##             timbre_11_max                    Top10
##                         0                        0
```

Question 8b

Prepare the data for a 10-fold cross-validation. Ensure that each split of the data has a balanced distribution of class labels.

```
# Data preprocessing for 10 fold cross validation

# Setting the seed for reproduciablility
set.seed(294)

# NO of fold
k_fold <- 10

# Creating the folds variable in the data frame
# music$folds <- createFolds(music$Top10, k = n_fold, list = FALSE)
cv_index_list <- createFolds(music$Top10, k = k_fold, list = TRUE,
returnTrain = FALSE)

# Checking the class label proportions
for(i in 1:k_fold) {
    fold <- music$Top10[cv_index_list[[i]]]
    fold_length <- length(fold)
    class0_prop <- round(length(fold[fold == 0]) / fold_length, 2)
    class1_prop <- round(length(fold[fold == 1]) /  fold_length, 2)
    print(paste("Fold" , i , "length", fold_length, sep = " "))
    print(paste("Class Label(0)", round(class0_prop, 2), sep = " "))
    print(paste("Class Label(1)", round(class1_prop, 2), sep = " "))
}
```

```
## [1] "Fold 1 length 757"
## [1] "Class Label(0) 0.85"
## [1] "Class Label(1) 0.15"
## [1] "Fold 2 length 757"
## [1] "Class Label(0) 0.86"
## [1] "Class Label(1) 0.14"
## [1] "Fold 3 length 758"
## [1] "Class Label(0) 0.86"
## [1] "Class Label(1) 0.14"
## [1] "Fold 4 length 758"
## [1] "Class Label(0) 0.85"
## [1] "Class Label(1) 0.15"
## [1] "Fold 5 length 757"
## [1] "Class Label(0) 0.85"
## [1] "Class Label(1) 0.15"
## [1] "Fold 6 length 758"
## [1] "Class Label(0) 0.85"
## [1] "Class Label(1) 0.15"
## [1] "Fold 7 length 757"
## [1] "Class Label(0) 0.85"
## [1] "Class Label(1) 0.15"
## [1] "Fold 8 length 758"
## [1] "Class Label(0) 0.85"
## [1] "Class Label(1) 0.15"
## [1] "Fold 9 length 757"
## [1] "Class Label(0) 0.85"
## [1] "Class Label(1) 0.15"
## [1] "Fold 10 length 757"
## [1] "Class Label(0) 0.86"
## [1] "Class Label(1) 0.14"
```

Question 8c

(15 points) Use kNN to predict whether a song is a hit. Estimate the generalization performance over the 10-folds, calculate and report the accuracy, error, and AUC performance on the testing data. Show these results for three values of k = 1; 3; 5; 7; 9.

```
# Setting the k values for KNN
k_values <- c(1, 3, 5, 7, 9)

# Running the KNN for the mentioned K values
# with 10 - fold cross validation
# Accuracy, error, AUC are rounded off to two decimal points
for(k in k_values) {
    cat("\n")
    print(paste("***Knn with K =", k, "***", sep = " "))
    cat("\n")
    total_acc <- c()
    total_error_rate <- c()
    total_AUC <- c()
    for(fold in 1:k_fold) {
        train_set <- music[-cv_index_list[[1]], ]
        test_set <- music[cv_index_list[[1]], ]
        knn_predicted_values <- knn(train = train_set, test = test_set, cl =
train_set$Top10, k = k)
        accuracy <- round(Accuracy(knn_predicted_values, test_set$Top10), 2)
        error_rate <- 1 - accuracy
        AUC <- round(AUC(knn_predicted_values, test_set$Top10), 2)
        total_acc <- c(total_acc, accuracy)
        total_error_rate <- c(total_error_rate, error_rate)
        total_AUC <- c(total_AUC, AUC)
    }
    report <- data.frame(Fold = c(1:10), Accuracy = total_acc, Error_rate =
total_error_rate, AUC = total_AUC)
    print(report)
    cat("==========================================================\n")
    print(paste("Overall accuracy is", round(sum(total_acc) / k_fold, 2), sep
= " "))
    print(paste("Overall error is", round(sum(total_error_rate) / k_fold, 2),
sep =" "))
    print(paste("Overall area under the curve(AUC) is", round(sum(total_AUC)
/ k_fold, 2), sep = " "))
    cat("\n==========================================================\n")
}

##
## [1] "***Knn with K = 1 ***"
##
##    Fold Accuracy Error_rate  AUC
## 1     1     0.78       0.22 0.59
## 2     2     0.78       0.22 0.59
## 3     3     0.78       0.22 0.59
## 4     4     0.78       0.22 0.59
## 5     5     0.78       0.22 0.59
## 6     6     0.78       0.22 0.59
## 7     7     0.78       0.22 0.59
## 8     8     0.78       0.22 0.59
## 9     9     0.78       0.22 0.59
## 10   10     0.78       0.22 0.59
## ==========================================================
## [1] "Overall accuracy is 0.78"
## [1] "Overall error is 0.22"
## [1] "Overall area under the curve(AUC) is 0.59"
##
## ==========================================================
##
## [1] "***Knn with K = 3 ***"
```

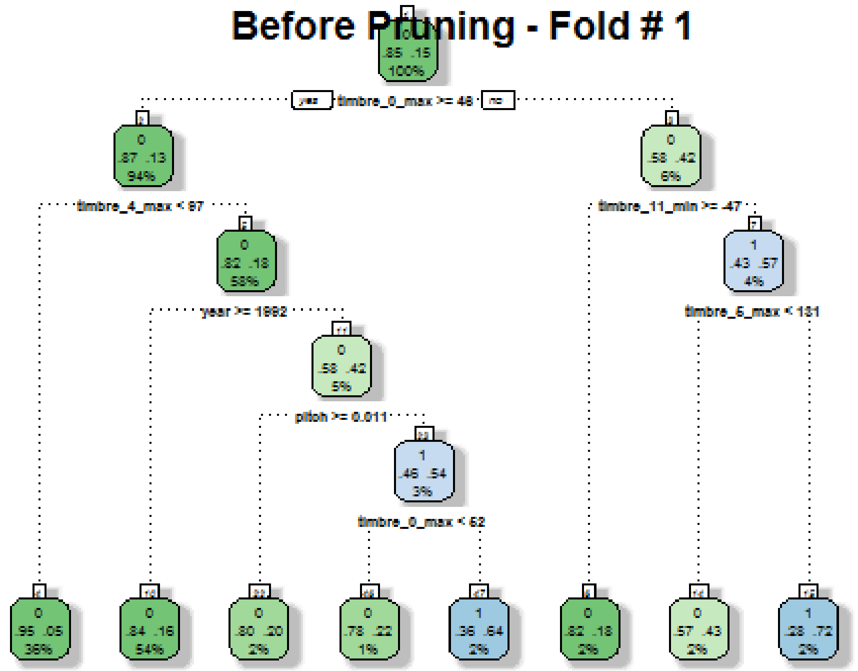
```
##
##    Fold Accuracy Error_rate  AUC
## 1     1     0.82       0.18 0.57
## 2     2     0.82       0.18 0.57
## 3     3     0.82       0.18 0.57
## 4     4     0.82       0.18 0.57
## 5     5     0.82       0.18 0.57
## 6     6     0.82       0.18 0.57
## 7     7     0.82       0.18 0.57
## 8     8     0.82       0.18 0.57
## 9     9     0.82       0.18 0.57
## 10   10     0.82       0.18 0.57
## ==========================================================
## [1] "Overall accuracy is 0.82"
## [1] "Overall error is 0.18"
## [1] "Overall area under the curve(AUC) is 0.57"
##
## ==========================================================
##
## [1] "***Knn with K = 5 ***"
##
##    Fold Accuracy Error_rate  AUC
## 1     1     0.84       0.16 0.56
## 2     2     0.84       0.16 0.56
## 3     3     0.84       0.16 0.56
## 4     4     0.84       0.16 0.56
## 5     5     0.84       0.16 0.56
## 6     6     0.84       0.16 0.56
## 7     7     0.84       0.16 0.56
## 8     8     0.84       0.16 0.56
## 9     9     0.84       0.16 0.56
## 10   10     0.84       0.16 0.56
## ==========================================================
## [1] "Overall accuracy is 0.84"
## [1] "Overall error is 0.16"
## [1] "Overall area under the curve(AUC) is 0.56"
##
## ==========================================================
##
## [1] "***Knn with K = 7 ***"
##
##    Fold Accuracy Error_rate  AUC
## 1     1     0.84       0.16 0.53
## 2     2     0.84       0.16 0.53
## 3     3     0.84       0.16 0.53
## 4     4     0.84       0.16 0.53
## 5     5     0.84       0.16 0.53
## 6     6     0.84       0.16 0.53
## 7     7     0.84       0.16 0.53
## 8     8     0.84       0.16 0.53
## 9      9      0.84       0.16 0.53
## 10    10      0.84       0.16 0.53
## ==========================================================
## [1] "Overall accuracy is 0.84"
## [1] "Overall error is 0.16"
## [1] "Overall area under the curve(AUC) is 0.53"
##
## ==========================================================
##
## [1] "***Knn with K = 9 ***"
##
##    Fold Accuracy Error_rate  AUC
## 1     1     0.84       0.16 0.52
## 2     2     0.84       0.16 0.52
## 3     3     0.84       0.16 0.52
## 4     4     0.84       0.16 0.52
## 5     5     0.84       0.16 0.52
## 6     6     0.84       0.16 0.52
## 7     7     0.84       0.16 0.52
## 8     8     0.84       0.16 0.52
## 9     9     0.84       0.16 0.52
## 10   10     0.84       0.16 0.52
## ==========================================================
## [1] "Overall accuracy is 0.84"
## [1] "Overall error is 0.16"
## [1] "Overall area under the curve(AUC) is 0.52"
##
## ==========================================================
```

Question 8d

(12 points) Use decision trees to predict whether a song is a hit. Estimate the generalization performance over the 10-folds, calculate and report the accuracy, error, and AUC performance on the testing data. Show the results for two different sized decision trees (consider different amounts of pruning).

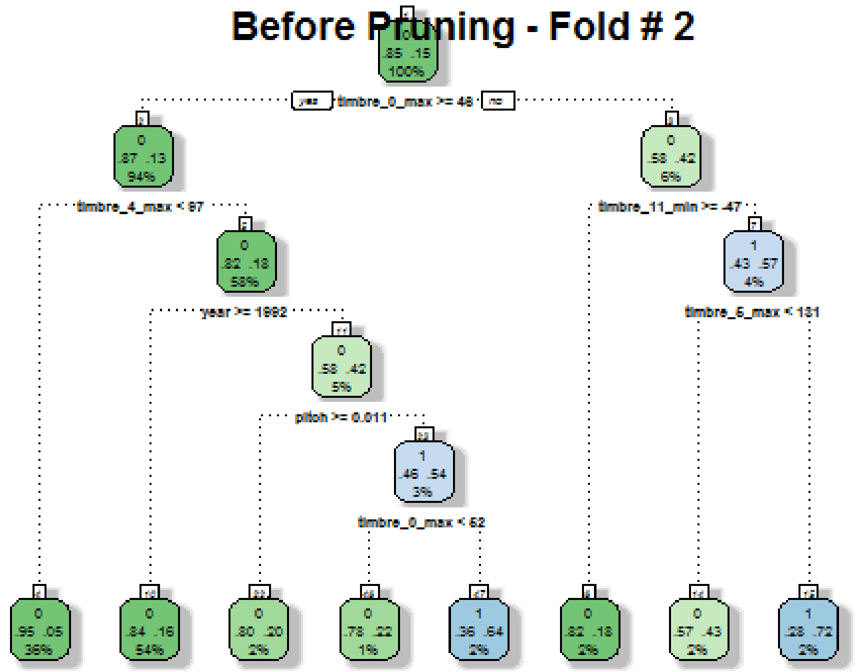
```
# Running the Decision Tree with 10 - fold cross validation
# Accuracy, error, AUC are rounded off to two decimal points
decisionTree <- function() {

    for(cp in c(0.01, 0.1)) {

        # Intailize variables
        total_acc <- c()
        total_error_rate <- c()
        total_AUC <- c()

        cat("\n")
        print(paste("Complexity Parameter used is", cp, sep = " "))
        cat("\n")
        for(fold in 1:k_fold) {
            train_set <- music[-cv_index_list[[k_fold]], ]
            test_set <- music[cv_index_list[[k_fold]], ]

            tree_fit <- rpart(Top10 ~ ., method="class", data = train_set,
minsplit = 4, minbucket = 2)
            fancyRpartPlot(tree_fit, main = paste("Before Pruning - Fold #",
fold, sep = " "))

            # # Find cp with minimum Cross validate error from rpart results
            # min_error_cp <-
tree_fit$cptable[which.min(tree_fit$cptable[,"xerror"]), "CP"]
            #
            # # pruning the tree with minimum cross validate error
            # ptree_fit <- prune(tree_fit, cp = min_error_cp)
            # plot(ptree_fit, main = paste("After Pruning - Fold #", fold,
sep = " "))

            # Calculating Accuracy. Error, AUC
            test_prediction <- predict(tree_fit, test_set, type = 'class')
            accuracy <- round(Accuracy(test_prediction, test_set$Top10), 2)
            error_rate <- 1 - accuracy
            AUC <- round(AUC(test_prediction, test_set$Top10), 2)
            total_acc <- c(total_acc, accuracy)
            total_error_rate <- c(total_error_rate, error_rate)
            total_AUC <- c(total_AUC, AUC)
        }

        report <- data.frame(Fold = c(1:10), Accuracy = total_acc, Error_rate
= total_error_rate, AUC = total_AUC)
        print(report)
        cat("==========================================================\n")
        print(paste("Overall accuracy is", round(sum(total_acc) / k_fold, 2),
sep = " "))
        print(paste("Overall error is", round(sum(total_error_rate) / k_fold,
2), sep =" "))
        print(paste("Overall area under the curve(AUC) is",
round(sum(total_AUC) / k_fold, 2), sep = " "))
        cat("\n==========================================================\n")
    }
}
decisionTree()

##
## [1] "Complexity Parameter used is 0.01"
```

Before Pruning - Fold # 1



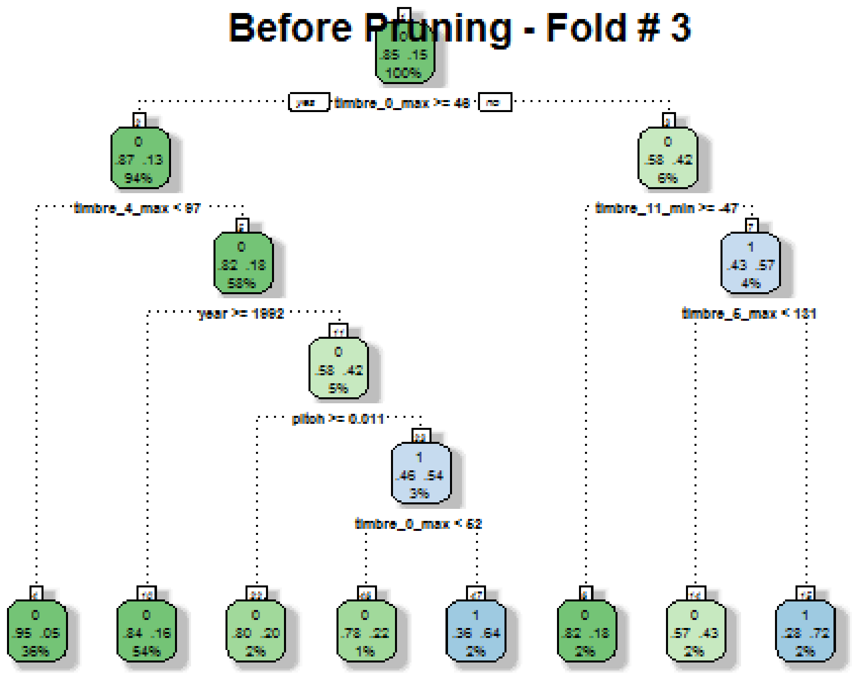
Rattle 2017-Apr-23 21:58:51 Raghavendran

Before Pruning - Fold # 2



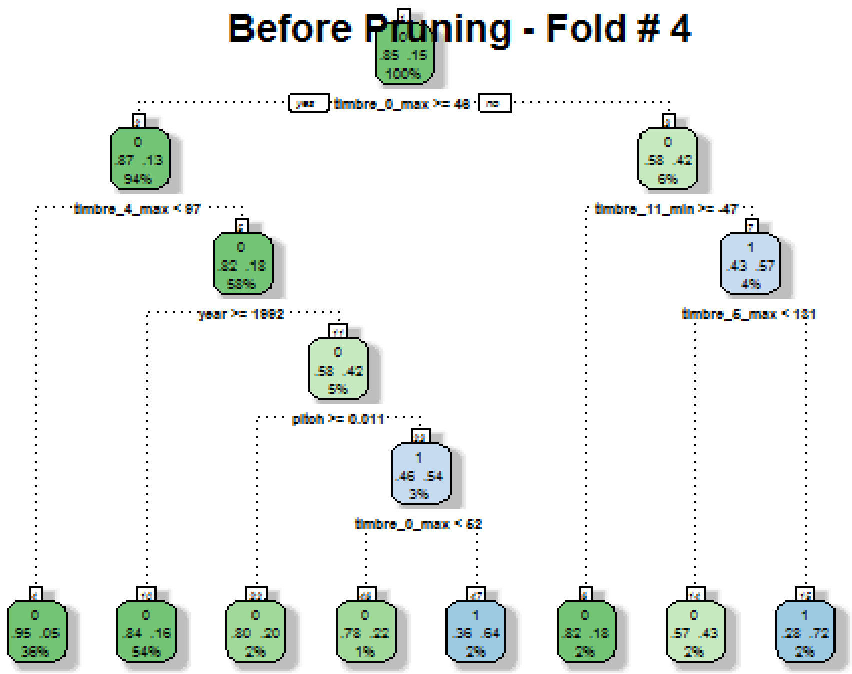
Rattle 2017-Apr-23 21:58:52 Raghavendran

Before Pruning - Fold # 3



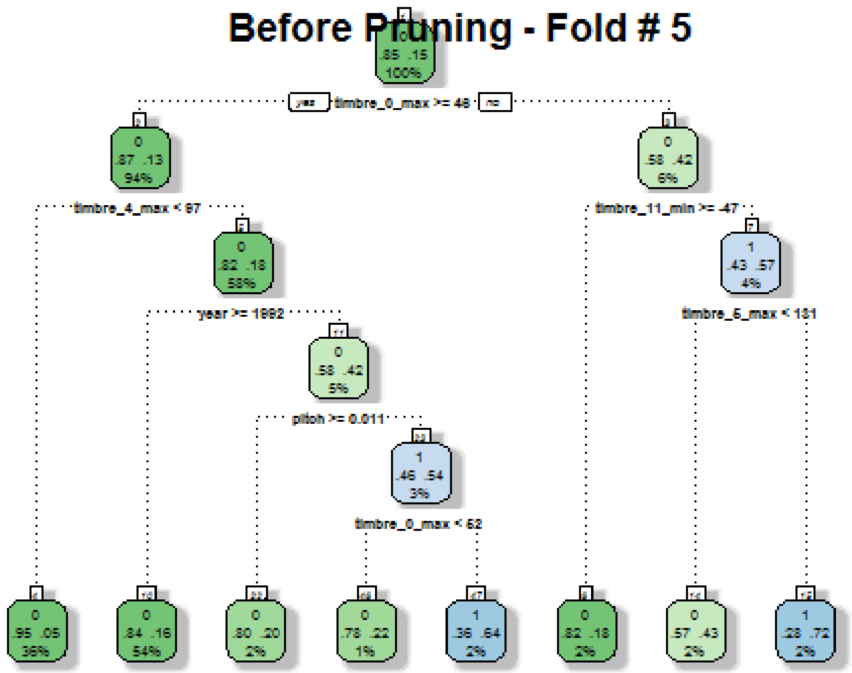
Rattle 2017-Apr-23 21:58:53 Raghavendran

Before Pruning - Fold # 4



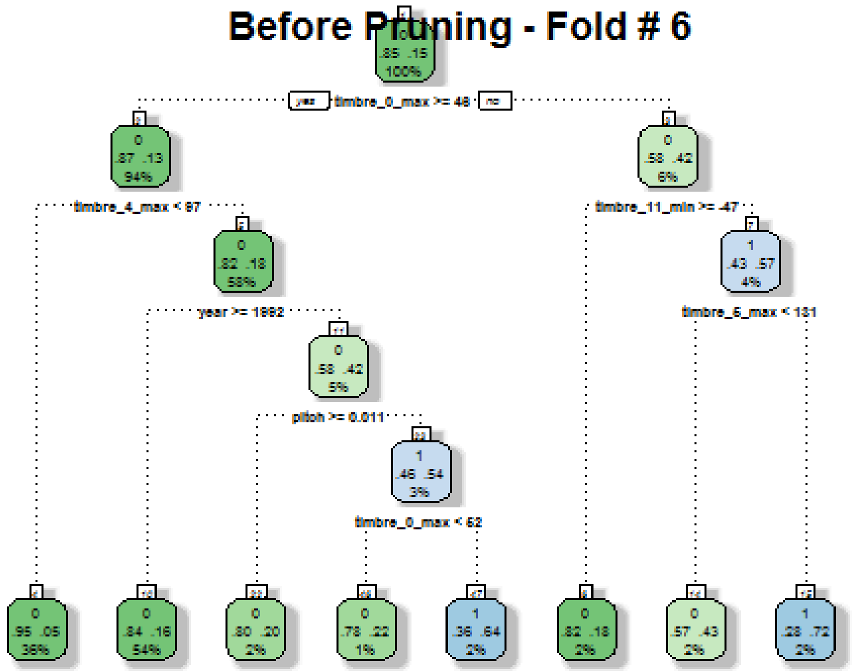
Rattle 2017-Apr-23 21:58:54 Raghavendran

Before Pruning - Fold # 5



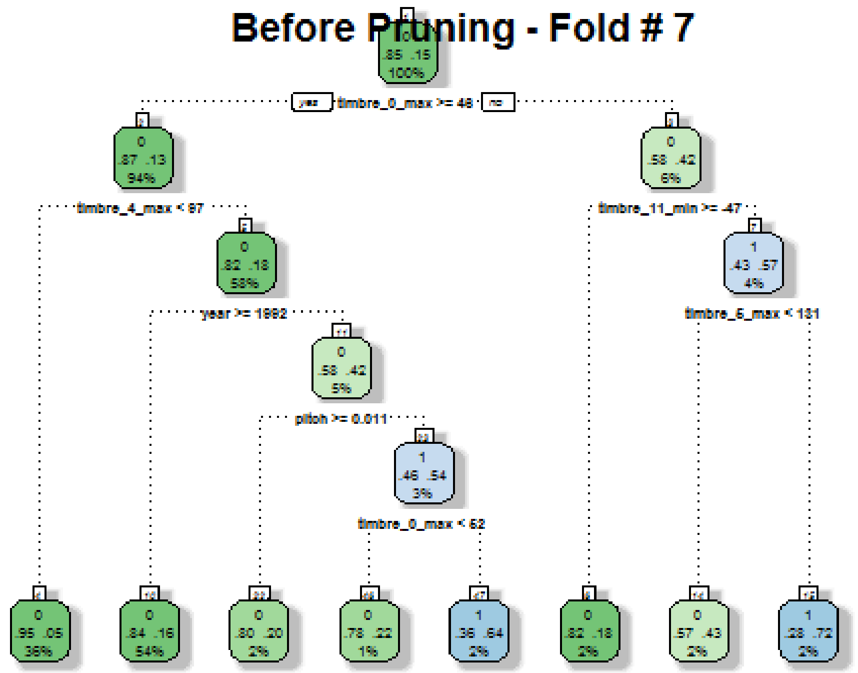
Rattle 2017-Apr-23 21:58:54 Raghavendran

Before Pruning - Fold # 6



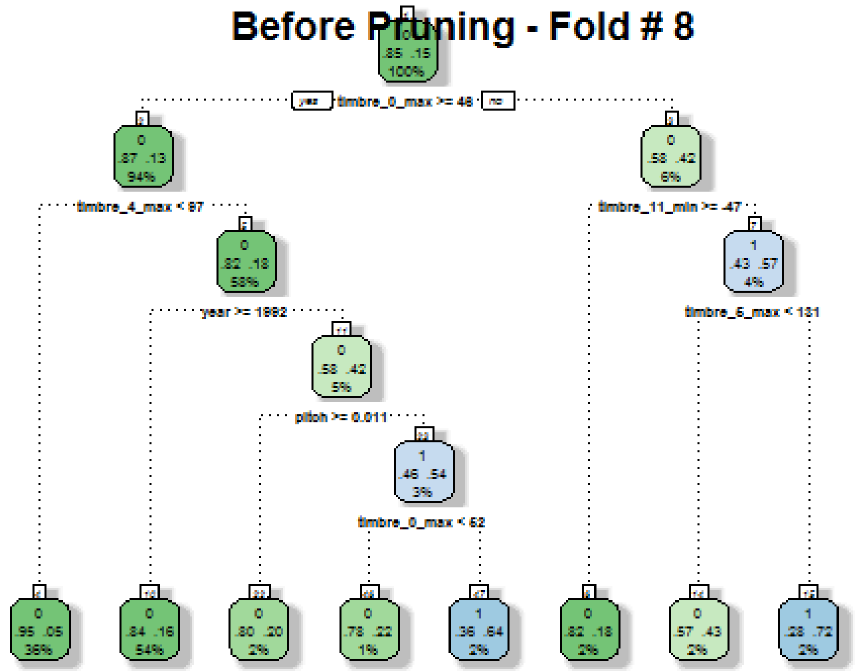
Rattle 2017-Apr-23 21:58:55 Raghavendran

Before Pruning - Fold # 7

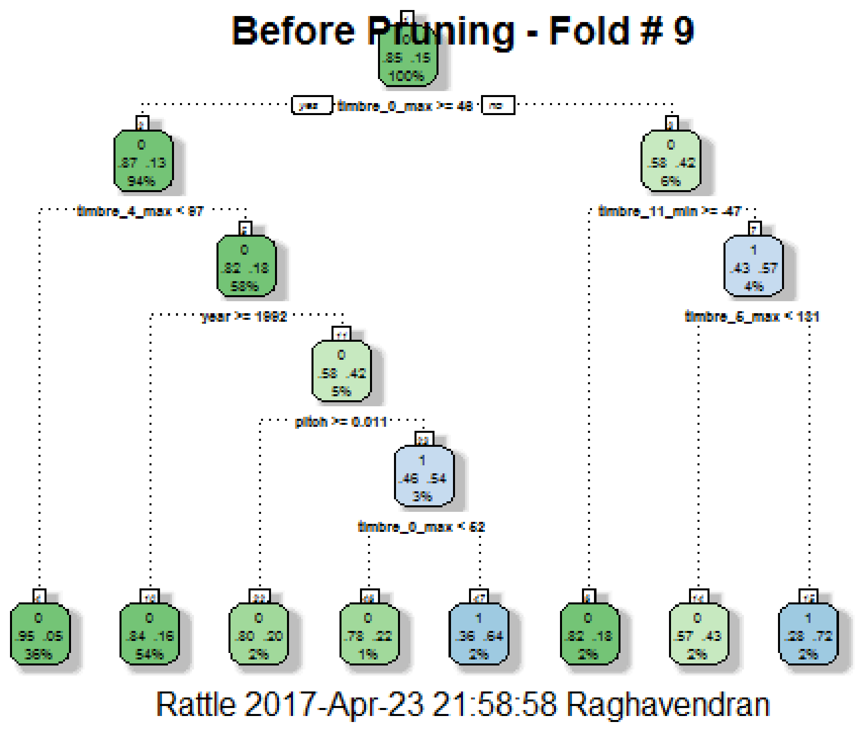


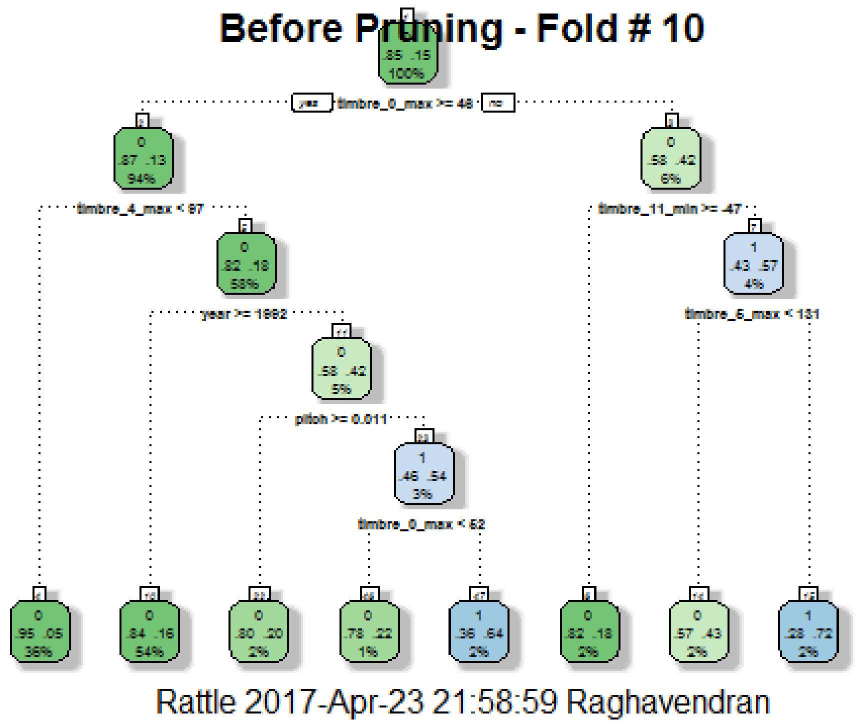
Rattle 2017-Apr-23 21:58:56 Raghavendran

Before Pruning - Fold # 8

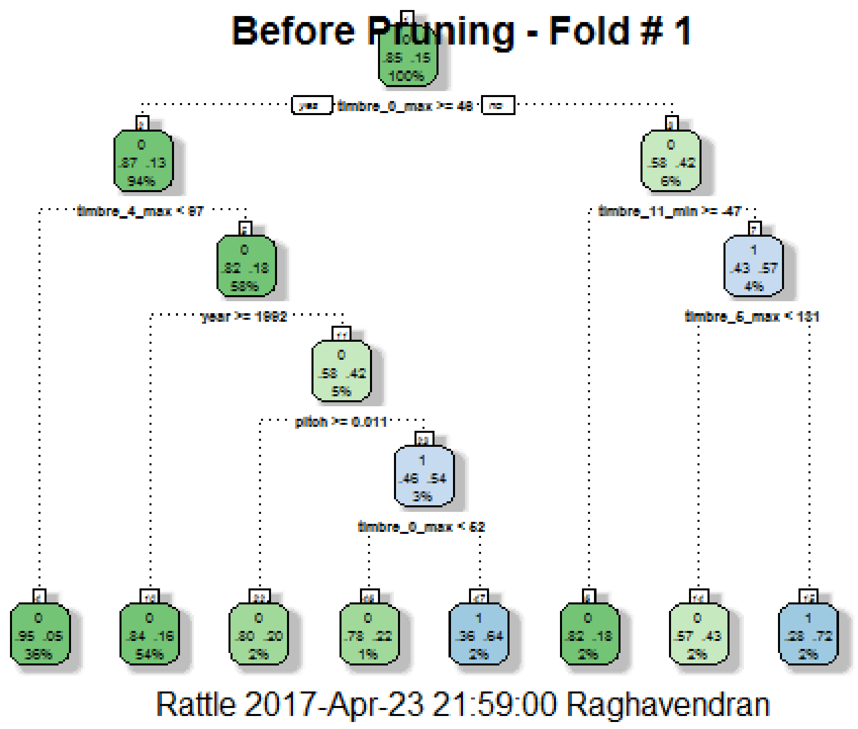


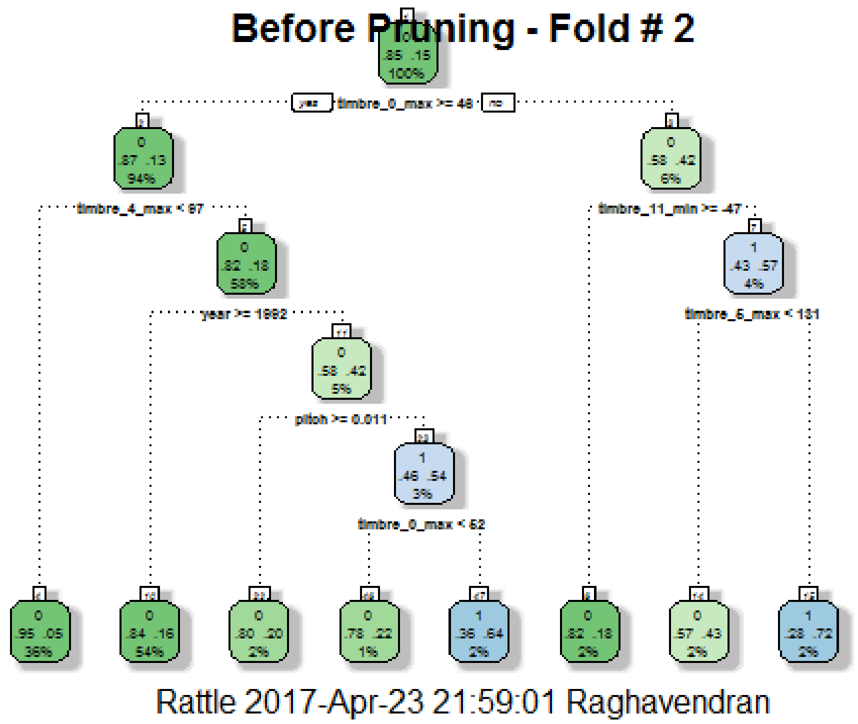
Rattle 2017-Apr-23 21:58:57 Raghavendran



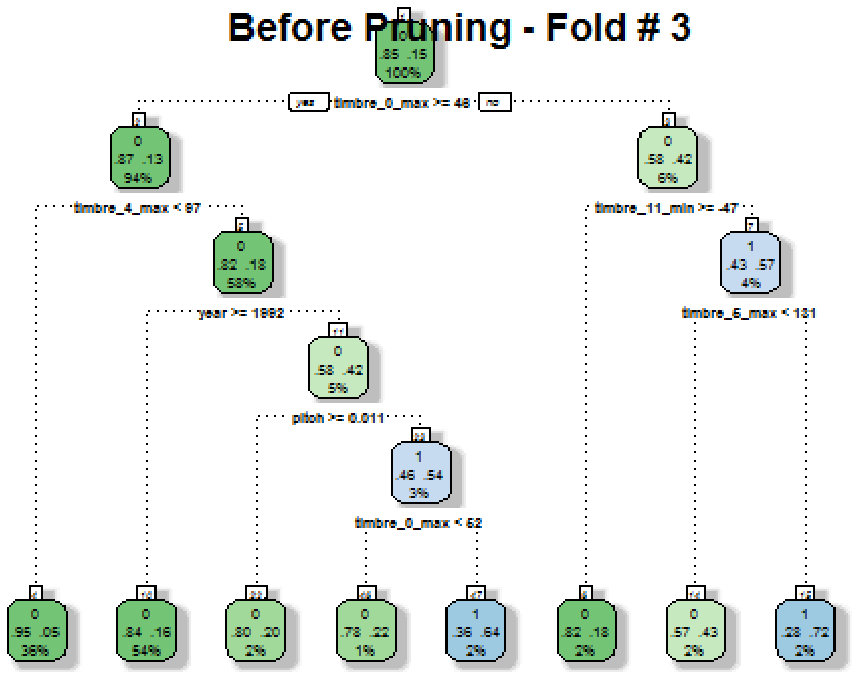


```
##     Fold Accuracy Error_rate  AUC
## 1      1     0.87       0.13 0.57
## 2      2     0.87       0.13 0.57
## 3      3     0.87       0.13 0.57
## 4     4     0.87       0.13 0.57
## 5     5     0.87       0.13 0.57
## 6     6     0.87       0.13 0.57
## 7     7     0.87       0.13 0.57
## 8     8     0.87       0.13 0.57
## 9     9     0.87       0.13 0.57
## 10   10     0.87       0.13 0.57
## ==========================================================
## [1] "Overall accuracy is 0.87"
## [1] "Overall error is 0.13"
## [1] "Overall area under the curve(AUC) is 0.57"
##
## ==========================================================
##
## [1] "Complexity Parameter used is 0.1"
```


Before Pruning - Fold # 1
0 .85 .15 100%
yes
timbre_0_max >= 48
no
0 .87 .13 94%
0 .58 .42 6%
timbre_4_max < 67
timbre_11_min >= -47
0 .82 .18 58%
1 .43 .57 4%
year >= 1992
timbre_5_max < 181
0 .58 .42 5%
pitch >= 0.011
1 .46 .54 3%
timbre_0_max < 52
0 .95 .05 36%
0 .84 .16 54%
0 .80 .20 2%
0 .78 .22 1%
1 .36 .64 2%
0 .82 .18 2%
0 .57 .43 2%
1 .28 .72 2%
Rattle 2017-Apr-23 21:59:00 Raghavendran

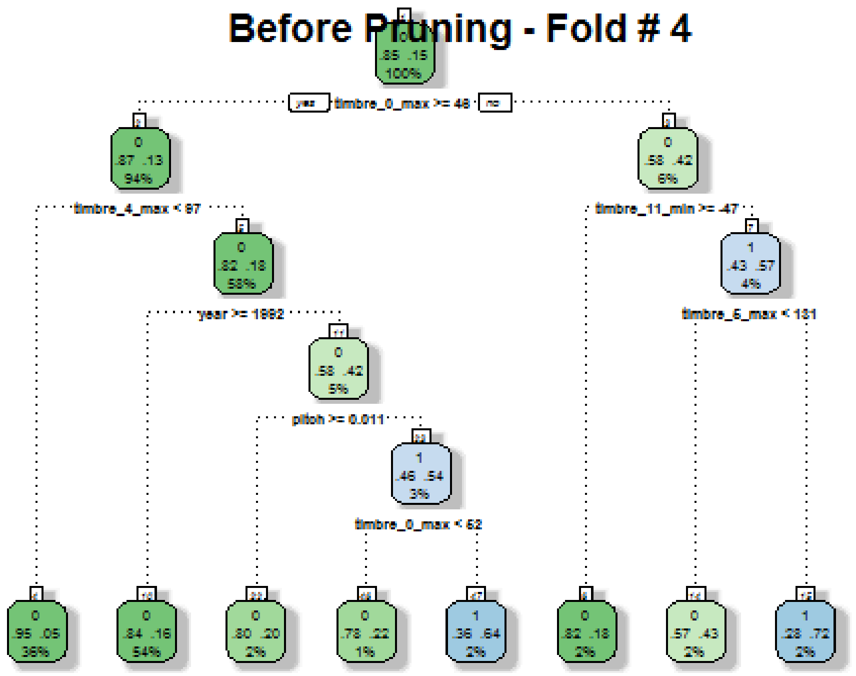

Before Pruning - Fold # 2
0 .85 .15 100%
yes
timbre_0_max >= 48
no
0 .87 .13 94%
0 .58 .42 6%
timbre_4_max < 67
timbre_11_min >= -47
0 .82 .18 58%
1 .43 .57 4%
year >= 1992
timbre_5_max < 181
0 .58 .42 5%
pitch >= 0.011
1 .46 .54 3%
timbre_0_max < 52
0 .95 .05 36%
0 .84 .16 54%
0 .80 .20 2%
0 .78 .22 1%
1 .36 .64 2%
0 .82 .18 2%
0 .57 .43 2%
1 .28 .72 2%
Rattle 2017-Apr-23 21:59:01 Raghavendran

Before Pruning - Fold # 3



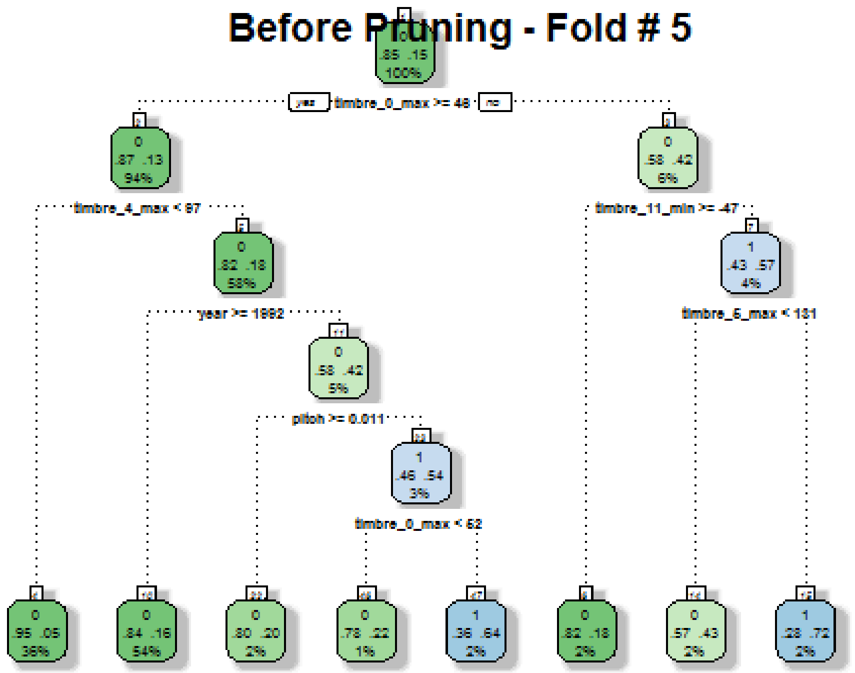
Rattle 2017-Apr-23 21:59:02 Raghavendran

Before Pruning - Fold # 4



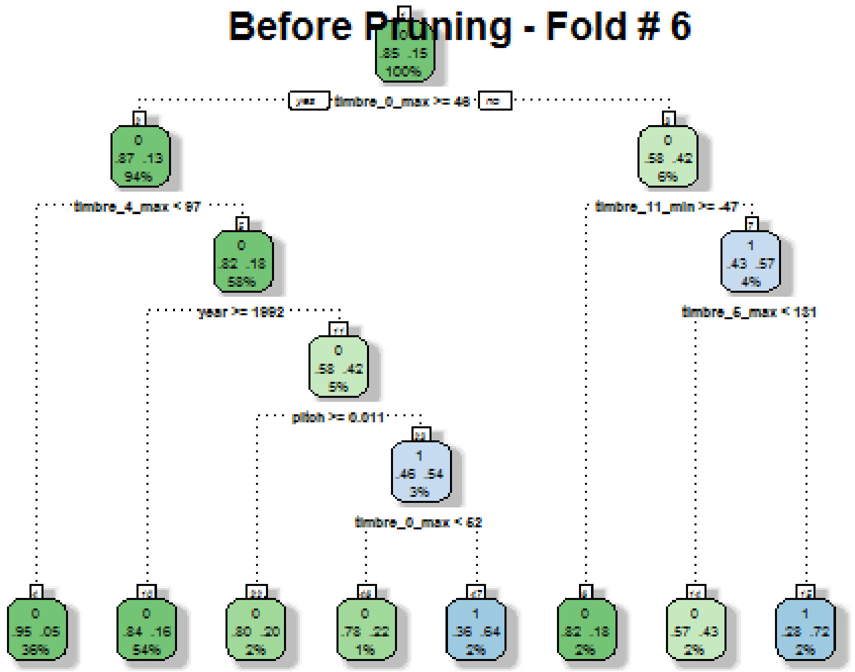
Rattle 2017-Apr-23 21:59:03 Raghavendran

Before Pruning - Fold # 5



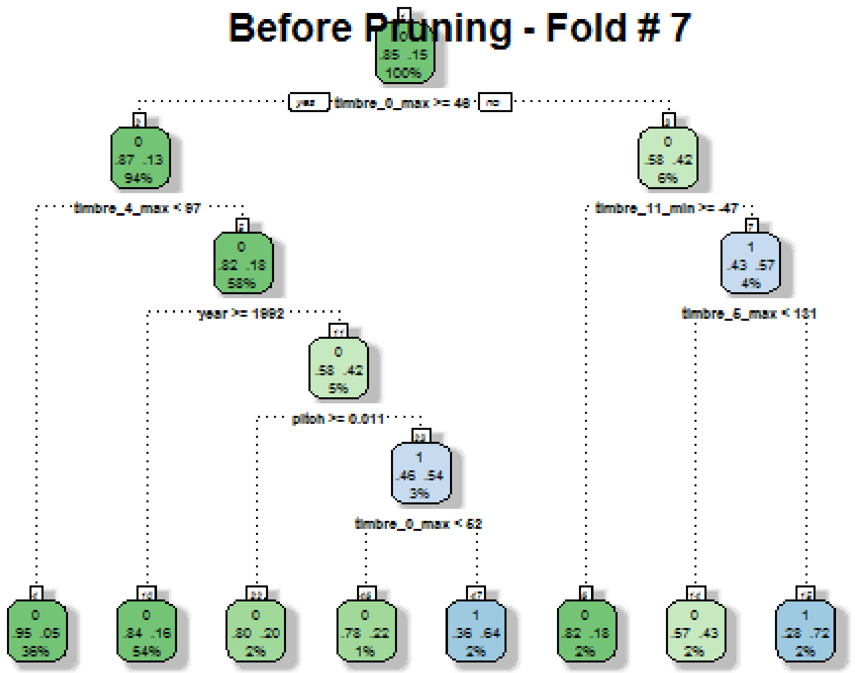
Rattle 2017-Apr-23 21:59:04 Raghavendran

Before Pruning - Fold # 6



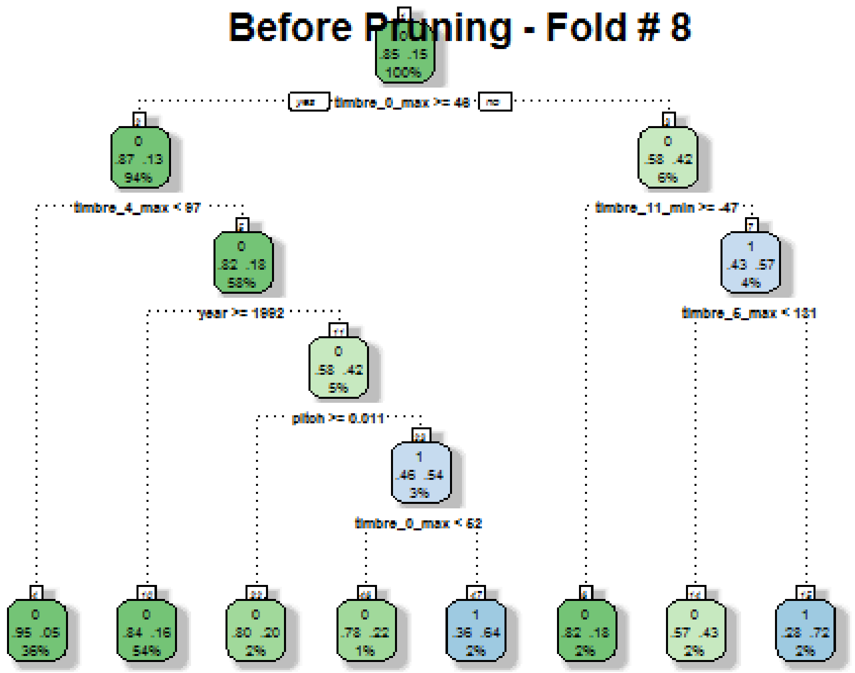
Rattle 2017-Apr-23 21:59:05 Raghavendran

Before Pruning - Fold # 7

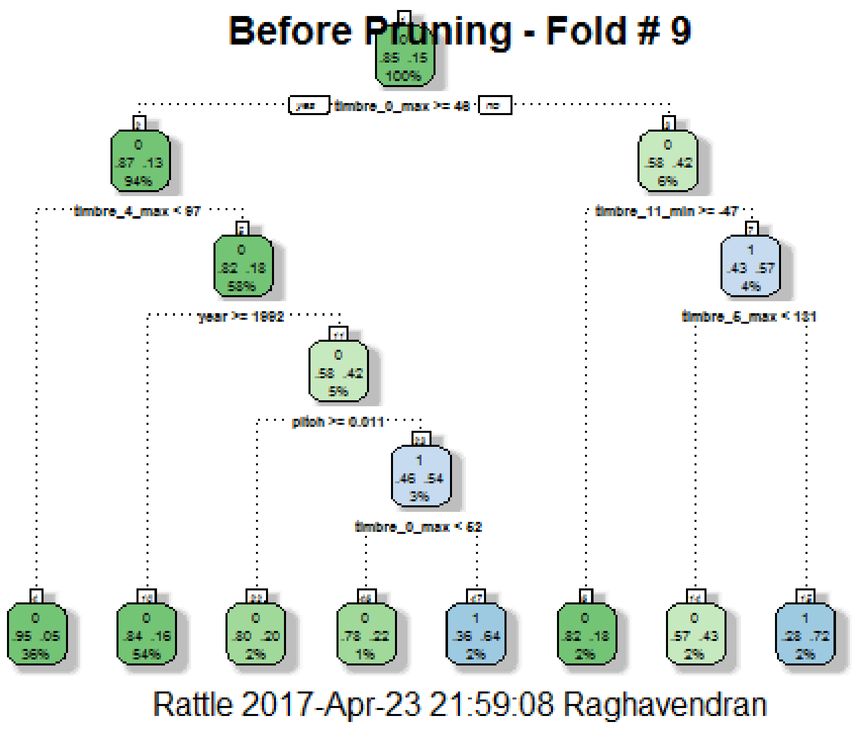


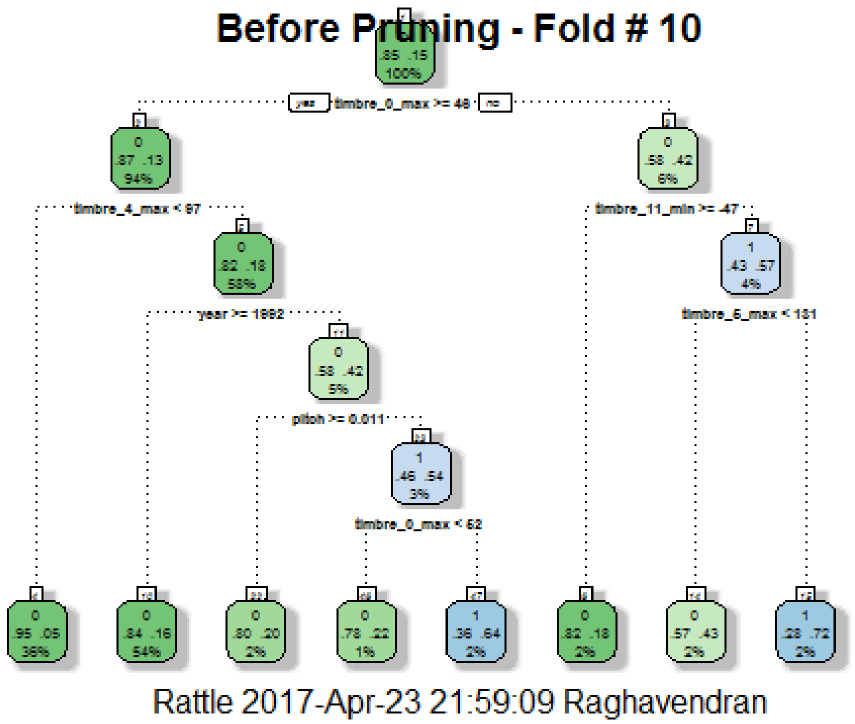
Rattle 2017-Apr-23 21:59:06 Raghavendran

Before Pruning - Fold # 8



Rattle 2017-Apr-23 21:59:07 Raghavendran





```
##     Fold Accuracy Error_rate  AUC
## 1      1     0.87       0.13 0.57
## 2      2     0.87       0.13 0.57
## 3      3     0.87       0.13 0.57
## 4     4     0.87       0.13 0.57
## 5     5     0.87       0.13 0.57
## 6     6     0.87       0.13 0.57
## 7     7     0.87       0.13 0.57
## 8     8     0.87       0.13 0.57
## 9     9     0.87       0.13 0.57
## 10   10     0.87       0.13 0.57
## ==========================================================
## [1] "Overall accuracy is 0.87"
## [1] "Overall error is 0.13"
## [1] "Overall area under the curve(AUC) is 0.57"
##
## ==========================================================
```

Question 8e

(15 points) Use a Naive Bayes classifier to predict whether a song is a hit. Calculate and report the accuracy, error, and AUC performance on the testing data.

```
# Intailize variables
total_acc <- c()
total_error_rate <- c()
total_AUC <- c()

# Running the Naive bayes with 10 - fold cross validation
# Accuracy, error, AUC are rounded off to two decimal points
naiveBayesModel <- function() {
    for(fold in 1:k_fold) {
        train_set <- music[-cv_index_list[[fold]], ]
        test_set <- music[cv_index_list[[fold]], ]

        # Fitting the Naive bayes model
        nb_fit <- naiveBayes(as.factor(Top10) ~ ., data = train_set)

        # Calculating Accuracy. Error, AUC
        test_prediction <- predict(nb_fit, test_set, type = "class")

        accuracy <- round(Accuracy(test_prediction, test_set$Top10), 2)
        error_rate <- 1 - accuracy
        AUC <- round(AUC(test_prediction, test_set$Top10), 2)
        total_acc <- c(total_acc, accuracy)
        total_error_rate <- c(total_error_rate, error_rate)
        total_AUC <- c(total_AUC, AUC)
    }

    report <- data.frame(Fold = c(1:10), Accuracy = total_acc, Error_rate =
total_error_rate, AUC = total_AUC)
    print(report)
    cat("==========================================================\n")
    print(paste("Overall accuracy is", round(sum(total_acc) / k_fold, 2), sep
= " "))
    print(paste("Overall error is", round(sum(total_error_rate) / k_fold, 2),
sep =" "))
    print(paste("Overall area under the curve(AUC) is", round(sum(total_AUC)
/ k_fold, 2), sep = " "))
    cat("\n==========================================================\n")
}
naiveBayesModel()
```

```
##    Fold Accuracy Error_rate  AUC
## 1     1     0.72       0.28 0.67
## 2     2     0.75       0.25 0.72
## 3     3     0.74       0.26 0.73
## 4     4     0.78       0.22 0.73
## 5     5     0.72       0.28 0.64
## 6     6     0.72       0.28 0.66
## 7     7     0.74       0.26 0.68
## 8     8     0.75       0.25 0.67
## 9     9     0.71       0.29 0.66
## 10   10     0.76       0.24 0.70
## ==========================================================
## [1] "Overall accuracy is 0.74"
## [1] "Overall error is 0.26"
## [1] "Overall area under the curve(AUC) is 0.69"
##
## ==========================================================
```

Question 8h

(5 points) Discuss whether the selection of the negative samples included in the data set may influence the results.

It is true that an increase in the ratio of positive to negative samples was found to greatly influence most of the evaluating parameters of Machine Learning. In a majority of cases, a training dataset consisting of a disprortionately high number of examples from one class will result in a classifier that is biased towards this majority class

Question 8i

(5) points (bonus)) Re-run the analysis in (c) using a nested cross-validation to determine the best value of k and to determine the best parameters of the SVM.

```
# Setting the k values for KNN
k_values <- c(1, 3, 5, 7, 9)

# Intializing the variables
max_accuracy <- 0
max_k <- 0

# Running the KNN for the mentioned K values
# with 10 - fold cross validation
```

```r
# Accuracy, error, AUC are rounded off to two decimal points
for(k in k_values) {
    total_acc <- c()
    total_error_rate <- c()
    total_AUC <- c()
    for(fold in 1:k_fold) {
        train_set <- music[-cv_index_list[[k_fold]], ]
        test_set <- music[cv_index_list[[k_fold]], ]
        knn_predicted_values <- knn(train = train_set, test = test_set, cl =
train_set$Top10, k = k)
        accuracy <- round(Accuracy(knn_predicted_values, test_set$Top10), 2)
        error_rate <- 1 - accuracy
        AUC <- round(AUC(knn_predicted_values, test_set$Top10), 2)
        total_acc <- c(total_acc, accuracy)
    }
    current_accuracy <- sum(total_acc) / k_fold
    if(current_accuracy > max_accuracy) {
        max_accuracy <- current_accuracy
        max_k <- k
    }

}
print(paste("The best value of k is", max_k, sep = " "))
```

```
## [1] "The best value of k is 7"
```

End of the assignment