Solution 3

STAT6306

Introduction

For this assignment, let's attempt to make a spam filter. Usually, this would involve a lot of text processing on a huge number of emails. In this case, someone has created a feature matrix for us. The feature matrix has rows given by individual emails and columns given by the number of each word or character that appears in that email, as well as three different numerical measures regarding capital letters (average length of consecutive capitals, longest sequence of consecutive capitals, and total number of capital letters).
The supervisor, Y, is given by the user supplied label marking that email as either spam (Y = 1) or not (Y = 0). Here is a function that may be useful for this assignment:

```
misClass =function(pred.class,true.class,produceOutput=FALSE){
  confusion.mat = table(pred.class,true.class)
  if(produceOutput){
    return(1-sum(diag(confusion.mat))/sum(confusion.mat))
  }
  else{
    print('misclass')
    print(1-sum(diag(confusion.mat))/sum(confusion.mat))
    print('confusion mat')
    print(confusion.mat)
  }
}
# this can be called using:
#     (assuming you make the appropriately named test predictions)
# misClass(Y.hat,Y_0)
```

Question 1

Read in the R data set spam.Rdata (note: when the object is an Rdata file, use load(spam.Rdata) and read the documentation file spambase.Documentation.

```
##SOLUTION
load("spam.Rdata")
```

What object(s) is/are loaded into memory? What objects are inside that object? How many emails do we have total? What features are in this data set?

SOLUTION

```
#Objects loaded in memory
ls(spam)
```

```
## [1] "column_labels"    "covariate_labels" "train"
## [4] "XdataF"           "Xmat"             "Y"
```

```
#number of emails:
length(spam$train)
```

```
## [1] 4601
```

```
#number of Features
ncol(spam$XdataF)
```

```
## [1] 57
```

```
#These are words/punctuation like:
head(names(spam$XdataF))
```

```
## [1] "make"    "address" "all"     "X3d"     "our"     "over"
```

Question 2

Let's make a training and test set.

```
train = spam$train
test  = !train
X     = spam$XdataF[train,]
X_0   = spam$XdataF[test,]
Y     = factor(spam$Y[train])
Y_0   = factor(spam$Y[test])
```

How many observations are in the training set (that is, what is n)? How many observations are in the test set?

SOLUTION

```
nrow(X)
```

```
## [1] 4355
```

```
nrow(X_0)
```

```
## [1] 246
```

Question 3

Read the document about using SQL and R (Rsql.Rmd and Rsql.pdf). Use the same ideas to create a SQL database that has 4 tables, one for each of the training/test feature/supervisor.

Once this database has been created, read in the information back into R from SQL. Include your code here for these steps.

```
## SOLUTION
write.csv(X,file='Xtrain.csv',row.names = FALSE)
write.csv(X_0,file='Xtest.csv',row.names = FALSE)
write.csv(Y,file='Ytrain.csv',row.names = FALSE)
write.csv(Y_0,file='Ytest.csv',row.names = FALSE)

#Get ride of objects from R's memory as you will be reloading them
#from the database you will be creating
rm(X);rm(X_0);rm(Y);rm(Y_0)
```

```
if(!require('sqldf')){install.packages('sqldf', repos = 'http://cran.us.r-project.org'); require('sqldf
```

```
## Loading required package: sqldf
```

```
## Loading required package: gsubfn
```

```
## Loading required package: proto
```

```
## Loading required package: RSQLite
```

```
db = dbConnect(SQLite(), dbname='spam.sqlite')

X = read.csv(file='Xtrain.csv')
dbWriteTable(conn = db, name = 'Xtrain',
             value = X, row.names = FALSE,
             col.names = TRUE,overwrite=TRUE)
rm(X)
#print out first few rows/columns
dbReadTable(db,'Xtrain')[1:5,1:5]
```

```
##   make address  all X3d  our
## 1 0.00    0.64 0.64   0 0.32
## 2 0.21    0.28 0.50   0 0.14
## 3 0.06    0.00 0.71   0 1.23
## 4 0.00    0.00 0.00   0 0.63
## 5 0.00    0.00 0.00   0 0.63
```

```
X              = dbReadTable(db,'Xtrain')

X_0 = read.csv(file='Xtest.csv')
dbWriteTable(conn = db, name = 'Xtest',
             value = X_0, row.names = FALSE,
             col.names = TRUE,overwrite=TRUE)
#print out first few rows/columns
dbReadTable(db,'Xtest')[1:5,1:5]
```

```
##   make address  all X3d  our
## 1 0.00    0.00 0.00   0 1.85
## 2 0.00    0.00 0.00   0 1.88
## 3 0.00    0.00 0.00   0 0.00
## 4 0.05    0.07 0.10   0 0.76
## 5 0.00    0.45 0.45   0 0.45
```

```
X_0            = dbReadTable(db,'Xtest')
feature_labels = names(X)

Y = read.csv(file='Ytrain.csv')
dbWriteTable(conn = db, name = 'Ytrain',
             value = Y, row.names = FALSE,
             col.names = FALSE,overwrite=TRUE)
Y = as.factor(dbReadTable(db,'Ytrain')$x)

Ytest = read.csv(file='Ytest.csv')
dbWriteTable(conn = db, name = 'Ytest',
             value = Ytest, row.names = FALSE,
             col.names = FALSE,overwrite=TRUE)
Y_0 = as.factor(dbReadTable(db,'Ytest')$x)
```

```
dbDisconnect(db)
```

Question 4

Let's build a classifier using logistic lasso. Report the selected features and identify which ones are associated with an increase in the estimated probability of being spam and which ones are associated with a decrease in the estimated probability of being spam via a "word cloud" visualization. I have included some of the relevant code below. Complete the code (including installing wordcloud).

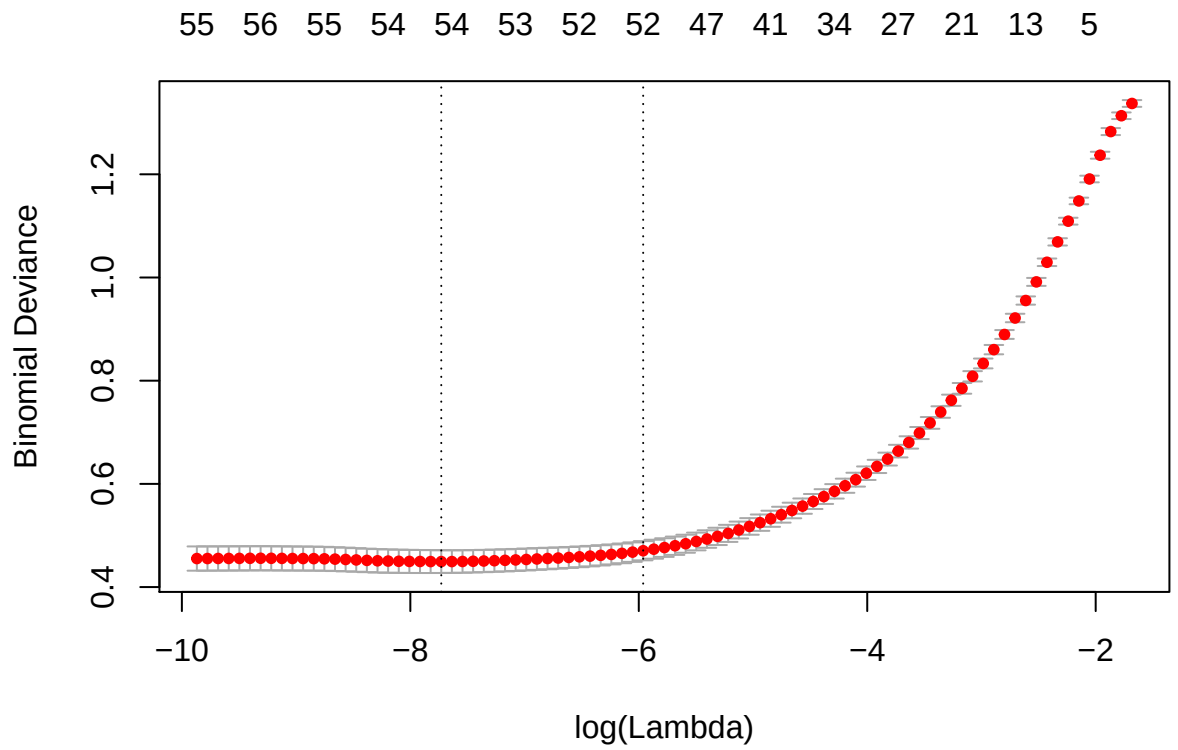
```
require(glmnet)
```

```
## Loading required package: glmnet
## Loading required package: Matrix
## Loading required package: foreach
## Loaded glmnet 2.0-13
```

```
Xmat = as.matrix(X,dimnames = NULL)
Ynum = Ynum = as.numeric(Y)-1

lasso.cv.glmnet = cv.glmnet(Xmat,Ynum,alpha=1,family='binomial')

##SOLUTION
# Check that the CV minimum doesn't occur on boundary
plot(lasso.cv.glmnet)
```




```
# Get the coefficient estimate, call it betaHat.lasso
betaHat.lasso    = coef(lasso.cv.glmnet,s='lambda.min')[-1]
```

```
# Get the selected model, call it S.lasso
(S.lasso          = which(abs(betaHat.lasso) > 1e-16))
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
## [24] 24 25 26 27 28 29 30 31 33 35 36 37 38 39 40 41 42 43 44 45 46 47 48
## [47] 49 50 51 52 53 54 56 57
```

```
if(!require('wordcloud')){install.packages('wordcloud', repos = 'http://cran.us.r-project.org'); require
```

```
## Loading required package: wordcloud
```

```
## Loading required package: RColorBrewer
```

```
negPos = betaHat.lasso[S.lasso] < 0
colVec = rep('red',length(negPos))
colVec[negPos] = 'blue'


wordcloud(names(X)[S.lasso],abs(betaHat.lasso[S.lasso]),
          min.freq = 1e-16, scale=c(4,.5),
          colors = colVec,ordered.colors=T,random.order=T, rot.per=.15)
```

Explain the meaning of the color and size of the words in this plot. Which word is most associated with the email being spam? Which word is least associated with being spam (that is, most associated with not being spam)?

SOLUTION

The size of the word is the magnitude of the estimated coefficient from the logistic lasso. Hence, it is the rate of change of the estimated log odds of being spam vs. not spam. Larger values indicates that the estimated log odds change more for a given change in a feature. The color indicates the sign of the coefficient. For a positive coefficient (red), an increase in the feature is associated with an increase in estimated log odds (and hence an increase in estimated probability of being spam). Negative coefficients are the opposite.

Hence 'punc_dollar' is most associated with an increase in estimated probability of being spam. Likewise, 'CS' is the least associated with an increase in estimated probability of being spam.

Question 5: LDA

Run the following code

```
require(MASS)
```

```
## Loading required package: MASS
```

```
out.lda = lda(Y~.,data=X)
out.lda$prior
```

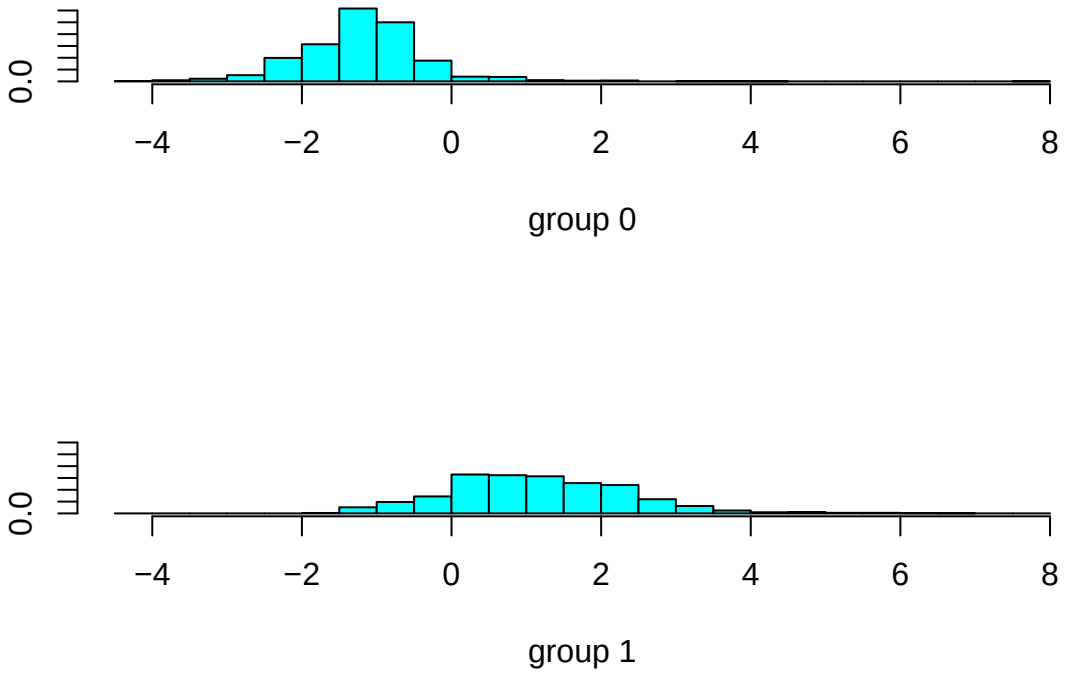
```
##         0         1
## 0.6096441 0.3903559
```

What do these 'prior' values correspond to?

SOLUTION

As mentioned in lecture, LDA provides supervised dimension reduction as well. We can visualize this dimension reduction via plotting the lda object. Recall that the dimension reduction is to a dimension 1 less than the number of levels of the supervisor. In this case, there are 2 levels (spam or no spam) and hence it is 1 dimensional. In this case, R plots two histograms: one for training observations with supervisor 'spam' and one for training observations with supervisor 'not spam'. Insights into the data can be made by noticing that the emails that are near to zero are more difficult to classify (there is a nice discussion of LDA in R here: http://uc-r.github.io/discriminant_analysis).

```
plot(out.lda)
```



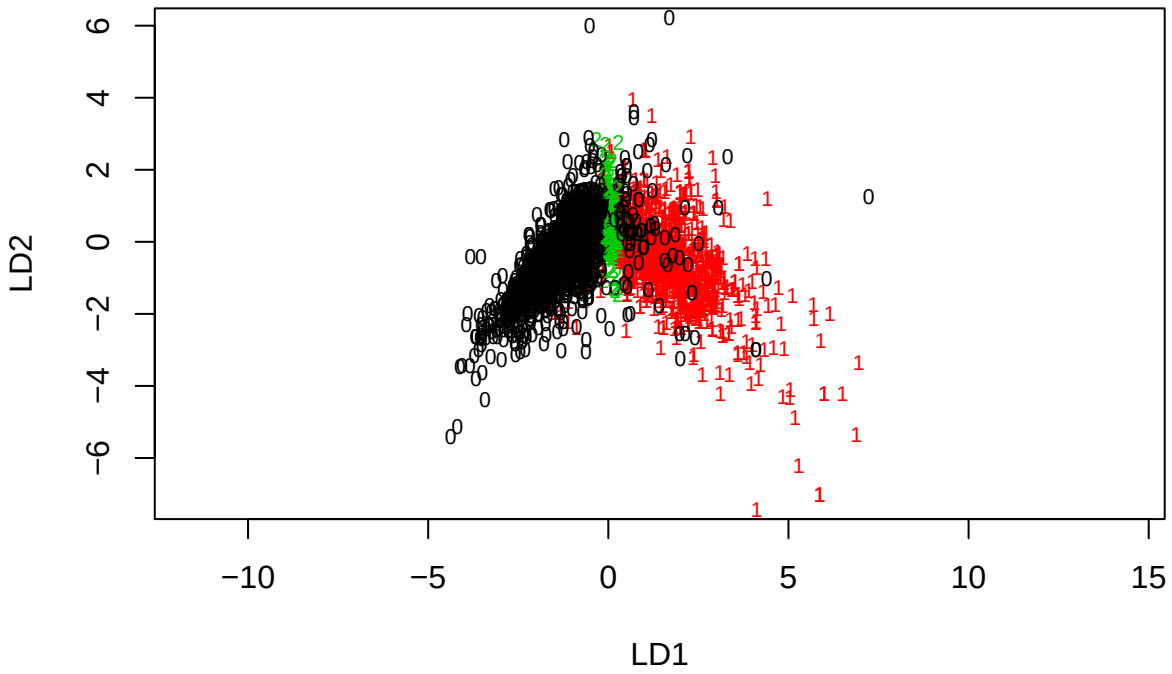
Let's try the following just for fun. Let's make a third level for the supervisor. Looking at the 'posterior' probability estimates, any probability near 0.5 indicates that we are relatively unsure of the classification. So, let's make the third level those observations with probabilities between 0.4 and 0.6

```
pred.lda = predict(out.lda)
unsure = pred.lda$posterior[,1] > 0.4 & pred.lda$posterior[,1] < 0.6
```

```
Ynew = Ynum
Ynew[unsure] = 2
table(Ynew)
```

```
## Ynew
##    0    1    2
## 2617 1507  231
```

```
out.lda = lda(Ynew~.,data=X)
plot(out.lda,col=Ynew+1)
```



Comment on this new plot. In particular, what dimension is it in and what seems to be the relationship between the levels of the supervisor?

SOLUTION:

It is in two-dimensional space. Indeed the training supervisors that we were most unsure of are located at the boundary of the more sure classifications.

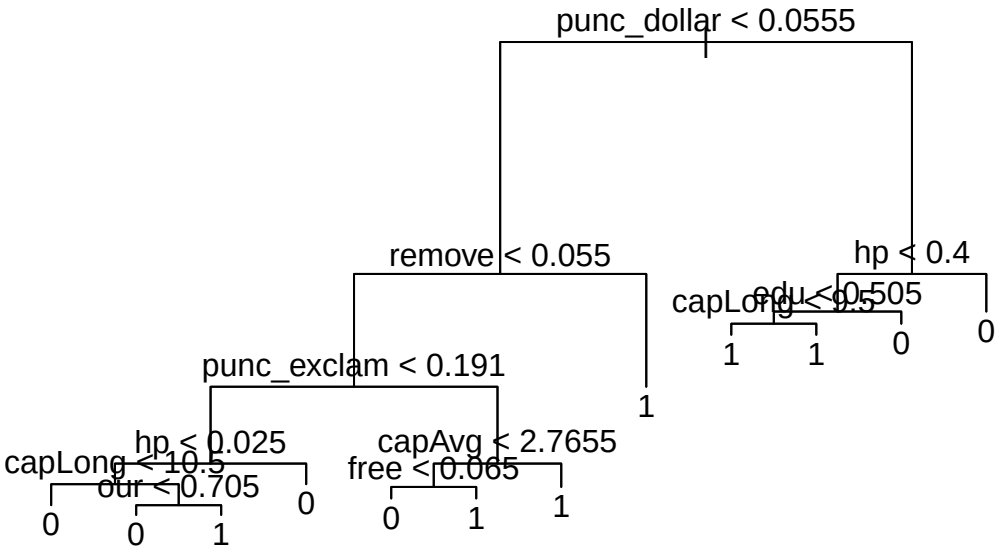
Question 6: Trees

Run the following code

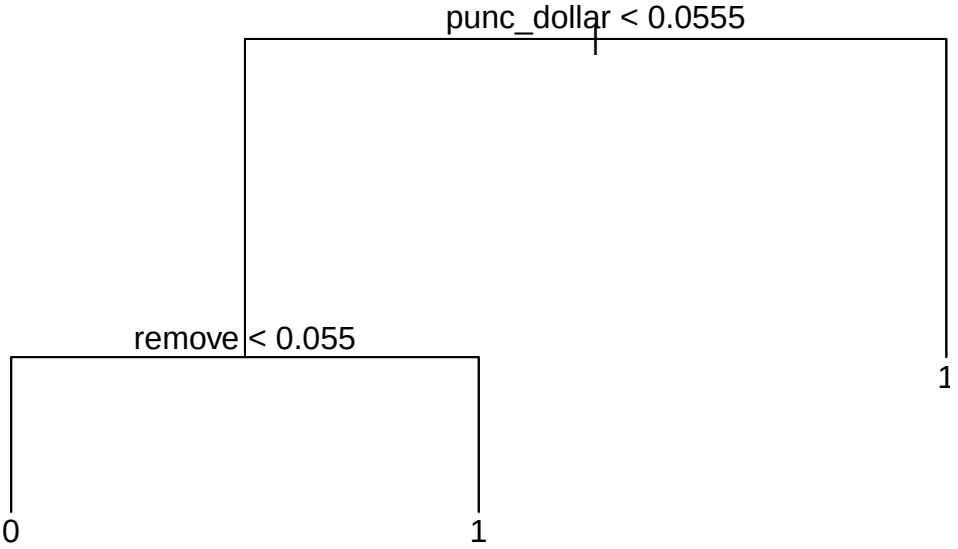
```
if(!require('tree')){install.packages('tree', repos = 'http://cran.us.r-project.org'); require('tree')}
```

```
## Loading required package: tree
```

```
out.tree = tree(Y~.,data=X)
plot(out.tree)
text(out.tree)
```




```
tmp.tree = prune.tree(out.tree,best=3)
plot(tmp.tree)
text(tmp.tree)
```



Interpet the first split point/feature

SOLUTION

'punc_dollar' is the most important feature.

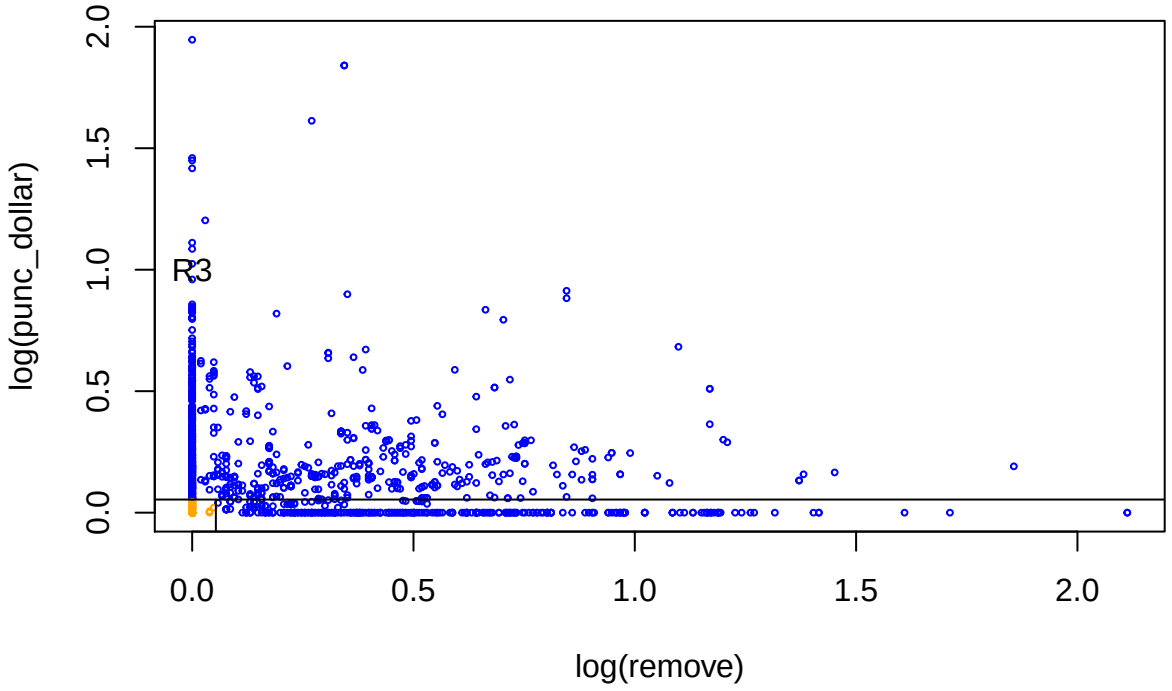
Make the corresponding partition view for this dendrogram

```
#SOLUTION
punc_dollar = X$punc_dollar
remove      = X$remove
offSet      = 1

logPunc_dollar = log(punc_dollar + offSet)
logRemove      = log(remove + offSet)
```

```
removeSplit         = 0.055
logRemoveSplit      = log(removeSplit + offSet)
punc_dollarSplit    = 0.0555
logPunc_dollarSplit = log(punc_dollarSplit + offSet)

color =rep('blue',nrow(X))
color[logRemove < logRemoveSplit & logPunc_dollar < logPunc_dollarSplit] = 'orange'
plot(logRemove,logPunc_dollar,xlab='log(remove)',ylab='log(punc_dollar)',cex=.4,col=color)
abline(h = logPunc_dollarSplit)#punc_dollar
segments(x0 = logRemoveSplit, y0 = -10, y1 = logPunc_dollarSplit)#remove
text(x =c(-3.75,1,0),y =c(-4,-4,1), labels=c('R1','R2','R3'))
```



Question 7: Misclassification rates and confusion matrices

Fit an unpruned classification tree to the training data (hint: you've already done that on this h/w). Get the associated test misclassification rate and test confusion matrix. Compare this to the misclassification rate and test confusion matrices for the logistic lasso.

```
# Get the test predictions via logistic lasso
Xmat_0 = as.matrix(X_0,dimnames = NULL)
Ynum_0 = as.numeric(Y_0)-1
Yhat.lasso   = Yhat.lasso = predict(lasso.cv.glmnet, Xmat_0,
                    s='lambda.min',type='class')
Yhat.tree    = predict(out.tree,X_0,type='class')
##SOLUTION: Use the misclass function
misClass(Yhat.tree,Y_0)
```

```
## [1] "misclass"
## [1] 0.07317073
## [1] "confusion mat"
```

```
##           true.class
## pred.class   0   1
##          0 127  12
##          1   6 101
```

```
misClass(Yhat.lasso,Y_0)
```

```
## [1] "misclass"
## [1] 0.05691057
## [1] "confusion mat"
##           true.class
## pred.class   0   1
##          0 128   9
##          1   5 104
```