STAT 2450 Assignment 7 (50 points)

Your name here

Banner: B00783546

Problem 1: Surviving the Titanic (32 points)

Load the librairies

```r
if(!require(Hmisc)) install.packages("Hmisc",repos = "http://cran.us.r-project.org")
```

```
## Loading required package: Hmisc

## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

## Loading required package: ggplot2

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```r
#library(Hmisc)
library("caret")
```

```
##
## Attaching package: 'caret'

## The following object is masked from 'package:survival':
##
##     cluster
```

```r
library("rpart")
library("tree")
library("e1071")
```

```
##
## Attaching package: 'e1071'

## The following object is masked from 'package:Hmisc':
##
##     impute
```

```r
library(ggplot2)
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

Load the data

```r
mytrain = read.csv("https://mathstat.dal.ca/~fullsack/DATA/titanictrain.csv")
mytest = read.csv("https://mathstat.dal.ca/~fullsack/DATA/titanictest.csv")
mytitanic = rbind(mytest,mytrain)
nrec=nrow(mytitanic)
```

You will be using the column 'Survived' as the outcome in our models. This should be treated as a factor. All other columns are admissible as predictors of this outcome.

HINT-1: you can use the following template to split the data into folds, e.g. for cross-validation.

Randomly shuffle your data

yourData<-yourData[sample(nrow(yourData)),]

Create 10 pre-folds of equal size

myfolds <- cut(seq(1,nrow(yourData)),breaks=10,labels=FALSE)

use these pre-folds for cross-validation

for(i in 1:10){ # loop over each of 10 folds # recover the indexes of fold i and define the indexes of the test set testIndexes <- which(myfolds==i,arr.ind=TRUE) # define yout test for this fold testData <- yourData[testIndexes,] # define your training set for this fold as the complement trainData <- yourData[-testIndexes,] # .... }

HINT-2: Use the following template to split data into a train and a test set of roughly the same size

set.seed(44182) # or use the recommended seed trainindex=sample(1:nrec,nrec/2,replace=F) mytrain=mydata[trainindex,] # training set mytest=mydata[-trainindex,] # testing set = complementary subset of mydata

1. Define a 5 pre-folds of equal size of 'mytitanic' in a variable called 'myfolds'

(2 points)

```
set.seed(2255)
# shuffle
#
# Create 5 folds of equal size
# myfolds ...
myData<-mytitanic[sample(nrow(mytitanic)),]
myfolds<-cut(seq(1,nrow(mytitanic)),breaks=5,labels=FALSE)
```

2. Use pre-fold number 3 to define a testing and a training set named 'mytest' and 'mytrain'

(2 points)

```
i=3 # fold number to use
mytest=NULL
mytrain=NULL
for(j in 1:3){ # loop over each of 3 folds
    # recover the indexes of fold i  and define the indexes of the test set
    testIndexes <- which(myfolds==i,arr.ind=TRUE)
    # define yout test  for this fold
    mytest<- mytitanic[testIndexes, ]
    # define your training set for this fold as the complement
    mytrain <- mytitanic[-testIndexes, ]

}
```

3. Fit a Random Forest model to the 'mytrain' dataset. Use the column 'Survived' as a factor outcome. Require importance to be true and set the random seed to 523. (This is the 'trained model'). (2 points)

```
# Fitting Random Forest Classification to the training set 'mytrain'
set.seed(523) # or use the recommended seed

randomf<-randomForest(as.factor(Survived)~.,data=mytrain,importance=T)
```

4. Plot the trained model results.

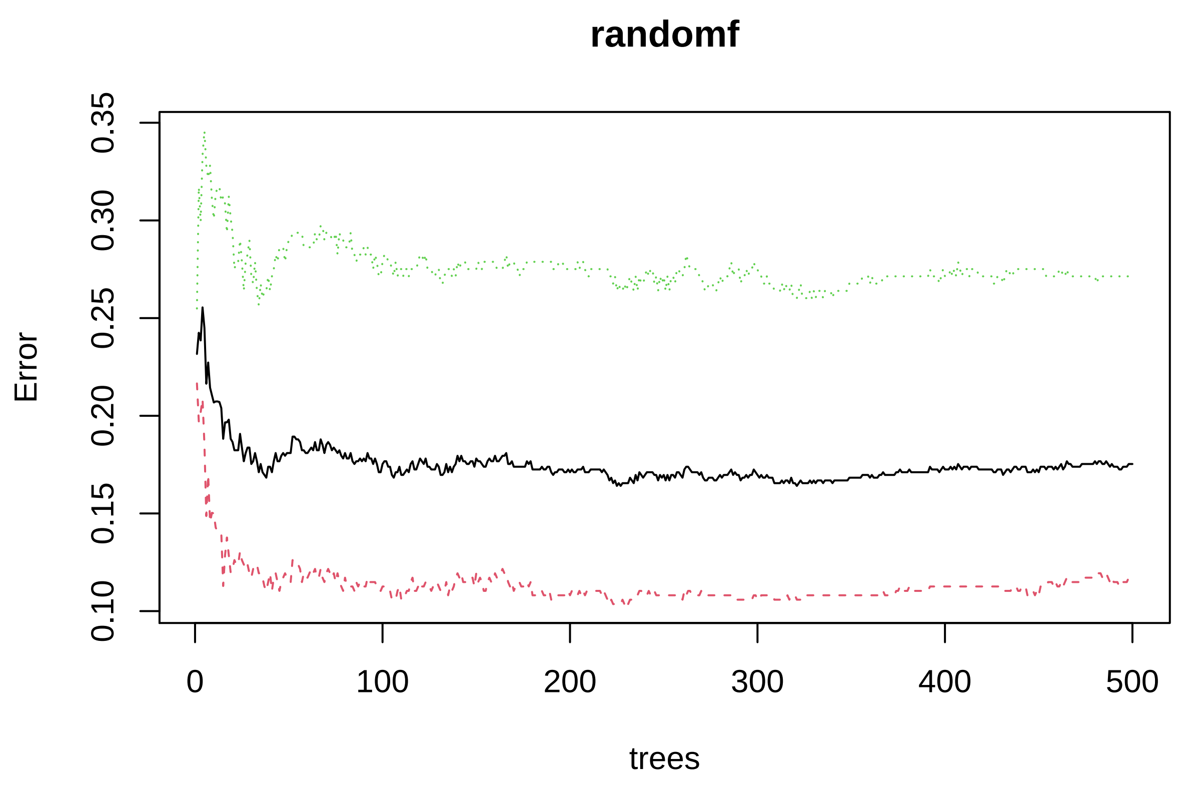
   Has the OOB error rate roughly equilibrated with 50 trees?

   Has the OOB error rate roughly equilibrated with 500 trees?

   What is the stationary value of the OOB error rate?

   Which of death or survival has the smallest prediction error? (4 points)

```
plot(randomf)
```



```
#
```

5. Calculate the predictions on 'mytest', the misclassification error and the prediction accuracy. (2 points)

```
# Predicting survival on mytest
randomf.pred <- predict(randomf,newdata=mytest[,-which(names(mytest)=="Survived")])

table(randomf.pred,mytest$Survived)
```

```
##
## randomf.pred  0  1
##            0 89 16
##            1 16 57
```

```
mean(mytest$Survived!=randomf.pred)
```

```
## [1] 0.1797753
```

```
mean(mytest$Survived==randomf.pred)
```

```
## [1] 0.8202247
```

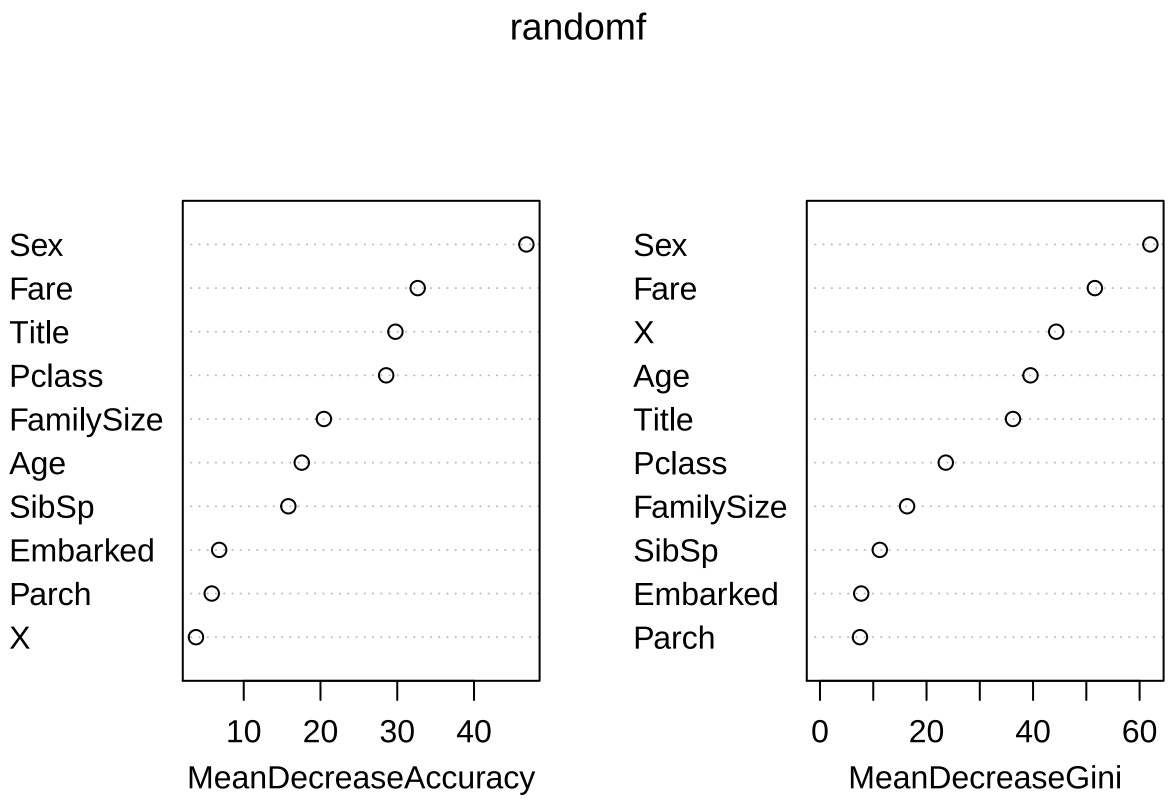
The misclassification error is 0.1797753 and the prediction accuracy is 0.8202247. Results depend on situations.

6. Print and plot the importance of predictors in the trained model. (2 points)

```r
importance(randomf)
```

```
##                    0         1 MeanDecreaseAccuracy MeanDecreaseGini
## X           1.214946  4.168355             3.781623        44.348532
## Pclass     15.455910 25.189177            28.559860        23.611363
## Sex        37.726791 32.905445            46.821328        62.025090
## Age        11.469183 12.058883            17.565394        39.517004
## SibSp      15.310405  2.984965            15.812302        11.228062
## Parch       4.220621  4.309217             5.838419         7.508218
## Fare       19.357370 26.195799            32.663747        51.600136
## Embarked    6.607929  2.893748             6.801355         7.739985
## Title      28.584023 14.504982            29.760527        36.230830
## FamilySize 15.747570 12.839416            20.442957        16.351215
```

```r
varImpPlot(randomf)
```



Now you are going to have a more direct look at predictors for the records in 'mytest'.

Tabulate the chances of survival by the column 'Title'. What do you conclude? (2 points) Which other predictor would have given you the same information? (1 points)

Are the predictors independent? (1 points)

```r
table(mytest$Title,mytest$Survived)
```

```
##
##          0  1
##   Master  2  4
##   Miss   15 34
##   Mr     79 14
##   Mrs     6 21
##   Other   3  0
```

Conclusion: Male death is composed of the majority.

```r
table(mytest$Sex,mytest$Survived)
```

```
##
##          0  1
##   female 21 55
##   male   84 18
```

Sex predictor gives me the same information.

What is the median fare of passengers ? (1 points) Hint: use the column 'Fare'

```r
median((mytest$Fare))
```

```
## [1] 13.95
```

Tabulate the survival according to the binary variable mytest$Fare $<$ 15 (2 points)

```r
table(mytest$Fare < 15,mytest$Survived)
```

```
##
##         0  1
##   FALSE 37 48
##   TRUE  68 25
```

```r
rm(mytrain,mytest)
#mytrain
```

7. Complete the code of the following function, which returns a vector of classification accuracies for *nrep* random splits into a training and testing sets of size half the number of records in the dataset 'mytitanic'. (4 points)

```r
dotitan <- function(nrep,ntree,mtry){
set.seed(495)
acc = NULL
for(i in 1:nrep){

rm(mytrain,mytest)
nrec=nrow(mytitanic)

#define a train -test split as recommended  in the hints

# Fit a Random Forest Classification to the training set, using ntree trees and mtry predictors

# Predict the response on the testing set

# tabulate the prediction accuracy  ( Confusion matrix )

# compute the misclassification error

# compute the classification accuracy

}

# return the classification accuracy

}
```

Run the function with 100 replicates, 500 trees per fit and 4 variables. (1 points) Compute the mean accuracy and plot the histogram. Is the prediction performance of random forest highly variable? (2 points)

8. Once again, define a train-test split ('mytrain' and 'mytest') of 'mytitanic' of size ntrain=nrec/2, as recommended in the hints. Take 332 for random seed.

Run 50 independent fits of the random forest model, all using the SAME dataset mytrain. Accumulate the accuracy of each fit in an array of size 50. Plot the histogram of this array. Do the different fits produce similar accuracies? (4 points)

Problem 2 (18 points)

ISLR, chapter 8, problem 8. DO ALL PARTS.

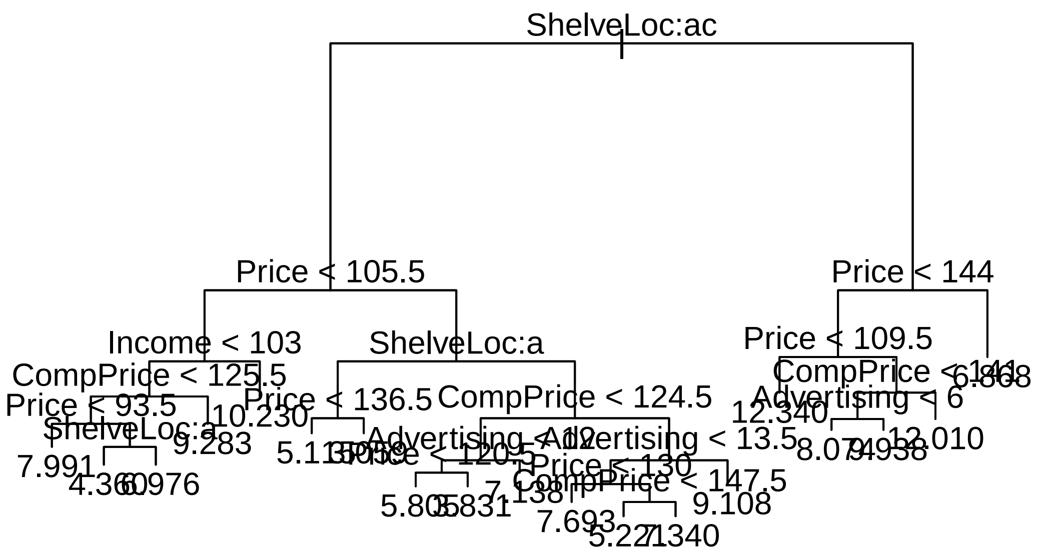
For part (a), use the following split into training and test sets.

```
set.seed(44182)
library(ISLR)
library(randomForest)

attach(Carseats)
n=nrow(Carseats)
indices=sample(1:n,n/2,replace=F)
cstrain=Carseats[indices,]
cstest=Carseats[-indices,]
```

b) 5 points for plot and an estimate of test set MSE.

```
library(tree)
train.tree=tree(Sales~.,data=cstrain)
plot(train.tree)
text(train.tree)
```



```r
testpredict=predict(train.tree, newdata = cstest)
testMSE = mean((cstest$Sales-testpredict)^2)
testMSE
```

```
## [1] 3.942756
```

The test MSE is 4.932839. c) 3 points for reporting test set MSE for the pruned tree.

```r
salesCV = cv.tree(train.tree)
bestSize = salesCV$size[salesCV$dev == min(salesCV$dev)]

tree.pruned = prune.tree(train.tree, best = bestSize)
prunepredict = predict(tree.pruned, newdata = cstest)
pruneMSE = mean((cstest$Sales-prunepredict)^2)
pruneMSE
```

```
## [1] 3.979211
```

The test set MSE for pruned tree is 4.932839. It is equal with MSE training set data. Results depend on situation. d) 3 points for output, and test set MSE.

```r
#bagging
sales.rf = randomForest(Sales~., data = cstrain, mtry = 10, importance = T)
importance(sales.rf)
```

```
##                %IncMSE IncNodePurity
## CompPrice   19.6935385    152.607054
## Income       9.5439628     94.021568
## Advertising 15.2981851    137.808341
## Population   0.7352831     62.078578
## Price       43.9905023    382.213480
## ShelveLoc   58.4730442    472.813542
## Age          3.2196012     84.121903
## Education    7.3863866     60.510985
## Urban       -0.9495023      4.818845
## US           0.9241595      7.102677
```

```
trainpredict4d = predict(sales.rf, newdata = cstrain)
testpredict4d = predict(sales.rf, newdata = cstest)

testMSE4d=mean((cstrain$Sales-trainpredict4d)^2)
testMSE4d
```

```
## [1] 0.5070693
```

The test MSE is 0.5044889. e) 3 points. results will differ with different random sequences.

```
sales.rf = randomForest(Sales~., data = cstrain, mtry = 2, importance = T)
importance(sales.rf)
```

```
##                %IncMSE IncNodePurity
## CompPrice    8.9444300     143.95684
## Income       4.0956106     127.36996
## Advertising 11.9833672     149.31423
## Population  -2.5887055     114.80647
## Price       27.6667042     269.71665
## ShelveLoc   32.3120404     290.06913
## Age          4.6700413     138.50662
## Education   -0.5223171      81.96549
## Urban       -0.3818146      18.11997
## US           2.2991081      28.80897
```

```
trainpredict4e1 = predict(sales.rf, newdata = cstrain)
testpredict4e1 = predict(sales.rf, newdata = cstest)

testMSE4e1=mean((cstrain$Sales-trainpredict4e1)^2)
testMSE4e1
```

```
## [1] 0.873987
```

The test MSE for mtry=2 is 0.9026267. f) 4 points for some sensible discussion of how the results are seen to differ with a different value of mtry. Particular choice of mtry is not important, as long as two different values were used.

```
sales.rf = randomForest(Sales~., data = cstrain, mtry = 9, importance = T)
importance(sales.rf)
```

```
##                %IncMSE IncNodePurity
## CompPrice   19.6679400    150.757429
## Income       7.5983883     91.500846
## Advertising 15.9645108    145.718187
## Population   1.1154920     63.266051
## Price       47.2416741    376.174007
## ShelveLoc   58.1445385    463.184323
## Age          5.8772861     91.688277
## Education    6.4379013     61.865022
## Urban        0.7666238      5.319941
## US           2.4866481      7.210926
```

```r
trainpredict4e2 = predict(sales.rf, newdata = cstrain)
testpredict4e2 = predict(sales.rf, newdata = cstest)

testMSE4e2=mean((cstrain$Sales-trainpredict4e2)^2)
testMSE4e2
```

```
## [1] 0.5094481
```

From e the test MSE for mtry=2 is 0.9026267 and from f the test MSE for mtry=9 is 0.5134589. The value of mtry will affect the test MSE.