Lab 8: Random Forests

Sarah Wright & Tyusha Sarawagi

Introduction

In this lab, we will practice using the random forest method to perform classification and to make numerical predictions. In addition, we will talk more about *cross validation* and *parameter tuning.*

We will be using the famous Titanic data set which includes data for many passengers on the Titanic, including whether or not they survived.

The files have already been cleaned up a bit for you and are split into testing and training sets available on Moodle.

The categories in the data represent, in order: * whether the passenger survived (1) or died (0), * Ticket Class (1 = 1st, 2 = 2nd, 3 = 3rd), * Sex, * Age (NA if unknown), * Siblings/spouses on board, * Parents/children on board, * Fare paid, * Port of Embarkation (either Cherbourg, Queenstown, or Southampton).

Random Forest Lab

Let's start with some simple exploratory data analysis and reasonable conjectures about who might have survived the Titanic.

1) Load the two data sets. Based on the characteristics above, which two or three features do you think are going to be most important for predicting survival? Give a brief justification for each. **Your datasets should both have 8 columns! Remove the variable 'X' if it exists after importing!**

```
library(class)
library(dplyr)
library(tidyverse)
library(ggplot2)
library(rpart)
library(rpart.plot)
library(Stat2Data)
```

```
set.seed(4)
test <- read.csv("/Users/sewii/Documents/CLASSES_Spring2022/Data325_AppliedDataScience/Titanic_test.csv
train <- read.csv("/Users/sewii/Documents/CLASSES_Spring2022/Data325_AppliedDataScience/Titanic_train.cs
train <-test <-test[,-1]
train[,-1]
```

```
##     Pclass    Sex  Age SibSp Parch     Fare Embarked
## 1        3   male 22.0     1     0   7.2500        S
## 2        3   male 35.0     0     0   8.0500        S
## 3        1   male 54.0     0     0  51.8625        S
```

```
## 4        2   male   NA     0     0  13.0000        S
## 5        3   male   NA     0     0   7.8958        S
## 6        3   male 22.0     0     0   7.2292        C
## 7        3 female 16.0     5     2  46.9000        S
## 8        3   male 32.0     0     0  56.4958        S
## 9        3   male   NA     0     0   8.0500        S
## 10       3   male 22.0     0     0   9.0000        S
## 11       3 female   NA     0     0   7.7875        Q
## 12       2 female 17.0     0     0  10.5000        S
## 13       3   male 16.0     1     3  34.3750        S
## 14       3   male   NA     0     0   8.0500        S
## 15       1   male 71.0     0     0  34.6542        C
## 16       2 female 34.0     0     1  23.0000        S
## 17       3   male   NA     0     0   7.8958        S
## 18       1   male 21.0     0     1  77.2875        S
## 19       3   male 33.0     0     0   8.6542        S
## 20       3   male   NA     0     0   7.7750        S
## 21       3 female   NA     1     0  24.1500        Q
## 22       3   male 70.5     0     0   7.7500        Q
## 23       2   male 29.0     1     0  21.0000        S
## 24       3 female  2.0     4     2  31.2750        S
## 25       2   male 32.5     1     0  30.0708        C
## 26       3   male 33.0     0     0   7.8958        C
## 27       2 female 29.0     1     0  26.0000        S
## 28       1 female 19.0     0     2  26.2833        S
## 29       1   male 24.0     0     0  79.2000        C
## 30       3   male 19.0     0     0   6.7500        Q
## 31       3   male 27.0     0     0   7.7958        S
## 32       3   male 55.5     0     0   8.0500        S
## 33       3   male 26.0     0     0   7.7750        S
## 34       3 female   NA     8     2  69.5500        S
## 35       2   male   NA     0     0  15.0500        C
## 36       1   male   NA     0     0  50.0000        S
## 37       1   male 45.0     0     0  26.5500        S
## 38       3   male   NA     8     2  69.5500        S
## 39       3   male 32.0     1     0  15.8500        S
## 40       3 female 16.0     0     0   7.7500        Q
## 41       1   male 40.0     0     0  31.0000        C
## 42       2   male 30.0     0     0  13.0000        S
## 43       2   male 30.0     0     0  10.5000        S
## 44       3   male 16.0     0     0   8.0500        S
## 45       2   male 18.0     0     0  13.0000        S
## 46       3 female   NA     3     1  25.4667        S
## 47       2   male 24.0     0     0  10.5000        S
## 48       2   male 19.0     0     0  10.5000        S
## 49       3 female   NA     1     0  14.4542        C
## 50       3 female   NA     1     0  15.5000        Q
## 51       3   male 22.0     0     0   7.1250        S
## 52       3 female 29.0     0     2  15.2458        C
## 53       1 female 30.0     0     0  86.5000        S
## 54       3   male   NA     0     0   7.7500        Q
## 55       3 female   NA     0     0   7.7500        Q
## 56       3 female   NA     0     0   7.7500        Q
## 57       3   male 16.0     0     0   9.5000        S
```

```
## 58       3   male 22.0     0     0   7.8958        S
## 59       3 female 24.0     0     0   8.8500        S
## 60       3   male 24.0     0     0   7.8958        S
## 61       1 female 36.0     0     0 135.6333        C
## 62       1 female   NA     1     0 133.6500        S
## 63       1   male 29.0     1     0  66.6000        S
## 64       1 female 41.0     0     0 134.5000        C
## 65       2 female 40.0     0     0  13.0000        S
## 66       1   male   NA     0     0  35.0000        S
## 67       3   male 15.0     1     1   7.2292        C
## 68       3   male   NA     0     0   7.2250        C
## 69       1 female 22.0     0     1  55.0000        S
## 70       3 female   NA     0     0   7.8792        Q
## 71       3   male 35.0     0     0   7.0500        S
## 72       3 female   NA     0     0   7.2292        C
## 73       3 female  3.0     3     1  21.0750        S
## 74       1   male 27.0     0     2 211.5000        C
## 75       3   male 20.0     0     0   4.0125        C
## 76       2   male 23.0     0     0  10.5000        S
## 77       2   male 34.0     1     0  21.0000        S
## 78       3 female   NA     3     1  25.4667        S
## 79       3   male   NA     0     0   6.8583        Q
## 80       1 female 33.0     1     0  90.0000        Q
## 81       2   male 30.0     0     0  13.0000        S
## 82       3   male 29.0     0     0   7.8750        S
## 83       3   male 18.0     1     1  20.2125        S
## 84       3   male   NA     0     0   7.2500        S
## 85       2 female 28.0     0     0  13.0000        S
## 86       1   male 65.0     0     0  26.5500        S
## 87       2 female 50.0     0     0  10.5000        S
## 88       3   male   NA     0     0   8.0500        S
## 89       2 female 33.0     1     2  27.7500        S
## 90       2   male 34.0     1     0  21.0000        S
## 91       3   male 22.0     0     0   7.5208        S
## 92       2   male   NA     0     0   0.0000        S
## 93       3 female 63.0     0     0   9.5875        S
## 94       1   male 25.0     1     0  91.0792        C
## 95       3   male  9.0     1     1  15.9000        S
## 96       3   male   NA     1     0  19.9667        S
## 97       3   male 21.0     0     0   7.2500        S
## 98       1 female 25.0     1     2 151.5500        S
## 99       2 female 33.0     0     2  26.0000        S
## 100      3   male 24.0     0     0   7.4958        S
## 101      3   male   NA     0     0  24.1500        Q
## 102      2 female  7.0     0     2  26.2500        S
## 103      1 female 22.0     0     2  49.5000        C
## 104      2   male 27.0     0     0  26.0000        S
## 105      3   male 22.0     0     0   7.2250        C
## 106      3 female 22.0     0     0   7.7750        S
## 107      1   male   NA     0     0 227.5250        C
## 108      2   male 28.0     0     0  13.5000        S
## 109      3   male 19.0     0     0   7.8958        S
## 110      3 female 29.0     0     4  21.0750        S
## 111      1 female 39.0     1     0  55.9000        S
```

```
## 112      2 female 25.0     1     1  30.0000        S
## 113      1 female 18.0     0     2  79.6500        S
## 114      3   male 22.0     0     0   8.0500        S
## 115      3   male 47.0     0     0   7.2500        S
## 116      3   male 44.0     0     0   8.0500        S
## 117      2 female 22.0     1     2  41.5792        C
## 118      2   male 57.0     0     0  12.3500        Q
## 119      1 female 21.0     0     0  77.9583        S
## 120      1   male 80.0     0     0  30.0000        S
## 121      3 female 41.0     0     5  39.6875        S
## 122      3   male 20.0     0     0   7.8542        S
## 123      3   male   NA     0     0  56.4958        S
## 124      3 female 23.0     0     0   7.5500        S
## 125      3 female 32.0     1     1  15.5000        Q
## 126      1   male 50.0     2     0 133.6500        S
## 127      3   male 40.0     0     0   7.2250        C
## 128      2   male 32.0     2     0  73.5000        S
## 129      3 female 18.0     0     0   9.8417        S
## 130      3 female 43.0     1     6  46.9000        S
## 131      3   male 20.0     0     0   9.2250        S
## 132      2   male 60.0     1     1  39.0000        S
## 133      1 female 15.0     0     1 211.3375        S
## 134      1 female 18.0     1     0 227.5250        C
## 135      2 female 45.0     0     0  13.5000        S
## 136      1   male 48.0     1     0  52.0000        S
## 137      2 female 27.0     0     0  10.5000        S
## 138      2 female  6.0     0     1  33.0000        S
## 139      3 female 25.0     1     0   7.9250        S
## 140      1 female 29.0     0     0 211.3375        S
## 141      2   male 23.0     0     0  13.0000        S
## 142      3 female 48.0     1     3  34.3750        S
## 143      1   male 35.0     0     0 512.3292        C
## 144      3   male   NA     0     0   7.8958        S
## 145      1   male 36.0     1     0  78.8500        S
## 146      3   male  6.0     0     1  12.4750        S
## 147      3   male   NA     0     0  14.5000        S
## 148      3   male 20.0     0     0   7.2292        C
## 149      3 female 30.5     0     0   7.7500        Q
## 150      3   male 18.0     0     0   7.7500        S
## 151      3 female 13.0     0     0   7.2292        C
## 152      3   male 25.0     0     0   7.0500        S
## 153      2 female 31.0     1     1  26.2500        S
## 154      3   male 27.0     0     0   6.9750        S
## 155      3 female 18.0     0     0   7.7750        S
## 156      2   male 39.0     0     0  13.0000        S
## 157      3   male 10.0     3     2  27.9000        S
## 158      1 female 52.0     1     1  93.5000        S
## 159      3   male  2.0     4     1  39.6875        S
## 160      3   male   NA     0     0   6.9500        Q
## 161      3   male   NA     0     0  56.4958        S
## 162      3 female 15.0     1     0  14.4542        C
## 163      3   male 23.0     0     0   7.8542        S
## 164      3   male 32.0     0     0  56.4958        S
## 165      1   male   NA     0     0  29.7000        C
```

```
## 166      3   male 34.5     0     0   6.4375        C
## 167      3   male  4.0     4     2  31.2750        S
## 168      3 female  9.0     1     1  15.2458        C
## 169      1 female 45.0     1     1 164.8667        S
## 170      3   male   NA     0     0   7.2292        C
## 171      3   male 41.0     2     0  14.1083        S
## 172      2   male 24.0     0     0  13.0000        S
## 173      3   male 26.0     0     0   7.8958        S
## 174      2 female 28.0     1     0  24.0000        C
## 175      3   male 20.0     0     0   9.8458        S
## 176      3   male 19.0     0     0   7.8958        S
## 177      3   male 25.0     0     0   7.0500        S
## 178      3 female 39.0     0     5  29.1250        Q
```

```
head(test, 10)
```

```
##    Survived Pclass    Sex Age SibSp Parch    Fare Embarked
## 1         0      3   male  22     1     0  7.2500        S
## 2         0      3   male  35     0     0  8.0500        S
## 3         0      1   male  54     0     0 51.8625        S
## 4         1      2   male  NA     0     0 13.0000        S
## 5         0      3   male  NA     0     0  7.8958        S
## 6         0      3   male  22     0     0  7.2292        C
## 7         0      3 female  16     5     2 46.9000        S
## 8         1      3   male  32     0     0 56.4958        S
## 9         0      3   male  NA     0     0  8.0500        S
## 10        0      3   male  22     0     0  9.0000        S
```

```
head(train,10)
```

```
##    Survived Pclass    Sex Age SibSp Parch    Fare Embarked
## 1         0      3   male  22     1     0  7.2500        S
## 2         0      3   male  35     0     0  8.0500        S
## 3         0      1   male  54     0     0 51.8625        S
## 4         1      2   male  NA     0     0 13.0000        S
## 5         0      3   male  NA     0     0  7.8958        S
## 6         0      3   male  22     0     0  7.2292        C
## 7         0      3 female  16     5     2 46.9000        S
## 8         1      3   male  32     0     0 56.4958        S
## 9         0      3   male  NA     0     0  8.0500        S
## 10        0      3   male  22     0     0  9.0000        S
```

```
#I suspect that survived , sex, pclass, parch will be the most important for predicting survival
```

2) In the training data set, what percentage of the male passengers survived? What percentage of the female passengers survived?

```
xtabs(~Survived + Sex, data = train)
```

```
##          Sex
## Survived female male
##        0     20   92
##        1     46   20
```

```
#count(Survived, x = train)

percent_Male = 89/274
percent_Female =  185 / 274

percent_Male
```

```
## [1] 0.3248175
```

```
percent_Female
```

```
## [1] 0.6751825
```

```
#32.4% of Male passangers survived, and 67.5% of female passangers survived
```

There are some missing age values in the testing and training data sets. One option would be to just toss out all of this data, but you can lose a lot of data this way. Another approach is to *impute* the missing data. This means that we just fill in a best guess for each missing data point in the test and train sets.

3) Replace all of the missing age values in **both** the test and train sets with the median age of **all** the passengers for which age data is available in both data sets. The function `is.na()` applied to a vector will return a logical vector indicating the location of `NA` values.

```
train$Age[is.na(train$Age)] <- mean(train$Age, na.rm = T)
test$Age[is.na(test$Age)] <- mean(test$Age, na.rm = T)

head(train, 10)
```

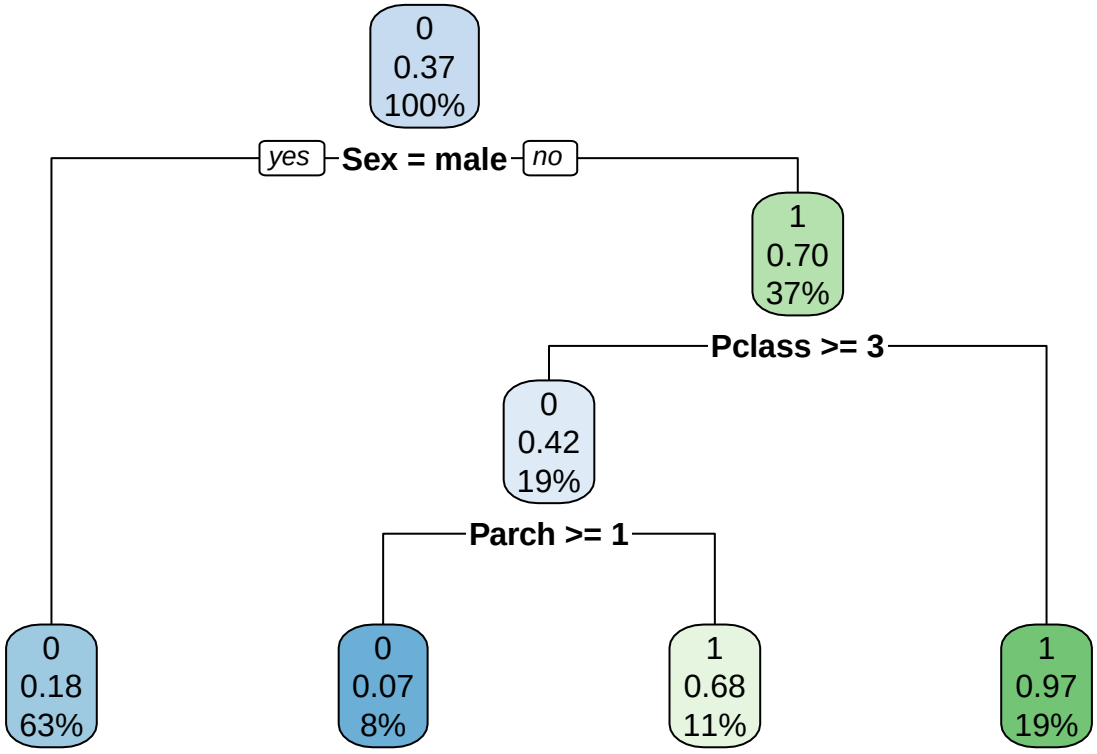
```
##    Survived Pclass    Sex      Age SibSp Parch    Fare Embarked
## 1         0      3   male 22.00000     1     0  7.2500        S
## 2         0      3   male 35.00000     0     0  8.0500        S
## 3         0      1   male 54.00000     0     0 51.8625        S
## 4         1      2   male 28.76071     0     0 13.0000        S
## 5         0      3   male 28.76071     0     0  7.8958        S
## 6         0      3   male 22.00000     0     0  7.2292        C
## 7         0      3 female 16.00000     5     2 46.9000        S
## 8         1      3   male 32.00000     0     0 56.4958        S
## 9         0      3   male 28.76071     0     0  8.0500        S
## 10        0      3   male 22.00000     0     0  9.0000        S
```

```
head(test,10)
```

```
##    Survived Pclass    Sex      Age SibSp Parch    Fare Embarked
## 1         0      3   male 22.00000     1     0  7.2500        S
## 2         0      3   male 35.00000     0     0  8.0500        S
## 3         0      1   male 54.00000     0     0 51.8625        S
## 4         1      2   male 28.76071     0     0 13.0000        S
## 5         0      3   male 28.76071     0     0  7.8958        S
## 6         0      3   male 22.00000     0     0  7.2292        C
## 7         0      3 female 16.00000     5     2 46.9000        S
## 8         1      3   male 32.00000     0     0 56.4958        S
## 9         0      3   male 28.76071     0     0  8.0500        S
## 10        0      3   male 22.00000     0     0  9.0000        S
```

4) Build a classification tree using the function `rpart()` on the training set to predict who will survive. (You might need to convert the `Survived` column of both the test and train sets to a factor column using `as.factor()`). Plot your classification tree below.

```
reg_tree <- rpart(as.factor(Survived)~., data = train)
rpart.plot(reg_tree)
```



5) Apply your model to the test set to predict who will survive. Print the confusion matrix below. What is the accuracy of this tree model? Remember to use `type = "class"`.

```
TreePrediction <- predict(reg_tree, newdata = test, type = "class")

table(TreePrediction, test$Survived)
```

```
##
## TreePrediction   0   1
##              0 105  21
##              1   7  45
```

```
#  145/178 predictions are made correctly with this model. In other words this tree model is 81% accura
```

Random Forests

One way to improve the predictive power of regression and classification trees is to build many trees and then either average the resulting predictions (for regression trees) or letting the trees "vote" on the resulting

classification (for classification trees).

In order for this to be effective, we need for each tree to be slightly different to give us different information about the data.

This can be achieved with a method called *bootstrap aggregation* (also called *bagging*). In this approach, we build multiple trees, each training on only a portion of the available data points (selected randomly, with replacement). Overall, this has a tendency to reduce the variance we get by training on different subsets of the data (more details are given in Section 8.2.1 of your textbook).

Another thing we can do, is when building each tree, restrict the training data to a random subset of the available factors. This is known as a *random forest* approach. Random, since the subsets of the variables are random, and forest, well, because there are lots of trees. By training the trees on different subsets of the factors, we decorrelate the resulting trees and prevent single factors from dominating (Random Forests are described in more detail in Section 8.2.2).

6) To build a random forest for the Titanic data set,

- Uncomment the code below
- Put in your training data as the data set.
- Choose the number of factors to consider at each step by specifying `mtry` (between 1 and 7 makes sense since we have 7 factors).
- Choose the number of trees to build by specifying `ntree`.

If you use `ntree = 1`, then your random forest is just a classification tree. If you do not specify the number of trees, the default is 500! In general, the predictive accuracy improves as you increase the number of trees, but only up to a certain point.

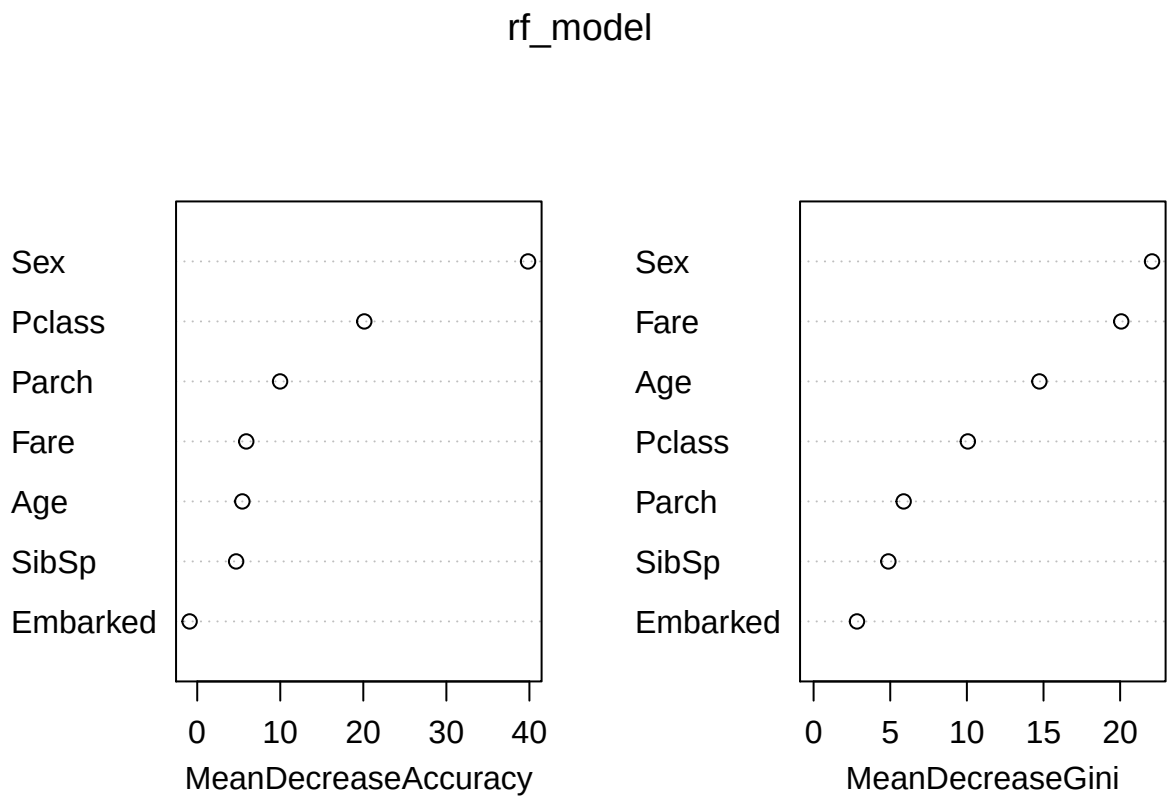
```
rf_model <- randomForest(as.factor(Survived) ~ .,
                         data =  train   , # your training data
                         mtry =  5   , # number of factors
                         ntree = 250   , # number of trees
                         importance = TRUE,
                         type = "class")
rf_model
```

```
##
## Call:
##  randomForest(formula = as.factor(Survived) ~ ., data = train,      mtry = 5, ntree = 250, importanc
##                Type of random forest: classification
##                      Number of trees: 250
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 23.6%
## Confusion matrix:
##    0  1 class.error
## 0 93 19   0.1696429
## 1 23 43   0.3484848
```

One appealing aspect of a random forest approach is that it inherently determines which factors are valuable (without requiring a preliminary factor selection process). For this reason, it is generally not necessary to restrict the variable set ahead of time to avoid overfitting.

7) Uncomment the code below to view the variable importance for the model above. Does this ranking of important factors agree with your initial impressions from Problem 1?

```
varImpPlot(rf_model)
```




```
#In problem 1, I assumed that sex, pclass, parch will be the most important for predicting survival. Us
```

The importance of each factor is determined by the percentage increase in either accuracy or the GINI index (a different measure of accuracy based on the "area under the curve") when that factor is excluded during the tree building process.

Cross Validation and Parameter Tuning

As we have seen in previous labs, there is a difference between the *train error* and the *test error*. When we are building a predictive model, we really want a model that minimizes the *test error*. One way we have addressed this is by splitting our data into a test set and a training set. This allows us to build models on the training set and then compare their performance on the test set.

However, the major problem with this approach is that the accuracy might vary greatly depending on the random split of our data into testing and training sets.

8) Below is a *for loop* that splits the original `Titanic` data set into train/test sets 50 different times. The train and test sets are called `t1` and `t2` respectively.

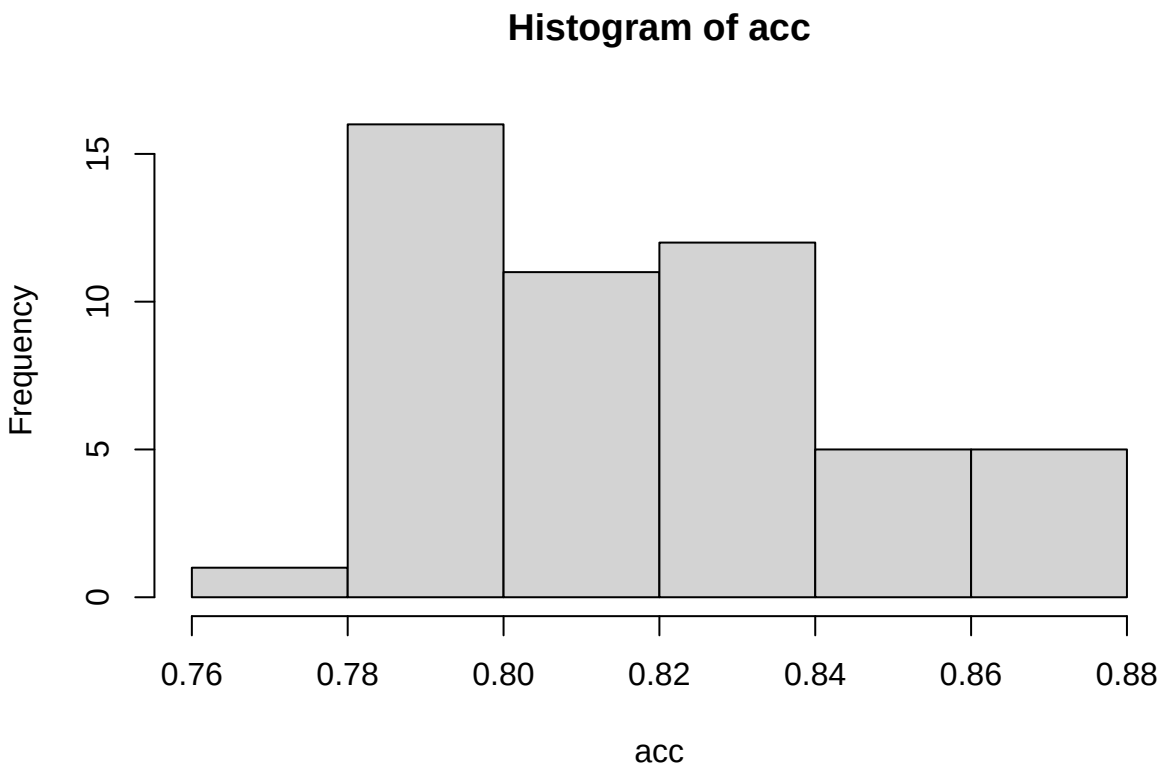
Uncomment and add to the loop code to * Build a classification tree on `t1` to predict `Survived` * Predict the `Survived` values in `t2` using your model. * Save the accuracy of the model as `acc[i]` (this will store the accuracy from each split in the vector `acc`).

```
Titanic_train <- train
Titanic_test <- test


Titanic <- rbind(Titanic_train, Titanic_test)

acc <- rep(0, 50)
for (i in 1:50){
  index <- sample(1:dim(Titanic)[1], replace = FALSE, .6*dim(Titanic)[1] )
  t1 <- Titanic[ index, ]
  t2 <- Titanic[-index, ]

   # Add your code here
  class_tree <- rpart(as.factor(Survived)~., data = t1)
  classTreePrediction <- predict(class_tree, newdata = t2, type = "class")
  table1 <- table( t2$Survived, classTreePrediction)
  acc[i] <- sum(diag(table1))/sum(table1) # divide the accuracy by the toal

}

acc
```

```
##  [1] 0.8041958 0.8181818 0.8181818 0.8531469 0.8251748 0.8251748 0.7902098
##  [8] 0.8321678 0.8531469 0.7902098 0.8251748 0.8391608 0.8531469 0.8251748
## [15] 0.8391608 0.8111888 0.7902098 0.8181818 0.8671329 0.8251748 0.8041958
## [22] 0.8741259 0.7832168 0.7832168 0.8181818 0.8041958 0.7902098 0.8601399
## [29] 0.7832168 0.7902098 0.7832168 0.8321678 0.7902098 0.8181818 0.7692308
## [36] 0.8671329 0.7972028 0.8041958 0.8391608 0.7902098 0.8461538 0.8391608
## [43] 0.8041958 0.8321678 0.7972028 0.8741259 0.7902098 0.8461538 0.7902098
## [50] 0.7972028
```

9) Create a histogram of the accuracy of your model below on different train/test splits. What is the lowest and highest accuracy value that you observe?

```
hist(acc)
```




```
#the lowest accuracy observed us 0.70 and the highest is 0.85
```

Many predictive models have parameters that we can change, or *tune*, to get better predictions. For example, in K-nearest neighbors, we can change the number of neighbors K to get different predictions for our data. The process of *parameter tuning* is the process of selecting the values of these parameters that maximize accuracy for our particular data set.

In order to tune the parameters, we need a good estimate of the accuracy of the model for each set of parameters. However, the example in the previous problems shows us that just splitting our data once can't really tell us how accurate our model is. For this reason, we use *cross validation*. This is essentially what you did in in the for loop above – we split the data multiple times and average the accuracy. Then, we can choose the parameters for the model to maximize the mean accuracy. (Technically, k-fold cross validation is a little different, because we split the data into k *folds* and then build the model k times, using each fold once as the test set and using all of the other data as the training set).

There are a lot of packages in R with functions for automatic parameter tuning. Probably the easiest function for tuning a random forest model is the `train` function from the `caret` package.

10) Install this package and uncomment the code below to load it. For tuning a random forest model, we are mostly interested in finding the best value of `mtry`, the number of variables used at each split when building the trees.

Uncomment the rest of the code to obtain a tuned random forest model. The `train()` function will automatically perform cross-validation with the selected values of `mtry` and save the best model as `tuned_model`.

```r
# This function may take a little while to run!
tuned_model <- train(x = Titanic_train[ , 2:8], y = as.factor(Titanic_train$Survived)
                     , tuneGrid = data.frame(mtry = 1:7)
                     , ntree = 250 # number of trees (passed to random forest)
                     , method = "rf")

tuned_model
```

```
## Random Forest
##
## 178 samples
##   7 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 178, 178, 178, 178, 178, 178, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   1     0.7954187  0.5365002
##   2     0.7915036  0.5431298
##   3     0.7823515  0.5272807
##   4     0.7753870  0.5140273
##   5     0.7682440  0.4995122
##   6     0.7653477  0.4919390
##   7     0.7603413  0.4815888
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 1.
```

11) You can manually `tune` the number of trees in your random forest (`ntree`) by trying out a few different values. When you have your best model, use it to predict who will survive in `Titanic_test` and report the accuracy.

```r
set.seed(3)
tuned_model <- train(x = Titanic_train[ , 2:8], y = as.factor(Titanic_train$Survived)
                     , tuneGrid = data.frame(mtry = 3) # must be between 1 : variables
                     , ntree = 200 #340 # number of trees (passed to random forest)
                     , method = "rf")

tuned_model
```

```
## Random Forest
##
## 178 samples
##   7 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 178, 178, 178, 178, 178, 178, ...
```

```
## Resampling results:
##
##   Accuracy   Kappa
##   0.7919225  0.5453779
##
## Tuning parameter 'mtry' was held constant at a value of 3
```

```
#340    0.78955
#500     0.7719022
#200     0.7918933
#100     0.7850712

# I chose Mtry to be 3 and ntree to be 200 and this was the best model. It has an accuracy of  0.780334
```

Most likely, this is greater than the accuracy of your classification tree model from Problem 5. But maybe not. There is still some randomness involved and it is always possible that on a particular data set, one method will outperform another. Still, by tuning parameters and doing cross-validation, we give ourselves the best chance of creating an accurate model.