Titanic Data Set Random Forest

Jen Dumiak

December 1, 2016

Random Forest Introduction

We will perform a random forest on our titanic data set to see how well certain features are at predicting whether a passenger survived or not. A major benefit of using a random forest is that we can let the model predict the important features.

Variable Choice and Data Cleaning

```
# Use the train data,  load here make na blank
titanic <- read.csv("C:/Users/jdumiak/Documents/Titanic/train.csv", header = TRUE, na.strings=
c(""))

summary(titanic)
```

```
##   PassengerId       Survived          Pclass     
##  Min.   :  1.0   Min.   :0.0000   Min.   :1.000  
##  1st Qu.:223.5   1st Qu.:0.0000   1st Qu.:2.000  
##  Median :446.0   Median :0.0000   Median :3.000  
##  Mean   :446.0   Mean   :0.3838   Mean   :2.309  
##  3rd Qu.:668.5   3rd Qu.:1.0000   3rd Qu.:3.000  
##  Max.   :891.0   Max.   :1.0000   Max.   :3.000  
##                                                  
##                                     Name         Sex           Age       
##  Abbing, Mr. Anthony                  :  1   female:314   Min.   : 0.42  
##  Abbott, Mr. Rossmore Edward          :  1   male  :577   1st Qu.:20.12  
##  Abbott, Mrs. Stanton (Rosa Hunt)     :  1                Median :28.00  
##  Abelson, Mr. Samuel                  :  1                Mean   :29.70  
##  Abelson, Mrs. Samuel (Hannah Wizosky):  1                3rd Qu.:38.00  
##  Adahl, Mr. Mauritz Nils Martin       :  1                Max.   :80.00  
##  (Other)                              :885                NA's   :177    
##      SibSp           Parch             Ticket         Fare       
##  Min.   :0.000   Min.   :0.0000   1601    :  7   Min.   :  0.00  
##  1st Qu.:0.000   1st Qu.:0.0000   347082  :  7   1st Qu.:  7.91  
##  Median :0.000   Median :0.0000   CA. 2343:  7   Median : 14.45  
##  Mean   :0.523   Mean   :0.3816   3101295 :  6   Mean   : 32.20  
##  3rd Qu.:1.000   3rd Qu.:0.0000   347088  :  6   3rd Qu.: 31.00  
##  Max.   :8.000   Max.   :6.0000   CA 2144 :  6   Max.   :512.33  
##                                   (Other) :852                   
##          Cabin     Embarked  
##  B96 B98    :  4   C   :168  
##  C23 C25 C27:  4   Q   : 77  
##  G6         :  4   S   :644  
##  C22 C26    :  3   NA's:  2  
##  D          :  3
```

```
##  (Other)    :186
##  NA's       :687
```

```
# Check for missing values
sapply(titanic,function(x) sum(is.na(x)))
```

```
## PassengerId    Survived      Pclass        Name         Sex         Age
##           0           0           0           0           0         177
##       SibSp       Parch      Ticket        Fare       Cabin    Embarked
##           0           0           0           0         687           2
```

```
# Cabin has way too many missing values, drop this variable immeadiately
# Also Pasenger ID since it is an index, name because it is too specific and ticket
# Check for unique values
sapply(titanic, function(x) length(unique(x)))
```

```
## PassengerId    Survived      Pclass        Name         Sex         Age
##         891           2           3         891           2          89
##       SibSp       Parch      Ticket        Fare       Cabin    Embarked
##           7           7         681         248         148           4
```

A brief glance at the data shows that Cabin has way too many missing variables, so we dropped this variable because random forests cannot handle missing data. Further, PassengerID, name, and ticket are too specific to one person making it the most important data to split the tree on. We will drop these features as well and are now ready to begin building our model.

The features we included to predict survived (1=survived, 0=dead) are:

```
    class, 1 = 1st class, 2 = 2nd class, 3 = third class
    sibsp=Number of Siblings/Spouses Aboard
    parch=Number of Parents/Children Aboard
    fare=Passenger Fare
    embarked=Port of Embarkation
```

We needed to clean our data because, as mentioned above, random forests cannot take missing values. We use a decision tree to determine the value of age and replace the two missing values in with ‘S’ because a majority of the data have that value. Lastly, we factor the data.

```
# Age has too many missing values, but is most likely important, we will replace the NA with w
hatever our decision tree predicts
Agefit <- rpart(Age ~ Pclass + Sex + SibSp + Parch + Fare + Embarked,
                data=titanic[!is.na(titanic$Age),],
                method="anova")
titanic$Age[is.na(titanic$Age)] <- predict(Agefit, titanic[is.na(titanic$Age),])

# Embark has two blanks, a majority have 'S' so we will just replace those NA with that value
titanic$Embarked[c(which(is.na(titanic$Embarked)))] <- "S"

# Fare has no NA value so good here
```

```
# Factor data
titanic$Pclass <- factor(titanic$Pclass, ordered = TRUE, levels = c("3","2","1"))
# Rename, 1st class is better than third class
levels(titanic$Pclass) <- c("Third Class", "Second Class", "First Class")
titanic$Sex <- as.factor(titanic$Sex)
titanic$Embarked <- factor(titanic$Embarked)
```

Train and Test: Random Forest Model

A crucial step to use in predictive modeling is to split the dataset into test and train to avoid overfitting to this specific partition of the data. We set a random seed and split the data into train and test and will now work with the train set.

```
# Function to split data, dataSplit
 dataSplit <- function(dataFrame, splitPercent, seed){
      # Split the data into two sets
      smp_size <- floor(splitPercent * nrow(dataFrame))

      # Set the seed to make your partition reproducible
      set.seed(seed)
      trainindex <- sample(seq_len(nrow(dataFrame)), size = smp_size)

      dataFrame[trainindex, 13]  <- "Train"
      dataFrame[-trainindex, 13] <- "Test"
      colnames(dataFrame)[13] <- "Label"
      train <- dataFrame[trainindex, ]
      test <- dataFrame[-trainindex, ]

      returnList <- list(dataFrame, train, test)
      return(returnList)
    }
outputList <- dataSplit(titanic, 0.5, 123)
titanic <- data.frame(outputList[1])
train <- data.frame(outputList[2])
test <- data.frame(outputList[3])
```

Now, we will create the random forest with 2000 trees from the train set using all the features we stated above.

```
# Create the random forest with 2000 trees from the train set, also monitor what variables are
 important
fit <- randomForest(as.factor(Survived) ~ Pclass + Sex + Age + Fare + SibSp + Parch +
             Embarked,
                    data=train,
                    importance=TRUE,
                    ntree=2000)
```

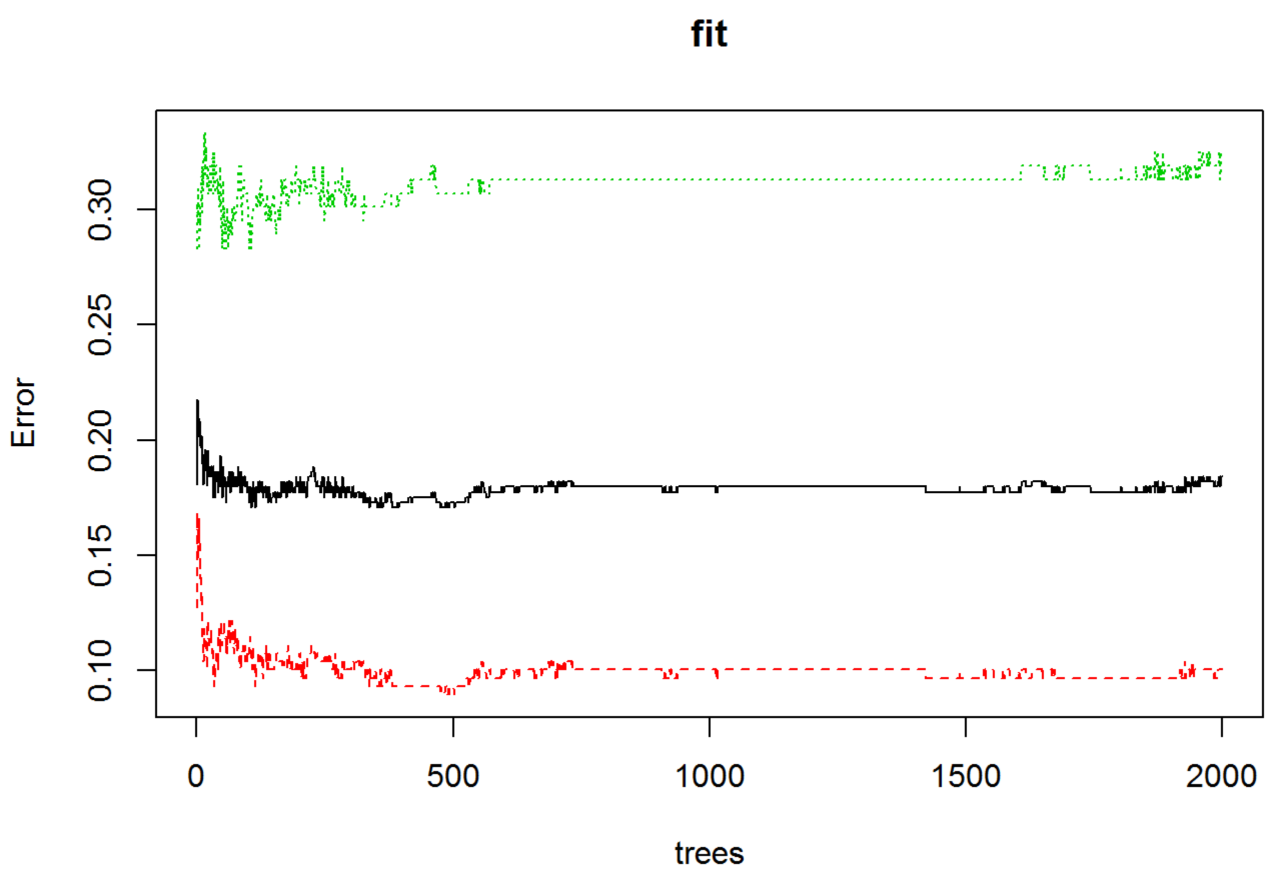
We have our random forest model and would like to see what variables are significant.

```
# See what variables are significant
print(fit) # view results
```

```
##
```

```
## Call:
##  randomForest(formula = as.factor(Survived) ~ Pclass + Sex + Age +      Fare + SibSp + Parc
h + Embarked, data = train, importance = TRUE,      ntree = 2000)
##                Type of random forest: classification
##                      Number of trees: 2000
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 18.43%
## Confusion matrix:
##     0   1 class.error
## 0 251  28   0.1003584
## 1  54 112   0.3253012
```

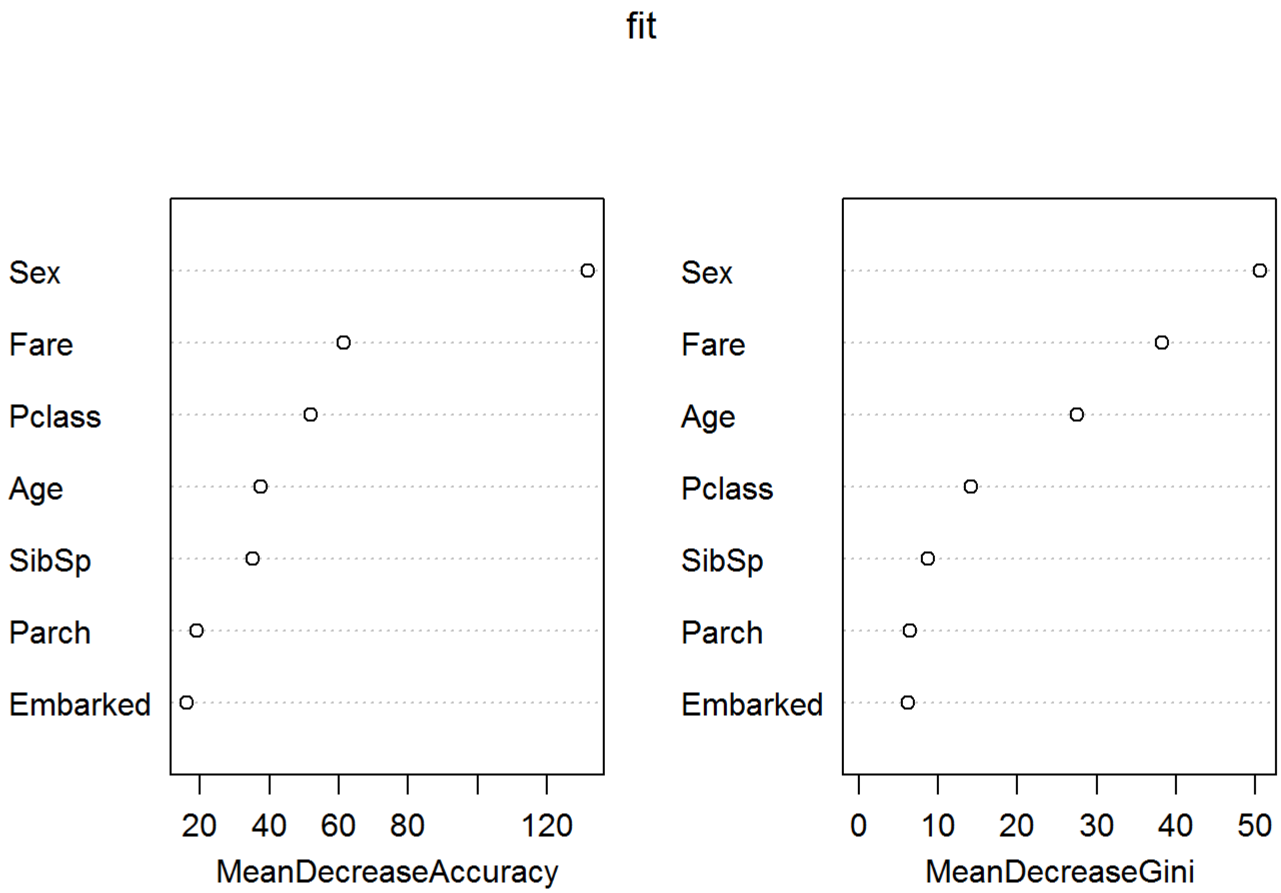
```
plot(fit) # see where the ntree flattens the error out
```




```
importance(fit) # importance of each predictor
```

```
##                   0          1 MeanDecreaseAccuracy MeanDecreaseGini
## Pclass    30.654422  42.239227             51.89329        14.080648
## Sex      100.932223 113.849111            131.87624        50.669766
## Age       28.927624  22.157358             37.47831        27.551426
## Fare      40.893011  41.465861             61.58564        38.274138
## SibSp     30.655145  16.176418             35.25014         8.653445
## Parch     17.189195   6.713374             19.00092         6.446714
## Embarked  -1.517577  22.575854             16.21618         6.234485
```

```
varImpPlot(fit)
```




```
# Variables most important are Sex, Fare, Pclass, and Age rest are insigificant to the model
```

The accuracy plot tests to see how worse the model performs without each feature, so a high decrease in accuracy would be expected for very predictive features. Meanwhile, the Gini plot measures how pure the nodes are at the end of the tree and tests to see the result if each feature is taken out; a high score means the feature was important. Finally, we can see that the most important features are Sex, Fare, Pclass, and Age, while the rest are insigificant to predicting whether a passenger survived.

Predicting Passenger Survival

Since we have our random forest fit from our train data, we want to predict whether a passenger survives using the test data to see how well our model performs. We will use the predict function to do so:

```
# Predict with the test data
pred <- data.frame(predict(fit, test, type = "class"))
```

We want to see how well our model did with the test data, so we will look at the accuracy rate.

```
# Look at the misclassification error to see how well model is doing
misclassificationError <- mean(pred != test$Survived)
```

```
print(paste('Accuracy',1-misclassificationError))
```

```
## [1] "Accuracy 0.800448430493274"
```

Here our accuracy rate is about 80%, meaning we correctly predicted whether the passenger survived 80% of the time.

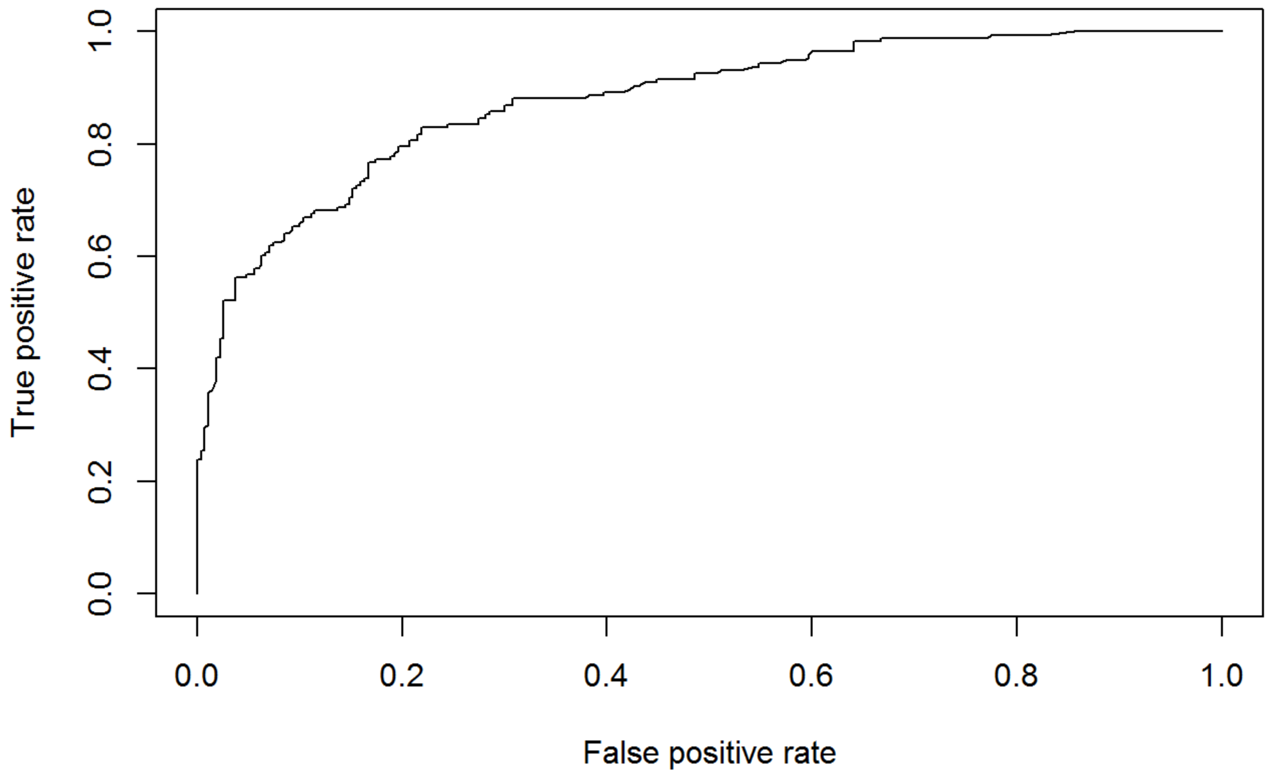
Evaluating Performance

We would like create more plots to visualize performance and evaluation. First, we will look at the confusion matrix which plots your false positive, false negative, true positive, and false positive rates. True Negatives - Case correctly predicted to be death False Negatives - Case predicted to be death, but actually survived False Positives - Case predicted to be survived, but actually death True Positives - Case correctly predicted to be survival We used the confusionMatrix command from the caret package, so our output gives us the confusion matrix in addition to other useful performance metrics.

```
# Plots to visualize performance & evaluation
# Confusion matrix
confusionmat <- confusionMatrix(pred[[1]], test$Survived)
```

Next we will look at the ROC curve and the AUC.

```
# ROC Curve + AUC
# Predictions are your continuous predictions of the classification, the labels are the binary
 truth for each variable
predroc <- data.frame(predict(fit, test, type = "prob"))
pr <- prediction(predroc[2], test$Survived)
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf)
```




```
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

```
## [1] 0.8777883
```

Both the ROC curve and AUC show that the model is an okay predictor of passenger survival. There are most likely other interations, like sex and age and age and class, that we are not accounting for in our model. In a future run, we could include these interactions in a dummy variable and hopefully improve our accuracy rate.