Homework 5

Jacob Plaza

11/8/2022

Loading and cleaning the data

```
train = read.csv(file = "/home/bultok/Documents/School/Senior Year/STAT 388/Homework 5/titanic/train.csv")
test = read.csv(file = "/home/bultok/Documents/School/Senior Year/STAT 388/Homework 5/titanic/test.csv")

train <- train[, colSums(is.na(train)) == 0]
```

First, I loaded in the training and test sets, and cleaned up the NA values in them.

Making “Survived” & “Sex” into factors to build the classification tree

```
Alive = factor(ifelse(train$Survived <= 0, "No", "Yes"))
train$Sex = as.factor(train$Sex)
train$Embarked = as.factor(train$Embarked)

test$Alive = "NA"
test$Sex = as.factor(test$Sex)
```

So that the classification tree would run, I also made Alive into a factor of whether or not someone survived. If they survived, Alive takes on the value of “yes”, and “no” if they died.

I also tried to convert variables which were of the character class into factors, in order to avoid future errors. Since some of the variables became factors with more than 32 levels, though, (such as the cabin) it wouldn’t make much sense to make them into factors so I left them as characters.

Building the classification tree

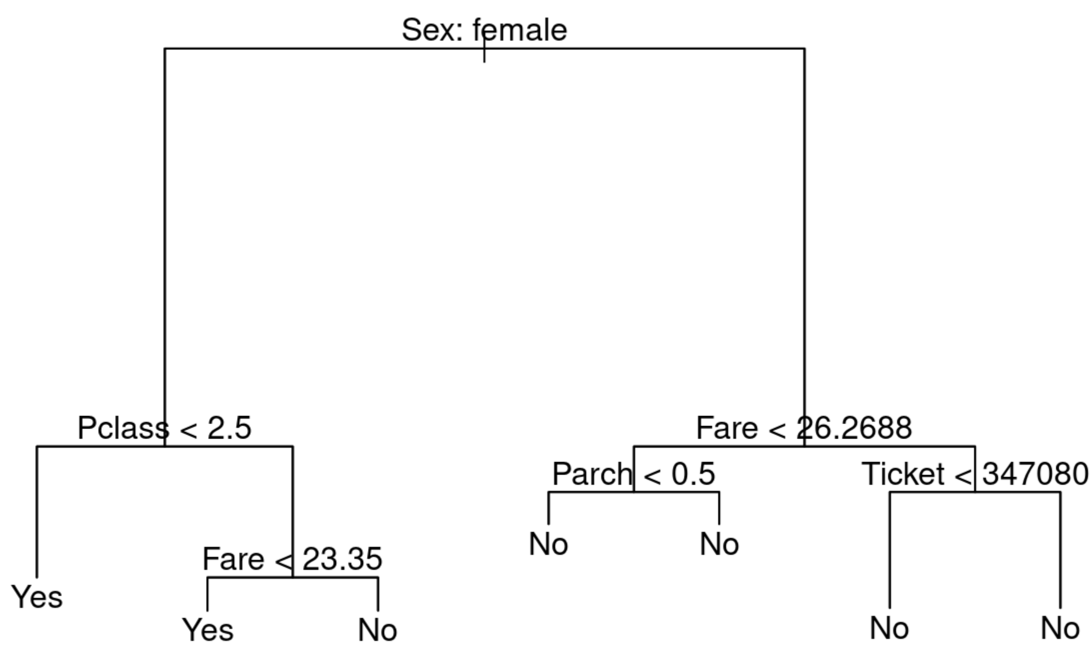
```
library(tree)
tree.titanic = tree(Alive ~ . - Survived, data = train)
```

```
## Warning in tree(Alive ~ . - Survived, data = train): NAs introduced by coercion
```

```
summary(tree.titanic)
```

```
##
## Classification tree:
## tree(formula = Alive ~ . - Survived, data = train)
## Variables actually used in tree construction:
## [1] "Sex"    "Pclass" "Fare"   "Parch"  "Ticket"
## Number of terminal nodes:  7
## Residual mean deviance:  0.7649 = 676.1 / 884
## Misclassification error rate: 0.1897 = 169 / 891
```

```
plot(tree.titanic)
text(tree.titanic, pretty = 0)
```



I build the first classification tree, which has a misclassification error rate of 0.2682.

Pruning

```
set.seed(7)
cv.titanic = cv.tree(tree.titanic, FUN = prune.misclass)
```

```
## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion
```

```
## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion
```

```
## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion
```

```
## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion
```

```
## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion
```

```
## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion
```

```
## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion
```

```
## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion
```

```
## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion
```

```
## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion
```

```
## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion
```

```
## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion
```

```
## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion
```

```
## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion
```

```
## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion
```

```
## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion
```

```
## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion
```

```
## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion
```

```
## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion
```

```
## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion
```

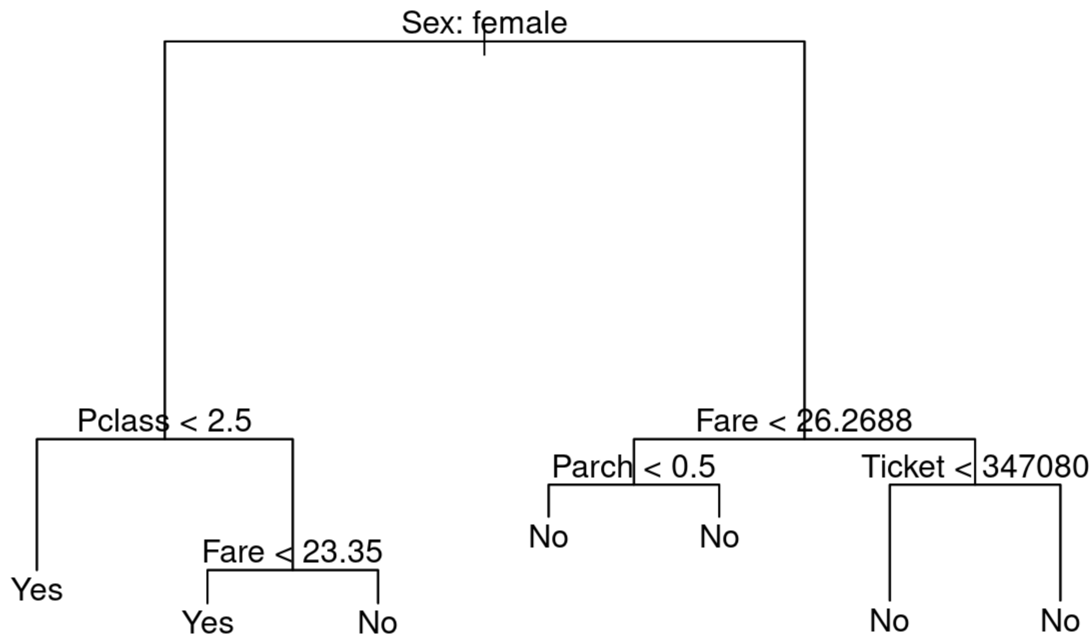
```
names(cv.titanic)
```

```
## [1] "size"   "dev"    "k"      "method"
```

```
cv.titanic
```

```
## $size
## [1] 7 4 2 1
##
## $dev
## [1] 175 175 190 342
##
## $k
## [1]  -Inf   0.0  10.5 152.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
prune.titanic = prune.misclass(tree.titanic, best = 7)
plot(prune.titanic)
text(prune.titanic, pretty = 0)
```




```
summary(prune.titanic)
```

```
##
## Classification tree:
## tree(formula = Alive ~ . - Survived, data = train)
## Variables actually used in tree construction:
## [1] "Sex"    "Pclass" "Fare"   "Parch"  "Ticket"
## Number of terminal nodes:  7
## Residual mean deviance:  0.7649 = 676.1 / 884
## Misclassification error rate: 0.1897 = 169 / 891
```

I then prune the tree, and preform k-fold cross-validation using cv.tree() to select the best level of complexity. From this, the tree with 7 terminal nodes yields the less number of cross-validation errors, at only 176 misclassifications.

So, my final treehas a error rate of 0.1897.

Growing a random forest

```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

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

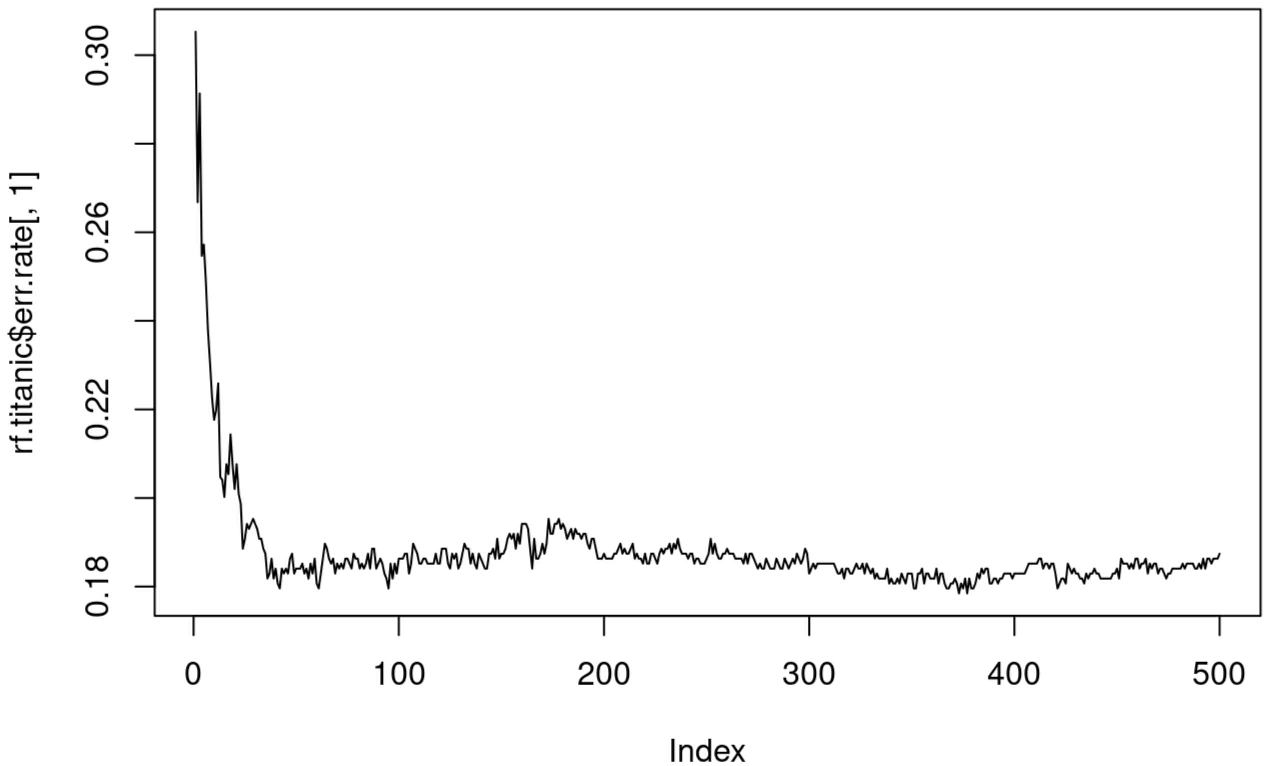
```
set.seed(1)
forest_train = subset(train, select = -c(Survived))

rf.titanic = randomForest(Alive ~ ., data = forest_train, mtry=sqrt(3), importance =
TRUE)
importance(rf.titanic)
```

```
##                    No       Yes MeanDecreaseAccuracy MeanDecreaseGini
## PassengerId -1.761130  2.440821            0.1965728         38.30653
## Pclass      22.252026 14.249955           26.0650419         24.82170
## Name         4.332171  8.386909            8.5309098         41.16696
## Sex         56.617870 66.487796           71.3035893         89.40287
## SibSp       13.853673  1.051318           13.2004496         11.10720
## Parch       19.517679  4.141440           19.2106533         13.77555
## Ticket      21.291721  5.396410           22.8779010         50.60357
## Fare        15.948552 18.940253           25.8055258         46.62750
## Cabin       18.818968  3.690834           19.7931030         27.63912
## Embarked     5.539661  9.718955           11.3017848         10.05976
```

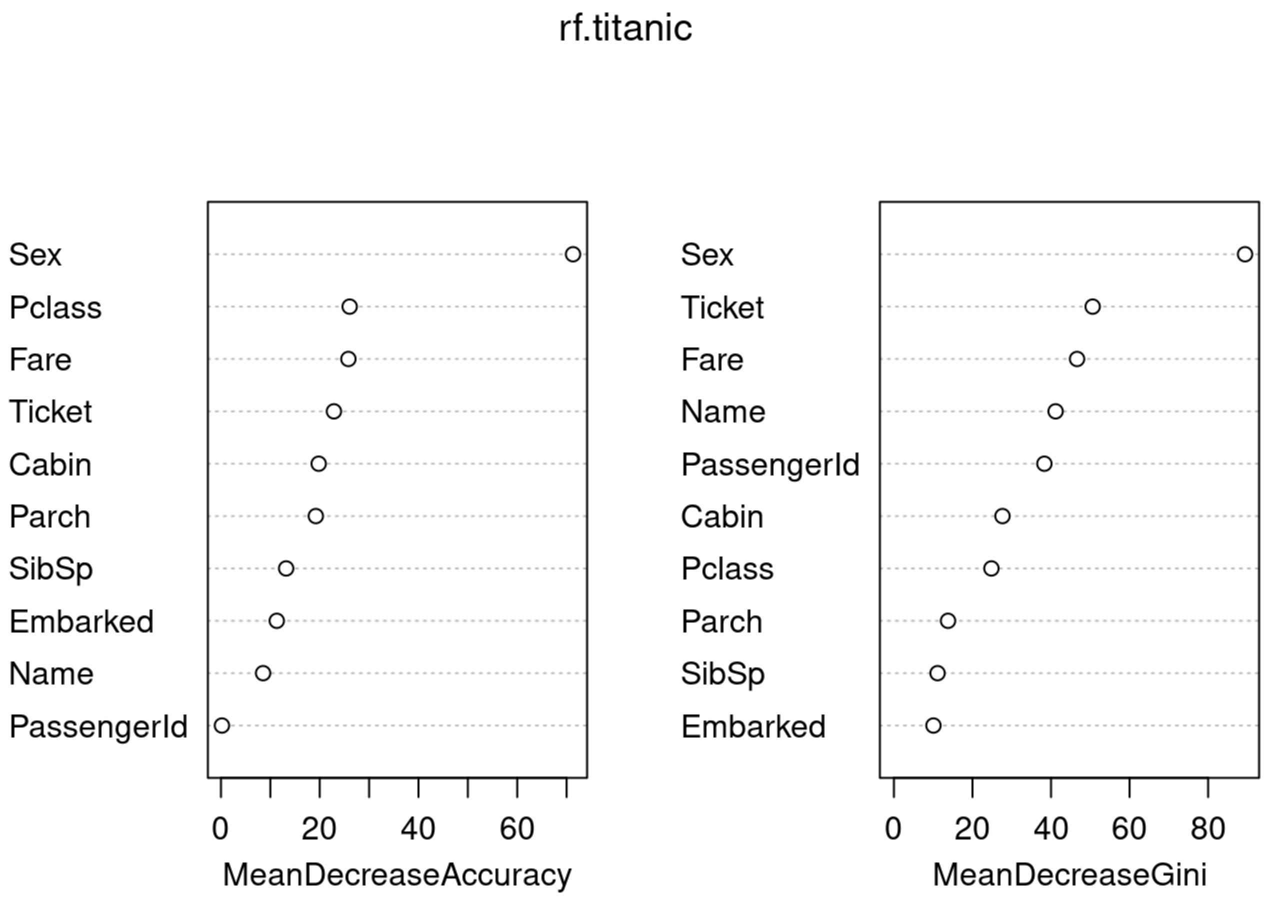
```
set.seed(1)
rf.titanic$err.rate[1,]
```

```
##       OOB        No       Yes
## 0.3052960 0.1518325 0.5307692
```

```
plot(rf.titanic$err.rate[,1], type = "l")
```




```
varImpPlot(rf.titanic)
```



The out-of-bag error rate reaches its height at a little over 0.26, before decreasing to a little below 0.18, and staying between 0.18 and 0.20.