RANDOM FORESTS

We saw in the chapter on decision trees that a single tree did not perform very well on the PenDigits dataset. In fact, the validation misclassification rate was around 25%.

Let's see if we can improve this model by ensembling more than one tree. Since the procedure for creating a decision tree is a definitive one (once we've decided on a measure of impurity and all the other specific parameters like maximum depth and complexity), we'll have to alter our inputs to the tree procedure. This is what a random forest does, it alters the data in two ways:

1. Select a subset of observations from the original training set (generally sample with replacement)
2. Select a subset of variables from the original training set

This subset of data is then fed into the decision tree algorithm any number of times, and the resulting trees each give a vote as to which class each observation belongs. The class with the most votes becomes the predicted class for the random forest.

```
> load("PenDigits.Rdata")
> library('randomForest')
> rf = randomForest(digit ~ .-digit, data=train, ntree=50, type='class')
```

We can then examine the confusion matrix to see how our model predicts each class:

```
> rf$confusion
```

```
    0   1   2   3   4   5   6   7   8   9 class.error
0 774   1   0   0   3   0   0   0   1   1 0.007692308
1   0 760  15   3   0   0   0   0   0   1 0.024390244
2   0  12 766   1   0   0   0   1   0   0 0.017948718
3   0   1   1 712   0   2   0   3   0   0 0.009735744
4   2   1   0   0 774   0   1   0   0   2 0.007692308
5   0   0   0   3   0 711   0   0   2   4 0.012500000
6   1   0   1   0   1   1 716   0   0   0 0.005555556
7   0   4   1   1   0   0   0 769   3   0 0.011568123
8   0   0   0   0   0   0   0   2 716   1 0.004172462
9   1   0   0   1   2   0   0   0   0 715 0.005563282
```

The classwise error rates are extremely small for the entire model! rf$err.rate will show the progression of misclassification rates as each individual tree is added to the forest for each class and overall (on the out of bag (OOB) data that was remaining after the data was sampled to train the tree).

```
> #rf$err.rate
> head(rf$err.rate)
```

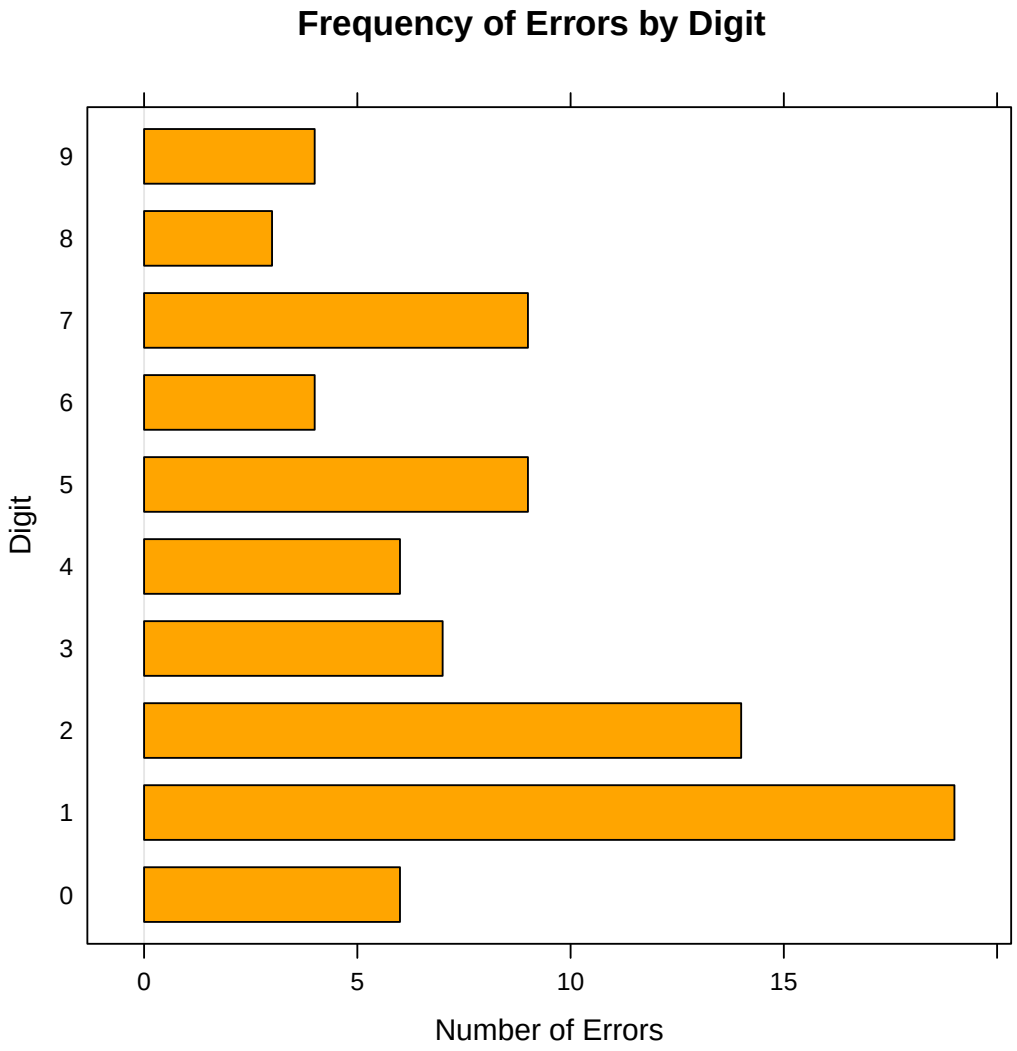
```
           OOB          0          1          2          3          4
[1,] 0.06099343 0.03484321 0.09025271 0.07067138 0.08058608 0.04561404
[2,] 0.06125643 0.02500000 0.10769231 0.08149780 0.06053812 0.03765690
[3,] 0.05413156 0.03589744 0.08510638 0.06866197 0.06607143 0.03061224
[4,] 0.05408845 0.04000000 0.08936826 0.07561728 0.06016260 0.02710843
[5,] 0.04829630 0.03295129 0.08595989 0.04680851 0.04961240 0.03262411
[6,] 0.04591473 0.03155007 0.07756233 0.04526749 0.04302671 0.02876712
             5          6          7          8          9
[1,] 0.04964539 0.04850746 0.05555556 0.07258065 0.06415094
[2,] 0.06320542 0.03472222 0.06666667 0.07125891 0.06763285
[3,] 0.04036697 0.03690037 0.05235602 0.05751391 0.06990291
[4,] 0.04854369 0.03636364 0.04658385 0.05481728 0.06260575
[5,] 0.05120482 0.03384615 0.04564907 0.04388715 0.06037152
[6,] 0.05224964 0.03387334 0.04395604 0.04491018 0.05864662
```

```
> rf$err.rate[50,]
```

```
       OOB          0          1          2          3          4
0.010808647 0.007692308 0.024390244 0.017948718 0.009735744 0.007692308
          5          6          7          8          9
0.012500000 0.005555556 0.011568123 0.004172462 0.005563282
```

The final misclassification rate on the out of bag data was just 0.01 using the ensemble of 50 trees! We might want to look at what digits we are struggling with to see if there is any pattern.

```
> library(lattice)
> wrong=train[rf$predicted!=train$digit,]
> barchart(wrong$digit, main = 'Frequency of Errors by Digit',
+          xlab = 'Number of Errors', ylab='Digit', col = 'orange')
```



Finally we should check our random forest model on the validation model as a final test of performance and screen for overfitting.

```
> vscores = predict(rf,test,type='class')
> cat('Validation Misclassification Rate:', sum(vscores!=test$digit)/nrow(test))
```

```
Validation Misclassification Rate: 0.03630646
```