Regression Tree and Random Forest

Using the same crime data set from statsci and in previous notebooks, build the best model using regression tree and random forest model.

```
library(tidyverse)
library(randomForest)
library(tree)
library(rpart.plot)
library(caret)
library(pROC)

#setwd("/Users/Ryan/Desktop/DS")

crime_data = read.table("uscrime.txt.",
                        sep="",
                        fill=FALSE,
                        strip.white=TRUE,
                        header = TRUE)
```

Loading in the many packages that will be used throughout the problem. Next is setting the working directory and reading in the crime data that was given to us. Once we build the model, we are trying to predict the crime rate given those values about the city and then see how well our model does compared to the PCA model.

Regression Tree

```
crime_tree <- tree(Crime~. , data = crime_data)
summary(crime_tree)
```

```
##
## Regression tree:
## tree(formula = Crime ~ ., data = crime_data)
## Variables actually used in tree construction:
## [1] "Po1" "Pop" "LF"  "NW"
## Number of terminal nodes:  7
## Residual mean deviance:  47390 = 1896000 / 40
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -573.900  -98.300   -1.545    0.000  110.600  490.100
```

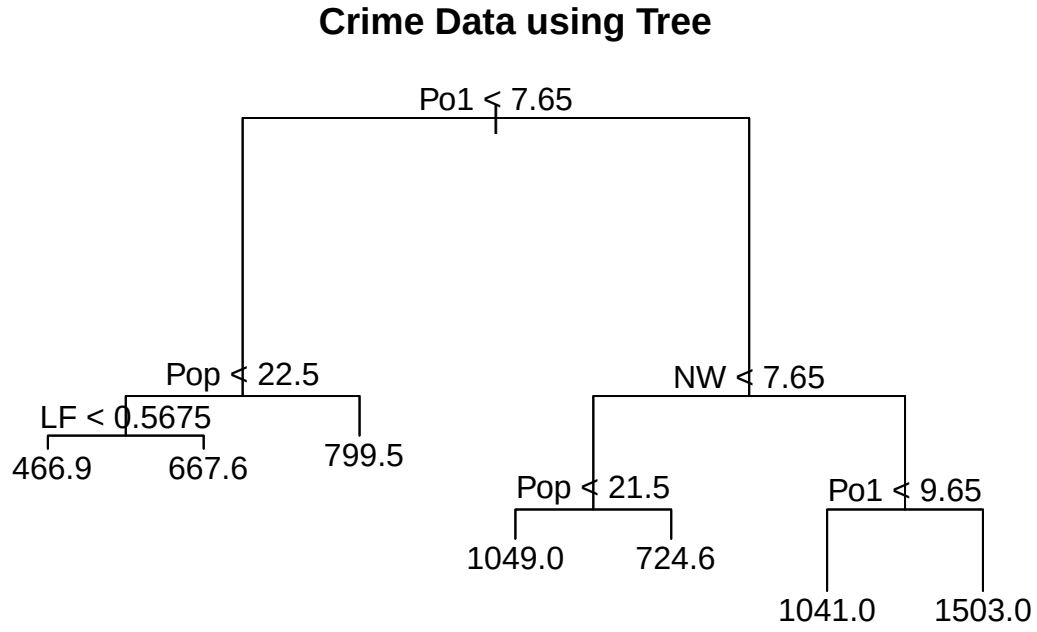
```
crime_tree$frame #how the tree is split
```

```
##       var  n        dev      yval splits.cutleft splits.cutright
```

```
## 1     Po1 47 6880927.66  905.0851          <7.65          >7.65
## 2     Pop 23  779243.48  669.6087          <22.5          >22.5
## 4      LF 12  243811.00  550.5000        <0.5675        >0.5675
## 8  <leaf>  7   48518.86  466.8571
## 9  <leaf>  5   77757.20  667.6000
## 5  <leaf> 11  179470.73  799.5455
## 3      NW 24 3604162.50 1130.7500          <7.65          >7.65
## 6     Pop 10  557574.90  886.9000          <21.5          >21.5
## 12 <leaf>  5  146390.80 1049.2000
## 13 <leaf>  5  147771.20  724.6000
## 7     Po1 14 2027224.93 1304.9286          <9.65          >9.65
## 14 <leaf>  6  170828.00 1041.0000
## 15 <leaf>  8 1124984.88 1502.8750
```

```
#there are 7 leaves

#plot the tree
plot(crime_tree)
text(crime_tree)
title('Crime Data using Tree')
```




```
prune.tree(crime_tree)$size
```

```
## [1] 7 6 5 4 3 2 1
```

```r
prune.tree(crime_tree)$dev
```

```
## [1] 1895722 2013257 2276670 2632631 3364043 4383406 6880928
```

```r
#we shouldnt prune because the deviance is actually getting larger with less leafs
#however this might change after cross validation
#7 leaves: 1895722
#6 leaves: 2013257
```

Building the regression tree model is very similar to the linear regression model. Frame shows us how many leaves in the tree we have, this model split it into 7 leaves. Next, we plot the tree function so you can get a sense of where the splits are at. Then we look at the number of leaves (size) and the deviance within those leaves. From the output it seems like 7 leaves gives us the best split, but we will check with corss validation. You can see that the deviance is about 100,000 less with 7 leaves compared to 6 leaves but this might be overfitting.

```r
set.seed(1982)
prune_tree <- cv.tree(object = crime_tree, FUN = prune.tree)
prune_tree$size
```

```
## [1] 7 6 5 4 3 2 1
```

```r
prune_tree$dev
```

```
## [1] 7973708 8024246 8359077 8150162 8809925 7163386 7887202
```

```r
which.min(prune_tree$dev)
```

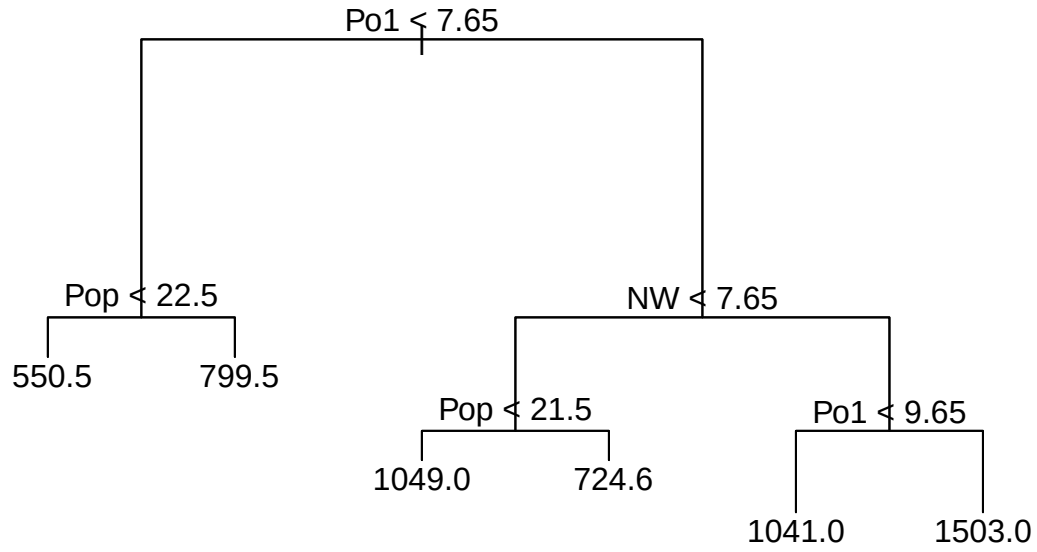
```
## [1] 6
```

```r
# it looks like 6 leaves is the best one after cross validation

crime_tree_prune <- prune.tree(crime_tree, best = 6)
summary(crime_tree_prune)
```

```
##
## Regression tree:
## snip.tree(tree = crime_tree, nodes = 4L)
## Variables actually used in tree construction:
## [1] "Po1" "Pop" "NW"
## Number of terminal nodes:  6
## Residual mean deviance:  49100 = 2013000 / 41
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -573.900  -99.520   -1.545    0.000  122.800  490.100
```

```r
plot(crime_tree_prune)
text(crime_tree_prune)
```




```
prune_predict <- predict(crime_tree_prune, data = crime_data[,1:15])
RSS <- sum((prune_predict - crime_data[,16])^2)
TSS <- sum((crime_data[,16] - mean(crime_data[,16]))^2)
R <- 1 - RSS/TSS
R #0.7074
```

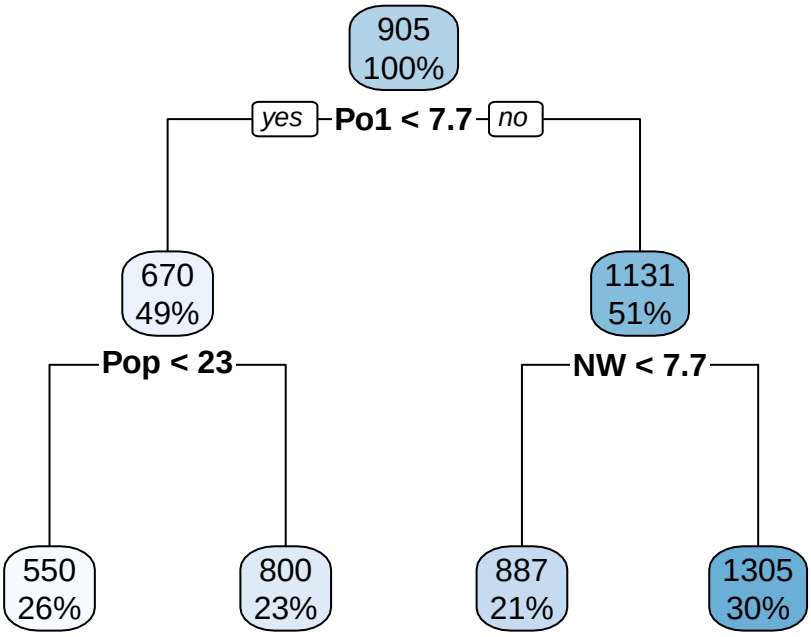
```
## [1] 0.7074149
```

First we cross validate the tree and prune it using the prune.tree function. The output is a lot different than when we ran the tree just once. It seems that 6 leaves give us the smallest deviance so we will use 6 leaves going forward. Then you take the original model and prune it using six leaves. We plot it again and compute the R-squared value which is 0.7074. Last week I got the value 0.645 so it seems that pruning the tree did better.

```
# i used rpart since its a regression tree
rr <- rpart(Crime~.,
            data = crime_data,
            control = (minsplit = seq(2,10,1)))

bestcp <- rr$cptable[which.min(rr$cptable[,"xerror"]),"CP"]
rr.pruned <- prune(rr, cp = bestcp)

rpart.plot(rr)
```




```
predict1 <- predict(rr.pruned, data = crime_data[,1:15])
RSS <- sum((predict1 - crime_data[,16])^2)
TSS <- sum((crime_data[,16] - mean(crime_data[,16]))^2)
R2 <- 1 - RSS/TSS
R2 #0.3629
```

```
## [1] 0
```

I also tried the other method for creating a regression tree which is rpart. However, how I determined the fit seemed to not work very well. My R-sqaured dropped significantly so I'll need more time to figure out how to optimize it even better.

Random Forest Model

```
num_pred <- 4
#recommendation is 1+log(n) or n/3 where n is the number of predictors

crime_rf <- randomForest(Crime ~.,
                         data = crime_data,
                         mtry = num_pred,
                         importance = TRUE)
crime_rf
```

```
##
## Call:
##  randomForest(formula = Crime ~ ., data = crime_data, mtry = num_pred,      importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 84010.19
##                     % Var explained: 42.62
```

```
importance(crime_rf)
```

```
##           %IncMSE IncNodePurity
## M       2.8019675     193186.88
## So      1.6920124      22600.91
## Ed      2.8279502     227413.64
## Po1    11.8991838    1168215.39
## Po2    10.6918123    1107643.82
## LF      3.3185404     304278.29
## M.F     0.6946095     317588.34
## Pop     1.2853999     396816.91
## NW      8.1411760     509295.71
## U1     -0.8129753     152044.67
## U2      3.3245800     204717.33
## Wealth  4.0216709     678249.54
## Ineq    1.3656841     197957.76
## Prob    9.2267276     788031.27
## Time    1.7803958     226090.37
```

```
#Po1, Po2, and NW

rf_predict <- predict(crime_rf, data=crime_data[,-16])
RSS <- sum((rf_predict - crime_data[,16])^2)
R3 <- 1 - RSS/TSS
R3
```

```
## [1] 0.4261705
```

The next part of the problem asked us to make a randomForest of the data. I orginally used mtry = 5 but that gave me a much lower R^2 (about 0.34) so after trying a few different options decided on 4. From the importance function the best predictors were Po1, Po2, NW, and Prob. We don't need to run cross validation because it is a randomForest.