Application

Refer to the Air Quality data described previously, and the analyses we have done with Ozone as the response variable, and the five explanatory variables (including the two engineered features).

Use regression trees to model the relationship between Ozone and all five explanatories as specified below.:

1. Fit a regression tree using only Temp and Wind, using cp=0.

(a) **Print out the cp table.**

```
# Title: STAT 452 Exercise 8 L13Q1
# Author: Injun Son
# Date: October 31, 2020

library(dplyr)
library(MASS)   # For ridge regression
library(glmnet) # For LASSO
library(nnet) # Fits neural net models
library(rgl)
library(rpart)
library(rpart.plot)
source("Helper Functions.R")
data = na.omit(airquality[, 1:4])
data$TWcp = data$Temp*data$Wind
data$TWrat = data$Temp/data$Wind

set.seed(8598176)

# 1. Fit a regression tree using only Temp and Wind, using cp=0.
# (a) Print out the cp table.
# i. According to minimum CV, what value of cp should be used for optimal
# pruning, and what is the relative CV error at this value?

fit.tree = rpart(Ozone ~ Temp + Wind, data = data, cp=0)

### Get the CP table
info.tree = fit.tree$cptable
```

```
           CP nsplit rel error    xerror      xstd
1 0.484385708      0 1.0000000 1.0068193 0.1705465
2 0.097982838      1 0.5156143 0.5469021 0.1798443
3 0.057062146      2 0.4176315 0.5690217 0.1811551
4 0.019726902      3 0.3605693 0.5260658 0.1525831
5 0.018716394      4 0.3408424 0.5185840 0.1519412
6 0.005142355      5 0.3221260 0.4861408 0.1409565
7 0.004575931      6 0.3169837 0.4878673 0.1409443
8 0.001882628      7 0.3124077 0.4930603 0.1410539
9 0.000000000      9 0.3086425 0.4901064 0.1410692
```

i. According to minimum CV, **what value of cp should be used for optimal pruning, and what is the relative CV error at this value?**

```
### We have to prune the tree manually. First, get the CP value with minimum
### CV error
ind.min = which.min(info.tree[,"xerror"])
CP.min.raw = info.tree[ind.min, "CP"]
```

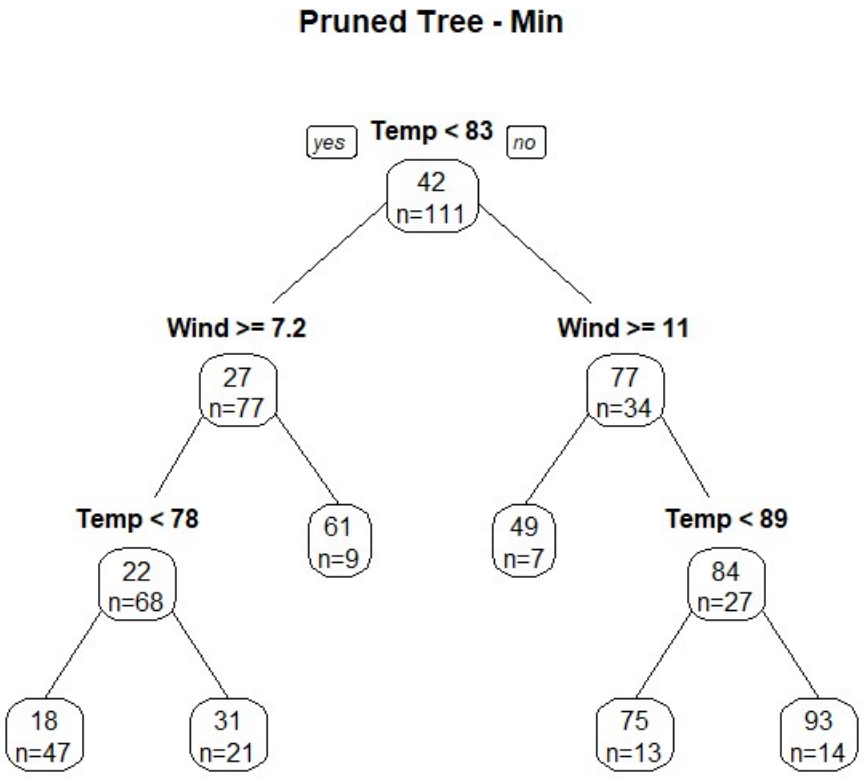
```
> CP.min.raw
[1] 0.005142355
```

Relative CV error is 0.3221260

(b) **Plot both the min-CV pruned tree and the 1SE pruned tree.**
i. Interpret the first split on the root node: **What variable and split location are used is used, how many observations go each way, and what are the new means in each node?**

```
#########################################################
# (b) Plot both the min-CV pruned tree and the 1SE pruned tree.
# i. Interpret the first split on the root node: What variable and split location
# are used is used, how many observations go each way, and what are
# the new means in each node?

### It's best to average the minimum CP value with the one from the row above
### using the geometric mean (i.e. multiply them together, then square root).
### If we implement this procedure directly, and the minimum CP value is in
### the first row, our code will probably give an error. We should write our
### code so that it can cope with this weird situation. This attitude is
### called defensive programming, and it is a very good habit to practice.

### Check if minimum CP value is in row 1. We can do this using an if-else
### statement. See this tutorial's video for details.
if(ind.min == 1){
  ### If minimum CP is in row 1, store this value
  CP.min = CP.min.raw
} else{
  ### If minimum CP is not in row 1, average this with the value from the
  ### row above it.

  ### Value from row above
  CP.above = info.tree[ind.min-1, "CP"]

  ### (Geometric) average
  CP.min = sqrt(CP.min.raw * CP.above)
}

### We now have our chosen CP value. We can prune our tree using the prune()
### function. The first input to prune() is the tree object, and we set "cp"
### equal to the CP value where we want to prune
fit.tree.min = prune(fit.tree, cp = CP.min)

### Next, we want to prune using the 1SE rule. Fortunately, the CP table
### gives us the CV standard error. First, find the minimum CV error plus 1
### standard error
err.min = info.tree[ind.min, "xerror"]
se.min = info.tree[ind.min, "xstd"]
threshold = err.min + se.min

```

```r
### Next, get the smallest tree with CV error below our threshold.
### Note: We limit our search to only trees which are no larger than our min CV
###       error tree.
ind.1se = min(which(info.tree[1:ind.min,"xerror"] < threshold))

### Get the corresponding CP value, averaging if necessary
CP.1se.raw = info.tree[ind.1se, "xerror"]
if(ind.1se == 1){
  ### If best CP is in row 1, store this value
  CP.1se = CP.1se.raw
} else{
  ### If best CP is not in row 1, average this with the value from the
  ### row above it.

  ### Value from row above
  CP.above = info.tree[ind.1se-1, "CP"]

  ### (Geometric) average
  CP.1se = sqrt(CP.1se.raw * CP.above)
}

### Prune the tree
fit.tree.1se = prune(fit.tree, cp = CP.1se)

##############################################################################
### A nice feature of tree models is that they make great plots. Let's  ###
### plot the full tree, and both pruned trees. We can plot trees using  ###
### the prp() function from the rpart.plot package.                     ###
##############################################################################
### The prp() function has many inputs We will just use two: type and extra.
### Setting both of these inputs to 1 gives a nice looking plot. Since
### prp() makes a plot, we can set the title using main. We also
### have to provide the fitted tree object which is being plotted.
prp(fit.tree, type = 1, extra = 1, main = "Full Tree")
prp(fit.tree.min, type = 1, extra = 1, main = "Pruned Tree - Min")
prp(fit.tree.1se, type = 1, extra = 1, main = "Pruned Tree - 1SE")
```




```
> fit.tree.min
n= 111

node), split, n, deviance, yval
      * denotes terminal node

 1) root 111 121801.900 42.09910
   2) Temp< 82.5 77  42143.250 26.77922
     4) Wind>=7.15 68  10886.750 22.25000
       8) Temp< 77.5 47   3863.404 18.27660 *
       9) Temp>=77.5 21   4620.571 31.14286 *
     5) Wind< 7.15 9  19322.000 61.00000 *
   3) Temp>=82.5 34  20659.560 76.79412
     6) Wind>=10.6 7   1183.429 48.71429 *
     7) Wind< 10.6 27  12525.850 84.07407
      14) Temp< 88.5 13   7307.231 74.53846 *
      15) Temp>=88.5 14   2938.929 92.92857 *
```

Temp variable was used in the root node and 77 variables that Temp<82.5 goes left and 34 variables that Temp >=82.5 goes right. The new mean is 26.77 for variables that Temp< 82.5 and the new mean is 76.79

Pruned Tree - 1SE


```
> fit.tree.1se
n= 111

node), split, n, deviance, yval
      * denotes terminal node

1) root 111 121801.9 42.0991 *
```

1SE pruned tree didn’t split the data.