Lab7

2023-04-05

```r
#Demonstrating convergence in incomes across racial groups using all-race/gender model
#set seed
HUID <- 21519588
set.seed(HUID)

#Demonstrating model across two generations
#Gen 1-2
parents_rank <- 57.9
kids_rank <- 33.31 + 0.351 * parents_rank
kids_rank
```

```
## [1] 53.6329
```

```r
#Gen 2-3
parents_rank = kids_rank
kids_rank = 33.31 + 0.351 * parents_rank
kids_rank
```

```
## [1] 52.13515
```

```r
#Iterating across multiple generations
generations <- seq(1,7,1)

parents_rank_white = 57.9
parents_rank_black = 32.7

#white gen for loop
for(i in generations){
  kids_rank <- 33.31 + 0.351 * parents_rank_white
  print(paste0("In generation ", i, ", parent_rank = ", parents_rank_white, ", child_rank = ", kids_ranl
  parents_rank_white <- kids_rank
}
```

```
## [1] "In generation 1, parent_rank = 57.9, child_rank = 53.6329"
## [1] "In generation 2, parent_rank = 53.6329, child_rank = 52.1351479"
## [1] "In generation 3, parent_rank = 52.1351479, child_rank = 51.6094369129"
## [1] "In generation 4, parent_rank = 51.6094369129, child_rank = 51.4249123564279"
## [1] "In generation 5, parent_rank = 51.4249123564279, child_rank = 51.3601442371062"
## [1] "In generation 6, parent_rank = 51.3601442371062, child_rank = 51.3374106272243"
## [1] "In generation 7, parent_rank = 51.3374106272243, child_rank = 51.3294311301557"
```

```r
#black gen for loop
for(i in generations){
  kids_rank <- 33.31 + 0.351 * parents_rank_black
  print(paste0("In generation ", i, ", parent_rank = ", parents_rank_black, ", child_rank = ", kids_ranl
  parents_rank_black <- kids_rank
}
```

```
## [1] "In generation 1, parent_rank = 32.7, child_rank = 44.7877"
## [1] "In generation 2, parent_rank = 44.7877, child_rank = 49.0304827"
## [1] "In generation 3, parent_rank = 49.0304827, child_rank = 50.5196994277"
## [1] "In generation 4, parent_rank = 50.5196994277, child_rank = 51.0424144991227"
## [1] "In generation 5, parent_rank = 51.0424144991227, child_rank = 51.2258874891921"
## [1] "In generation 6, parent_rank = 51.2258874891921, child_rank = 51.2902865087064"
## [1] "In generation 7, parent_rank = 51.2902865087064, child_rank = 51.312890564556"
```

Using the all-race/gender model, white and black inter-generational mobility outcomes converge around gen 7 at a rank of about 51.3. But we know that this is incorrect; let's find the steady state prediction for Black and Hispanic children using their respective rank-rank models:

```r
#Steady state for Black children
generations <- seq(1,7,1)
parents_rank_black = 32.7

#black gen for loop
for(i in generations){
  kids_rank <- 25.4 + 0.28 * parents_rank_black
  print(paste0("In generation ", i, ", parent_rank = ", parents_rank_black, ", child_rank = ", kids_rank
  parents_rank_black <- kids_rank
}
```

```
## [1] "In generation 1, parent_rank = 32.7, child_rank = 34.556"
## [1] "In generation 2, parent_rank = 34.556, child_rank = 35.07568"
## [1] "In generation 3, parent_rank = 35.07568, child_rank = 35.2211904"
## [1] "In generation 4, parent_rank = 35.2211904, child_rank = 35.261933312"
## [1] "In generation 5, parent_rank = 35.261933312, child_rank = 35.27334132736"
## [1] "In generation 6, parent_rank = 35.27334132736, child_rank = 35.2765355716608"
## [1] "In generation 7, parent_rank = 35.2765355716608, child_rank = 35.277429960065"
```

```r
#Steady state for Hispanic children
parents_rank_hisp = 36.17
for(i in generations){
  kids_rank <- 36.14 + 0.26 * parents_rank_hisp
  print(paste0("In generation ", i, ", parent_rank = ", parents_rank_hisp, ", child_rank = ", kids_rank)
  parents_rank_hisp <- kids_rank
}
```

```
## [1] "In generation 1, parent_rank = 36.17, child_rank = 45.5442"
## [1] "In generation 2, parent_rank = 45.5442, child_rank = 47.981492"
## [1] "In generation 3, parent_rank = 47.981492, child_rank = 48.61518792"
## [1] "In generation 4, parent_rank = 48.61518792, child_rank = 48.7799488592"
## [1] "In generation 5, parent_rank = 48.7799488592, child_rank = 48.822786703392"
## [1] "In generation 6, parent_rank = 48.822786703392, child_rank = 48.8339245428819"
## [1] "In generation 7, parent_rank = 48.8339245428819, child_rank = 48.8368203811493"
```

The steady state prediction for Black children is around 35.27 and for Hispanic children, 48.83.

Question 2 Cross-validation helps us avoid the overfit problem by addressing the bias-variance tradeoff in machine learning models. More complex models will eventually fit the noise of the training data, which causes the overfit problem. Cross-validation addresses that by evaluating a model's performance with different sets of training data taken from the original dataset. We can cross-validate a portion of the training data to find the optimal model complexity that minimizes RMSPE and over-fitting.

```r
#Implementing 5-fold cross-validation using two predictors

#Store predictor variables which all start with P_*
```

```r
vars <- colnames(training[,grep("^[P_]", names(training))])
vars
```

```
##   [1] "P_1"   "P_2"   "P_3"   "P_4"   "P_5"   "P_6"   "P_7"   "P_8"   "P_9"
##  [10] "P_10"  "P_11"  "P_12"  "P_13"  "P_14"  "P_15"  "P_16"  "P_17"  "P_18"
##  [19] "P_19"  "P_20"  "P_21"  "P_22"  "P_23"  "P_24"  "P_25"  "P_26"  "P_27"
##  [28] "P_28"  "P_29"  "P_30"  "P_31"  "P_32"  "P_33"  "P_34"  "P_35"  "P_36"
##  [37] "P_37"  "P_38"  "P_39"  "P_40"  "P_41"  "P_42"  "P_43"  "P_44"  "P_45"
##  [46] "P_46"  "P_47"  "P_48"  "P_49"  "P_50"  "P_51"  "P_52"  "P_53"  "P_54"
##  [55] "P_55"  "P_56"  "P_57"  "P_58"  "P_59"  "P_60"  "P_61"  "P_62"  "P_63"
##  [64] "P_64"  "P_65"  "P_66"  "P_67"  "P_68"  "P_69"  "P_70"  "P_71"  "P_72"
##  [73] "P_73"  "P_74"  "P_75"  "P_76"  "P_77"  "P_78"  "P_79"  "P_80"  "P_81"
##  [82] "P_82"  "P_83"  "P_84"  "P_85"  "P_86"  "P_87"  "P_88"  "P_89"  "P_90"
##  [91] "P_91"  "P_92"  "P_93"  "P_94"  "P_95"  "P_96"  "P_97"  "P_98"  "P_99"
## [100] "P_100" "P_101" "P_102" "P_103" "P_104" "P_105" "P_106" "P_107" "P_108"
## [109] "P_109" "P_110" "P_111" "P_112" "P_113" "P_114" "P_115" "P_116" "P_117"
## [118] "P_118" "P_119" "P_120" "P_121"
```

```r
#Create a training data frame with just predictors P_* and kfr_pooled_pooled_p25
training_subset <- subset(training, training==1, vars)
training_subset$kfr_pooled_pooled_p25 <- training[training$training==1,]$kfr_pooled_pooled_p25

#cross-validation
n <- nrow(training_subset)
K <- 5
B <- seq(1,20,1)

cv <- training_subset
cv$foldid <- rep(1:K,each=ceiling(n/K))[sample(1:n)]
OOS <- data.frame(fold=rep(NA,K*length(B) ),
                  squarederror=rep(NA,K*length(B) ),
                  maxdepth=rep(NA,K*length(B) ))

row <- 0

for(i in B){

  for(k in 1:K){

    row <- row + 1

    cvtrain <- subset(cv, foldid != k)

    cvfold <- subset(cv, foldid == k)

    cvtree <- rpart(kfr_pooled_pooled_p25 ~ P_12 + P_80,
                    data=cvtrain,
                    maxdepth = c(i),
                    cp=0)


    predfull <- predict(cvtree, newdata=cvfold)

    OOS$squarederror[row] <- sum((cvfold$kfr_pooled_pooled_p25 - predfull)^2)
```

```r
    OOS$maxdepth[row] <- i

    OOS$fold[row] <- k

  }

}

OOS
```

```
##     fold squarederror maxdepth
## 1      1     5384.794        1
## 2      2     4641.916        1
## 3      3     5746.422        1
## 4      4     5128.230        1
## 5      5     4337.194        1
## 6      1     4538.326        2
## 7      2     4154.131        2
## 8      3     5252.789        2
## 9      4     4591.880        2
## 10     5     4012.419        2
## 11     1     4400.510        3
## 12     2     3856.398        3
## 13     3     4755.413        3
## 14     4     4365.298        3
## 15     5     3910.760        3
## 16     1     3550.421        4
## 17     2     3854.592        4
## 18     3     4807.080        4
## 19     4     4273.982        4
## 20     5     3718.511        4
## 21     1     3481.735        5
## 22     2     4023.202        5
## 23     3     4583.083        5
## 24     4     4588.044        5
## 25     5     3525.753        5
## 26     1     3489.742        6
## 27     2     4071.722        6
## 28     3     4595.772        6
## 29     4     4537.064        6
## 30     5     3670.873        6
## 31     1     3490.964        7
## 32     2     4166.760        7
## 33     3     4584.770        7
## 34     4     4578.060        7
## 35     5     3918.333        7
## 36     1     3610.763        8
## 37     2     4198.802        8
## 38     3     4747.887        8
## 39     4     4864.464        8
## 40     5     4078.479        8
## 41     1     3681.679        9
## 42     2     4281.377        9
```

```
## 43     3     4973.341          9
## 44     4     4947.999          9
## 45     5     4128.395          9
## 46     1     3768.974         10
## 47     2     4300.370         10
## 48     3     5020.607         10
## 49     4     4993.543         10
## 50     5     4209.944         10
## 51     1     3797.818         11
## 52     2     4387.460         11
## 53     3     5022.392         11
## 54     4     4996.704         11
## 55     5     4205.684         11
## 56     1     3786.890         12
## 57     2     4391.268         12
## 58     3     5058.183         12
## 59     4     5000.416         12
## 60     5     4212.717         12
## 61     1     3791.447         13
## 62     2     4410.442         13
## 63     3     5060.055         13
## 64     4     5000.416         13
## 65     5     4221.388         13
## 66     1     3805.139         14
## 67     2     4410.442         14
## 68     3     5068.832         14
## 69     4     5000.416         14
## 70     5     4221.388         14
## 71     1     3805.139         15
## 72     2     4410.442         15
## 73     3     5082.219         15
## 74     4     5000.416         15
## 75     5     4221.388         15
## 76     1     3805.139         16
## 77     2     4410.442         16
## 78     3     5082.219         16
## 79     4     5000.416         16
## 80     5     4221.388         16
## 81     1     3805.139         17
## 82     2     4410.442         17
## 83     3     5082.219         17
## 84     4     5000.416         17
## 85     5     4221.388         17
## 86     1     3805.139         18
## 87     2     4410.442         18
## 88     3     5082.219         18
## 89     4     5000.416         18
## 90     5     4221.388         18
## 91     1     3805.139         19
## 92     2     4410.442         19
## 93     3     5082.219         19
## 94     4     5000.416         19
## 95     5     4221.388         19
## 96     1     3805.139         20
```

```
## 97     2     4410.442       20
## 98     3     5082.219       20
## 99     4     5000.416       20
## 100    5     4221.388       20
```

```
summary(OOS)
```

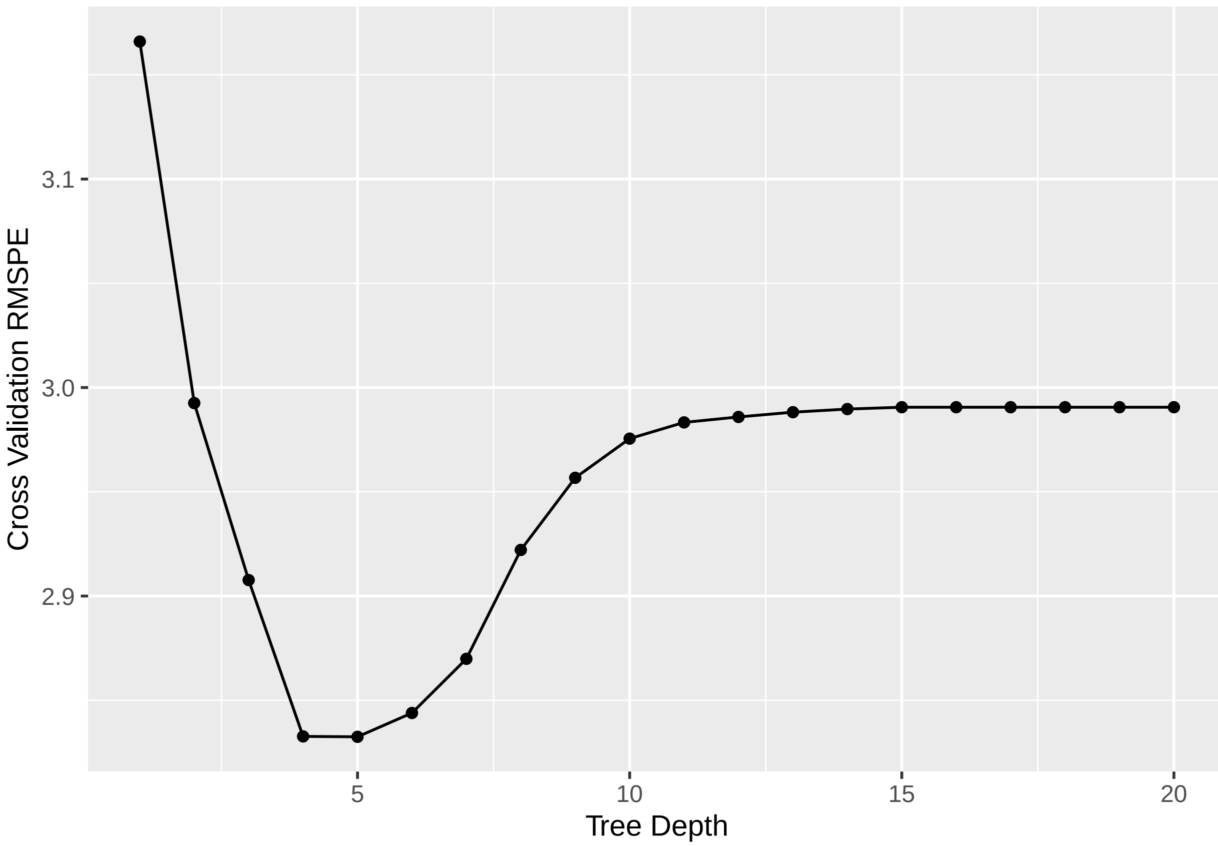
```
##       fold    squarederror     maxdepth
##  Min.   :1   Min.   :3482   Min.   : 1.00
##  1st Qu.:2   1st Qu.:4021   1st Qu.: 5.75
##  Median :3   Median :4396   Median :10.50
##  Mean   :3   Mean   :4414   Mean   :10.50
##  3rd Qu.:4   3rd Qu.:4994   3rd Qu.:15.25
##  Max.   :5   Max.   :5746   Max.   :20.00
```

```
ssr <- tapply(OOS$squarederror, OOS$maxdepth, sum)
ssr <- as.data.frame(ssr)
ssr$maxdepth <- seq(1,20,1)
ssr
```

```
##          ssr maxdepth
## 1  25238.56        1
## 2  22549.55        2
## 3  21288.38        3
## 4  20204.59        4
## 5  20201.82        5
## 6  20365.17        6
## 7  20738.89        7
## 8  21500.40        8
## 9  22012.79        9
## 10 22293.44       10
## 11 22410.06       11
## 12 22449.47       12
## 13 22483.75       13
## 14 22506.22       14
## 15 22519.60       15
## 16 22519.60       16
## 17 22519.60       17
## 18 22519.60       18
## 19 22519.60       19
## 20 22519.60       20
```

```
ssr$rmse <- sqrt(ssr$ssr / nrow(training))

ggplot(ssr, aes(x=maxdepth,y=rmse)) +
  geom_point() +
  geom_line() +
  labs(y = "Cross Validation RMSPE",
       x = "Tree Depth")
```




```
cv_optimal_depth = ssr$maxdepth[which.min(ssr$rmse)]
cv_optimal_depth
```

```
## [1] 5
```

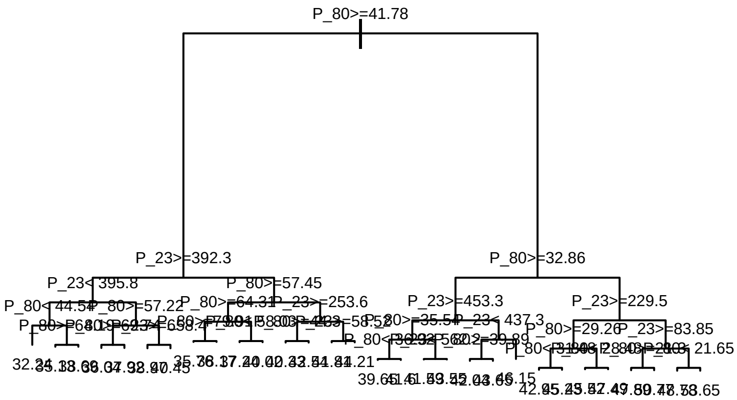
Question 3b The optimal tree depth for this training dataset is 5

Question 3c I am using the following two predictors: P_12 (Total Violent and Property Crimes Rate) and P_80 (Percent of Children Eligible for Free Lunch (Persons < 18 Years)).

```
#Using full training dataset to estimate tree of depth 5

tree <- rpart(kfr_pooled_pooled_p25 ~ P_23 + P_80,
                      data=training_subset,
                      maxdepth = cv_optimal_depth,
                      cp=0)

#visualize tree
plot(tree, margin = 0.2)
text(tree, cex = 0.5)
```




```r
#Calculate predictions for all rows in training sample
y_train_predictions_tree <- predict(tree, newdata=training_subset)
```

Question 4 Random forests improve upon decision trees in two distinct ways. First, they apply bagging to build a series of trees which are each trained on a subset of the original data. The average of each series' RMPSE is taken to determine the most accurate prediction model. Bagging averages across a large number of trees to cancel out the training data noise and left with real signal instruction. The other way is through input randomization. This reduces the correlation between trees, which improves model accuracy.