Lab6

2023-03-28

Question 1 When conducting prediction applications, we split our data into "test" and "training" datasets to predict what an outcome will be for a new observation as accurately as possible. We start with existing data where all outcomes have already been observed, and split it into training and test data. We fit a statistical model using the assigned training data. We then use this fitted model to predict an outcome variable of interest for "test" data to evaluate this model when making out of sample predictions.

```
#Question 2a: Creating Test and Training subsamples.

#Set the seed
HUID <- 21519588
set.seed(HUID)

#Assign a random number to each observation
df$random_number <- runif(length(df$cz))
view(df)

#2b: Generating training flag variable to identify and assign training and test observations
df$train_flag <- ifelse(df$random_number>= 0.5, 1, 0)
view(df)

#Report number of observations in training and test samples
sum(df$train_flag)
```

```
## [1] 375
```

```
sum(1-df$train_flag)
```

```
## [1] 366
```

The training sample, or control group, contains 375 observations. The test sample, or treatment group, contains 366 observations.

```
#Question 3: Creating training and test data frames
test <- df |>
       filter(train_flag == 0)
view(test)

train <- df |>
         filter(train_flag == 1)
view(train)
```

```
#Question 4a: Creating multivariate regression for absolute mobility at p = 25

#Four predictor variables chosen:
#share_hisp2010 #Hispanic share of population in 2010
#emp2000 #employment rate in 2000
#frac_coll_plus2000 #fraction of college degree attainment or more in 2000
#job_growth_1990_2010 #job growth rate between 1990-2010
```

```
mobilityreg <- lm(kfr_pooled_pooled_p25 ~ share_hisp2010 + emp2000 + frac_coll_plus2000 + job_growth_19
summary(mobilityreg)
```

```
##
## Call:
## lm(formula = kfr_pooled_pooled_p25 ~ share_hisp2010 + emp2000 +
##     frac_coll_plus2000 + job_growth_1990_2010, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.8754  -3.8242  -0.7427   3.3121  22.5468
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)           21.19064    2.85997   7.409 8.69e-13 ***
## share_hisp2010         4.17192    1.96278   2.126  0.03421 *
## emp2000               43.13663    5.65983   7.622 2.13e-13 ***
## frac_coll_plus2000   -17.29014    5.73252  -3.016  0.00274 **
## job_growth_1990_2010  -0.03223    0.01664  -1.937  0.05345 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.504 on 370 degrees of freedom
## Multiple R-squared:  0.1427, Adjusted R-squared:  0.1335
## F-statistic:  15.4 on 4 and 370 DF,  p-value: 1.161e-11
```

```
#4b: Using training model to predict absolute mobility for Milwaukee, WI
#Display data for Milwaukee, WI
summary(subset(df, cz == 24100))
```

```
##        cz           czname          kfr_pooled_pooled_p25 bowl_per_capita
##  Min.   :24100   Length:1           Min.   :38.89         Min.   :5.721
##  1st Qu.:24100   Class :character   1st Qu.:38.89         1st Qu.:5.721
##  Median :24100   Mode  :character   Median :38.89         Median :5.721
##  Mean   :24100                      Mean   :38.89         Mean   :5.721
##  3rd Qu.:24100                      3rd Qu.:38.89         3rd Qu.:5.721
##  Max.   :24100                      Max.   :38.89         Max.   :5.721
##  singleparent_share1990 singleparent_share2000 singleparent_share2010
##  Min.   :0.2262         Min.   :0.2965         Min.   :0.3404
##  1st Qu.:0.2262         1st Qu.:0.2965         1st Qu.:0.3404
##  Median :0.2262         Median :0.2965         Median :0.3404
##  Mean   :0.2262         Mean   :0.2965         Mean   :0.3404
##  3rd Qu.:0.2262         3rd Qu.:0.2965         3rd Qu.:0.3404
##  Max.   :0.2262         Max.   :0.2965         Max.   :0.3404
##  hhinc_mean2000  mean_commutetime2000 frac_coll_plus2000 frac_coll_plus2010
##  Min.   :84869   Min.   :23.58        Min.   :0.2515     Min.   :0.2893
##  1st Qu.:84869   1st Qu.:23.58        1st Qu.:0.2515     1st Qu.:0.2893
##  Median :84869   Median :23.58        Median :0.2515     Median :0.2893
##  Mean   :84869   Mean   :23.58        Mean   :0.2515     Mean   :0.2893
##  3rd Qu.:84869   3rd Qu.:23.58        3rd Qu.:0.2515     3rd Qu.:0.2893
##  Max.   :84869   Max.   :23.58        Max.   :0.2515     Max.   :0.2893
##  foreign_share2010 med_hhinc1990   med_hhinc2016   poor_share2010
##  Min.   :0.06456   Min.   :35061   Min.   :60341   Min.   :0.1312
##  1st Qu.:0.06456   1st Qu.:35061   1st Qu.:60341   1st Qu.:0.1312
```

```
##  Median :0.06456   Median :35061   Median :60341   Median :0.1312
##  Mean   :0.06456   Mean   :35061   Mean   :60341   Mean   :0.1312
##  3rd Qu.:0.06456   3rd Qu.:35061   3rd Qu.:60341   3rd Qu.:0.1312
##  Max.   :0.06456   Max.   :35061   Max.   :60341   Max.   :0.1312
##  poor_share2000    poor_share1990    share_white2010  share_black2010
##  Min.   :0.09775   Min.   :0.09566   Min.   :0.7119   Min.   :0.1586
##  1st Qu.:0.09775   1st Qu.:0.09566   1st Qu.:0.7119   1st Qu.:0.1586
##  Median :0.09775   Median :0.09566   Median :0.7119   Median :0.1586
##  Mean   :0.09775   Mean   :0.09566   Mean   :0.7119   Mean   :0.1586
##  3rd Qu.:0.09775   3rd Qu.:0.09566   3rd Qu.:0.7119   3rd Qu.:0.1586
##  Max.   :0.09775   Max.   :0.09566   Max.   :0.7119   Max.   :0.1586
##  share_hisp2010    share_asian2010   share_black2000  share_white2000
##  Min.   :0.09059   Min.   :0.01592   Min.   :0.1344   Min.   :0.7757
##  1st Qu.:0.09059   1st Qu.:0.01592   1st Qu.:0.1344   1st Qu.:0.7757
##  Median :0.09059   Median :0.01592   Median :0.1344   Median :0.7757
##  Mean   :0.09059   Mean   :0.01592   Mean   :0.1344   Mean   :0.7757
##  3rd Qu.:0.09059   3rd Qu.:0.01592   3rd Qu.:0.1344   3rd Qu.:0.7757
##  Max.   :0.09059   Max.   :0.01592   Max.   :0.1344   Max.   :0.7757
##  share_hisp2000    share_asian2000   gsmn_math_g3_2013 rent_twobed2015
##  Min.   :0.05953   Min.   :0.01126   Min.   :3.375     Min.   :887
##  1st Qu.:0.05953   1st Qu.:0.01126   1st Qu.:3.375     1st Qu.:887
##  Median :0.05953   Median :0.01126   Median :3.375     Median :887
##  Mean   :0.05953   Mean   :0.01126   Mean   :3.375     Mean   :887
##  3rd Qu.:0.05953   3rd Qu.:0.01126   3rd Qu.:3.375     3rd Qu.:887
##  Max.   :0.05953   Max.   :0.01126   Max.   :3.375     Max.   :887
##  traveltime15_2010    emp2000       mail_return_rate2010 popdensity2010
##  Min.   :0.2922    Min.   :0.6491   Min.   :81.96        Min.   :598.8
##  1st Qu.:0.2922    1st Qu.:0.6491   1st Qu.:81.96        1st Qu.:598.8
##  Median :0.2922    Median :0.6491   Median :81.96        Median :598.8
##  Mean   :0.2922    Mean   :0.6491   Mean   :81.96        Mean   :598.8
##  3rd Qu.:0.2922    3rd Qu.:0.6491   3rd Qu.:81.96        3rd Qu.:598.8
##  Max.   :0.2922    Max.   :0.6491   Max.   :81.96        Max.   :598.8
##  popdensity2000  job_growth_1990_2010 ann_avg_job_growth_2004_2013
##  Min.   :575.4   Min.   :5.551        Min.   :0.002124
##  1st Qu.:575.4   1st Qu.:5.551        1st Qu.:0.002124
##  Median :575.4   Median :5.551        Median :0.002124
##  Mean   :575.4   Mean   :5.551        Mean   :0.002124
##  3rd Qu.:575.4   3rd Qu.:5.551        3rd Qu.:0.002124
##  Max.   :575.4   Max.   :5.551        Max.   :0.002124
##  job_density_2013 random_number      train_flag
##  Min.   :294.1    Min.   :0.5273   Min.   :1
##  1st Qu.:294.1    1st Qu.:0.5273   1st Qu.:1
##  Median :294.1    Median :0.5273   Median :1
##  Mean   :294.1    Mean   :0.5273   Mean   :1
##  3rd Qu.:294.1    3rd Qu.:0.5273   3rd Qu.:1
##  Max.   :294.1    Max.   :0.5273   Max.   :1
```

```
#Generate absolute mobility and training models variables for Milwaukee
df_milwaukee <- df |>
                filter(cz == 24100) |>
                select(kfr_pooled_pooled_p25, share_hisp2010, emp2000, frac_coll_plus2000, job_growth_19
summary(df_milwaukee)
```

```
##  kfr_pooled_pooled_p25 share_hisp2010       emp2000       frac_coll_plus2000
##  Min.   :38.89         Min.   :0.09059   Min.   :0.6491   Min.   :0.2515
```

```
##  1st Qu.:38.89         1st Qu.:0.09059   1st Qu.:0.6491   1st Qu.:0.2515
##  Median :38.89         Median :0.09059   Median :0.6491   Median :0.2515
##  Mean   :38.89         Mean   :0.09059   Mean   :0.6491   Mean   :0.2515
##  3rd Qu.:38.89         3rd Qu.:0.09059   3rd Qu.:0.6491   3rd Qu.:0.2515
##  Max.   :38.89         Max.   :0.09059   Max.   :0.6491   Max.   :0.2515
##  job_growth_1990_2010
##  Min.   :5.551
##  1st Qu.:5.551
##  Median :5.551
##  Mean   :5.551
##  3rd Qu.:5.551
##  Max.   :5.551
```

```
#Use training model to predict absolute mobility
21.19064 + (4.17192*0.09059) +(43.13663*0.6491) - (17.29014 * 0.2515) -(0.03223 * 5.551)
```

```
## [1] 45.04118
```

```
45.04118-38.89
```

```
## [1] 6.15118
```

The actual absolute mobility for Milwaukee is 38.89. The training model generates an predicted absolute mobility of 45.04118. The prediction error is 6.15118.

```
#4C: Generate predictions for all observations in the training data
y_train_predictions_ols <- predict(mobilityreg, newdata=train)

#4D: Generate predictions for all observations in the test data
y_test_predictions_ols <- predict(mobilityreg, newdata=test)

#Generate squared prediction errors
OLS_performance_testset <- (test$kfr_pooled_pooled_p25 - y_test_predictions_ols)^2
OLS_performance_trainset <- (train$kfr_pooled_pooled_p25 - y_train_predictions_ols)^2

#Report the root mean squared prediction error
rmspe_test_ols <- sqrt(mean(OLS_performance_testset, na.rm=TRUE))
rmspe_train_ols <- sqrt(mean(OLS_performance_trainset, na.rm=TRUE))

rmspe_test_ols
```

```
## [1] 5.653217
```

```
rmspe_train_ols
```

```
## [1] 5.466711
```

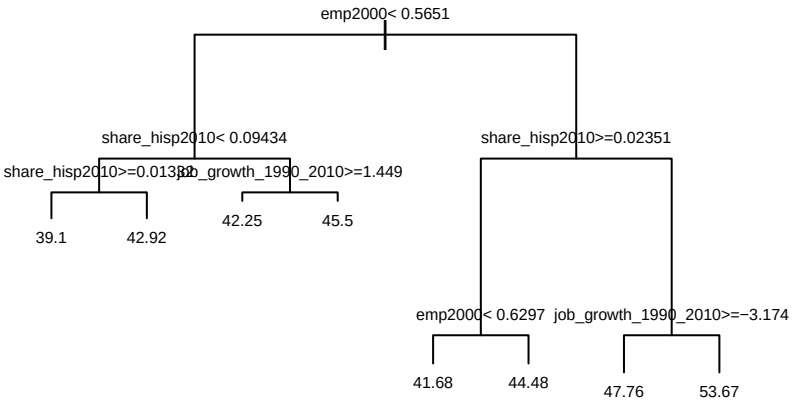
The prediction error in the test data is 5.65, while the prediction error in the training data is 5.467. The test data error is higher.

```
#Q5A: Estimate decision tree

mobilitytree <- rpart(kfr_pooled_pooled_p25 ~ share_hisp2010 + emp2000 + frac_coll_plus2000 + job_growth
                      data=train,
                      maxdepth = 3,
                      cp=0)


#Q5B: Visualize decision tree
```

```
#Visualize the fitted decision tree
plot(mobilitytree, margin = 0.2) # plot tree
text(mobilitytree, cex = 0.5) # add labels to tree
```



Using the decision tree, we can predict the rate of absolute mobility for Milwaukee. Milwaukee has a 2000 employment rate higher than 0.56, so we move to the right branch of the tree. The 2010 share of Hispanic population is greater than 0.02, so we move to the left-side sub-branch. Finally, 2000 employment is greater than 0.62, so the decision tree predicts that absolute mobility in Milwaukee is 44.48. The prediction error for Milwaukee is 5.59.

```
#Q5C-F: Calculating RMSPE in decision tree training vs. test data

#Calculate predictions for all rows in test and training samples
y_test_predictions_tree <- predict(mobilitytree, newdata=test)
y_train_predictions_tree <- predict(mobilitytree, newdata=train)

#Generate squared prediction errors
tree_performance_testset <- (test$kfr_pooled_pooled_p25 - y_test_predictions_tree)^2
tree_performance_trainset <- (train$kfr_pooled_pooled_p25 - y_train_predictions_tree)^2

#Report the root mean squared prediction error
rmspe_test_tree <- sqrt(mean(tree_performance_testset, na.rm=TRUE))
rmspe_train_tree <- sqrt(mean(tree_performance_trainset, na.rm=TRUE))

#Report the root mean squared prediction error
rmspe_test_tree
```

```
## [1] 5.967121
```

```
rmspe_train_tree
```

```
## [1] 4.804677
```

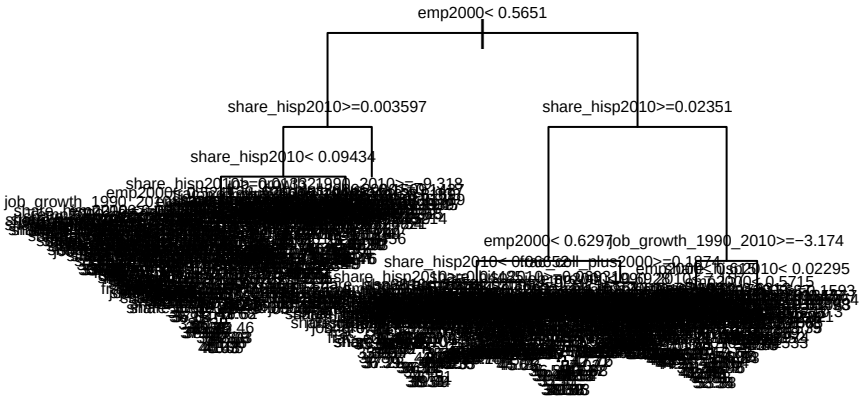
Again, prediction error in the test data is higher, at 5.967, compared to prediction error in the training data, which is 4.80.

```
#Q6: Illustrating overfitting problem in decision trees

big_tree <-rpart(kfr_pooled_pooled_p25 ~ share_hisp2010 + emp2000 + frac_coll_plus2000 + job_growth_199
             data=train,
             maxdepth = 30,
             cp=0,
             minsplit = 1,
```

```
             minbucket = 1)

#Visualize the fitted decision tree
plot(big_tree, margin = 0.2) # plot tree
text(big_tree, cex = 0.5) # add labels to tree
```




```
#Calculate predictions for all rows in test and training samples
y_test_predictions_big_tree <- predict(big_tree, newdata=test)
y_train_predictions_big_tree <- predict(big_tree, newdata=train)

#Generate squared prediction errors
big_tree_performance_testset <- (test$kfr_pooled_pooled_p25 - y_test_predictions_big_tree)^2
big_tree_performance_trainset <- (train$kfr_pooled_pooled_p25 - y_train_predictions_big_tree)^2

#Report the root mean squared prediction error
rmspe_test_big_tree <- sqrt(mean(big_tree_performance_testset, na.rm=TRUE))
rmspe_train_big_tree <- sqrt(mean(big_tree_performance_trainset, na.rm=TRUE))

#Report the root mean squared prediction error
rmspe_test_big_tree
```

```
## [1] 7.000565
```

```
rmspe_train_big_tree
```

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

Question 7 On the training sample, the large decision tree performs best, with RMSPE of 0, which makes sense, because we coded the tree to be so big that each observation is it's own leaf. On the test sample, however, the big tree performs the worst. The best performing test sample is the linear regression.