lab6_2024

March 27, 2024

1 Lab 6: Predicting Social Mobility using Trees and Regression

1.1 Methods/concepts: decision trees, prediction, in-sample vs. out of sample

Name: Shreya Chaturvedi

Email: shreyachaturvedi@hks.harvard.edu

HUID: 31575036

Lab: Thursday 3pm at HKS

Date: March 28, 2024

LAB DESCRIPTION

This is the first of three labs on prediction policy questions. In this lab, you will predict upward mobility using *decision trees* and *multivariable regression*. The measure of upward mobility that we will focus on is **Statistic 1: Absolute Mobility at the 25th Percentile** in each commuting zone (**kfr_pooled_pooled_p25**). For more details on the variables included in these data, see Table 1.

The focus of this lab will be on the *concepts* and not the coding. In the R labs, we will start from starter scripts that can either be run on your computer or the Ec 50 Jupyter Hub.

1.2 QUESTIONS

1. Why do we split our data into "test" and "training" datasets for prediction applications?

[1]:
```
# QUESTION 1 Code
```

Question 1 Answer

We split our data into "test" and "training" datasets for prediction applications primarily to evaluate how well our model can generalize to new, unseen data. The training dataset is used to teach the model the underlying patterns and relationships within the data, while the test dataset, which is held out and kept separate from the training process, allows us to assess the model's performance on examples it has never encountered before. This separation is crucial because if we were to evaluate the model solely on the data it was trained on, we would have no way of knowing whether it has truly learned generalizable concepts or if it has simply memorized the specific examples in the training set. By measuring performance on the test set, we can get an honest estimate of how well the model will extend to real-world data beyond just the finite training examples.

2. Now turn to the .R starter script and the mobility.dta data set. Just like in Lab 1 where we randomly assigned observations to treatment and control groups, in this week's lab we will randomly assign half of the data into a "test" and half into a "training" subsamples as follows:
   1. First, set the "seed" using your Harvard University ID number (set.seed(12345) in R), which will make your simulation reproducible, but different from your classmates' simulations. Then assign to each observation a random number drawn uniformly between 0 and 1.
   2. Generate a new variable *train_flag* that equals 1 ("training sample") if the number generated in part a is greater than or equal to 0.5, and otherwise equals 0 when the number is less than 0.5 ("test sample"). How many observations are in the training sample? How many are in your test sample?

[2]:
```
#clear the workspace
rm(list=ls()) # removes all objects from the environment

#Install and load haven package
if (!require(haven)) install.packages("haven"); library(haven)
if (!require(rpart)) install.packages("rpart"); library(rpart)

#Load stata data set
download.file("https://raw.githubusercontent.com/ekassos/ec50_s24/main/mobility.
↪dta", "mobility.dta", mode = "wb")
mobility <- read_dta("mobility.dta")

# QUESTION 2 Code
# Set a seed for reproducibility based on my Harvard ID
set.seed(31575036)

# Assign a random number between 0 and 1 to each row in the 'mobility' data␣
↪frame
mobility$random_num <- runif(nrow(mobility))

# Create a binary flag to identify rows for the training set (50% of the data)
mobility$training_flag <- ifelse(mobility$random_num < 0.5, 1, 0)

# Calculate and report the number of entries in each subset
training_count <- sum(mobility$training_flag)
test_count <- nrow(mobility) - training_count

training_count
test_count
nrow(mobility)
```

```
Loading required package: haven

Loading required package: rpart
```

352

389

741

Question 2 Answer

There are 352 and 389 observations in my training and test set respectively. This is inline with roughly 50% of the values in each (which is approximately 370) with some variation due to random sampling.

3. Then subset the mobility.dta data set to create two new data frames:
   1. train is the training dataset containing observations where `train_flag=1`
   2. test is the test dataset containing observations where `train_flag=0`

[3]:
```
# QUESTION 3 Code

## Create some data frames that just contain the training and test data
train <- subset(mobility, training_flag == 1)
test <- subset(mobility, training_flag == 0)
```

Question 3 Answer

Code as shown above. All the observations with the training flag set to 1 are in the train dataset and other way round for the test dataset.

4. Now we will use linear regression to predict upward mobility.
   1. Start by estimating a regression of **kfr_pooled_pooled_p25** on at least three predictor variables in the training data. Please choose at least three, but you can choose more than three if you want. The three that you choose should not include my two predictors: 'bowl_per_capita' and 'singleparent_share1990'. Pick your own!
   2. Use the estimated coefficients from the regression to predict **kfr_pooled_pooled_p25** for Milwaukee, WI, which has cz == 24100. For example, in a regression using bowling alleys per 100,000 residents and single parent families as predictors, we could simply use the estimated regression coefficients and the fact that Milwaukee, WI has 5.72 bowling alleys per 100,000 residents and 22.6% single parent families in 1990 to obtain the predicted value. What is the prediction error for Milwaukee?
   3. Obtain predictions for the **training data** and create a new variable called y_train_predictions_ols.
   4. Obtain predictions for the **test data** and create a new variable called y_test_predictions_ols.
   5. Calculate the **root mean squared prediction error** in the training data and the test data
   6. Compare the prediction error in the test vs. the training data. Which is higher?

[4]:
```
# QUESTION 4 Code

#Question 4 Code
#Modified linear regression using variables job growth rate, employment rate,
↪and share of poor people as independent variables
```

```
mobilityreg <- lm(kfr_pooled_pooled_p25 ~ job_growth_1990_2010 + emp2000 +
  poor_share2000, data=train)
summary(mobilityreg)

### Display data for Milwaukee, WI
summary(subset(mobility, cz == 24100))

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

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

#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))

print(paste("The Root Mean Squared Percentage Error (RMSPE) for the test
  dataset is:", rmspe_test_ols))
print(paste("The Root Mean Squared Percentage Error (RMSPE) for the training
  dataset is:", rmspe_train_ols))
```

```
Call:
lm(formula = kfr_pooled_pooled_p25 ~ job_growth_1990_2010 + emp2000 +
    poor_share2000, data = train)

Residuals:
     Min       1Q   Median       3Q      Max
-15.0651  -3.2940  -0.1796   2.9274  18.1049

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)          27.77955    4.70758   5.901 8.57e-09 ***
job_growth_1990_2010 -0.04892    0.01389  -3.522 0.000486 ***
emp2000              27.72915    6.69798   4.140 4.36e-05 ***
poor_share2000       -4.57024    7.46011  -0.613 0.540526
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.16 on 348 degrees of freedom
```

```
Multiple R-squared:  0.123,	Adjusted R-squared:  0.1154 
F-statistic: 16.27 on 3 and 348 DF,  p-value: 6.465e-10

       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_num     training_flag
 Min.   :294.1    Min.   :0.1871   Min.   :1
 1st Qu.:294.1    1st Qu.:0.1871   1st Qu.:1
 Median :294.1    Median :0.1871   Median :1
 Mean   :294.1    Mean   :0.1871   Mean   :1
 3rd Qu.:294.1    3rd Qu.:0.1871   3rd Qu.:1
 Max.   :294.1    Max.   :0.1871   Max.   :1

[1] "The Root Mean Squared Percentage Error (RMSPE) for the test dataset is:
6.06842973412517"
[1] "The Root Mean Squared Percentage Error (RMSPE) for the training dataset is:
5.13056256355922"
```

Question 4 Answer

A. I have chosen the variables job_growth_1990_2010, emp2000, and poor_share2000 as the dependent variables in the regression above.

B. To predict using the estimated coefficient, we need to compute 27.77955 -0.04892 x 5.551 + 27.72915 x 0.6491 - 4.57024 x 0.09775 which is approximately equal to 45.06025. The actual value of the independent variable is 38.89, therefore the prediction error is approximately 6.17 (45.06-38.89)

C, D, E. Codes shown above calculate the predictions and the prediction errors for training and testing data.

F. As expected, the training data has a smaller RMSE value than the testing data (5.13 vs 6.07).

5. Next we will use a decision tree to predict upward mobility.
    1. Estimate a decision tree to predict **kfr_pooled_pooled_p25** using the same predictor variables in the training data that you used for the regression.
    2. Visualize your decision tree in “tree form” and include your image in your solutions. Use

the graphical representation of the decision tree to predict **kfr_pooled_pooled_p25** for Milwaukee, WI. What is the prediction error for Milwaukee?

3. Obtain predictions for the **training data** and create a new variable called y_train_predictions_tree.
4. Obtain predictions for the entire **test data** and create a new variable called y_test_predictions_tree.
5. Calculate the **root mean squared prediction error** in the training data and the test data
6. Compare the prediction error in the test vs. the training data. Which is higher?

[5]:
```
# QUESTION 5 Code

#Question 5.        Prediction using decision tree

#### Trees example: modify this code to complete the coding exercise
## Method is rpart()
## Depth 3
mobilitytree <- rpart(kfr_pooled_pooled_p25 ~ job_growth_1990_2010 + emp2000 +␣
 ↪poor_share2000,
                      data=train,
                      maxdepth = 3,
                      cp=0)

#Options for rpart
#cp = complexity parameter, which controls the complexity of the tree. 0 is␣
 ↪most complex.
#If cp>0 the tree will only grown if the increase in tree size improve the␣
 ↪performance of the tree by at least cp.
#maxdepth = maximum depth of any node of the final tree, with the root node␣
 ↪counted as depth 0
#Other tuning parameters that can be changed
#help("rpart.control")

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

#Save figure
dev.copy(png,'figure1.png')
dev.off()

#Apply tree to predict Milwaukee, WI
summary(subset(mobility, cz == 24100))

#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)
```

```r
#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
print(paste("The Root Mean Squared Percentage Error (RMSPE) for the test␣
 ↪dataset is:", rmspe_test_tree))
print(paste("The Root Mean Squared Percentage Error (RMSPE) for the training␣
 ↪dataset is:", rmspe_train_tree))
```

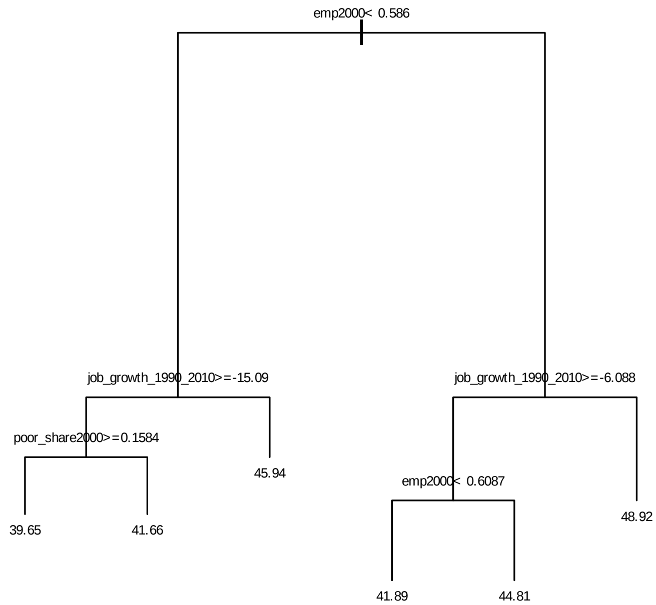
png: 3

png: 2

```
       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_num     training_flag
 Min.   :294.1    Min.   :0.1871   Min.   :1    
 1st Qu.:294.1    1st Qu.:0.1871   1st Qu.:1    
 Median :294.1    Median :0.1871   Median :1    
 Mean   :294.1    Mean   :0.1871   Mean   :1    
 3rd Qu.:294.1    3rd Qu.:0.1871   3rd Qu.:1    
 Max.   :294.1    Max.   :0.1871   Max.   :1    
```

```
[1] "The Root Mean Squared Percentage Error (RMSPE) for the test dataset is:
6.12432872352905"
[1] "The Root Mean Squared Percentage Error (RMSPE) for the training dataset is:
4.89880575737342"
```



Question 5 Answer

A. Code for decision tree using same vairables as OLS is given above and the resultant tree created is shown. B. Using the decision tree above, we traverse to find that the predicted value for Milwaukee, WI is 48.92 whereas the actual value is 38.89. The prediction error is therefore approximately 10.03 (48.92-38.89). C, D, E. Codes given above calculate the predictions and the prediction errors for training and testing data. F. As expected, the training data has a smaller RMSE value than the testing data (4.90 vs 6.12). The training data RMSE is slightly less than in the OLS case whereas the testing data RMSE is approximately the same.

6. To conclude this week's lab, we will illustrate the overfit problem. The key issue in using a decision tree to make predictions is choosing how big of a tree you want to grow. How many splits in the tree? Or in other words, the depth of tree.

   Decision trees have a tendency to overfit the training data. By growing a bigger and bigger tree, we can drive the in-sample prediction error down to zero. The tree that minimizes the in-sample error would be a tree where each observation is in its own leaf. But that large decision tree is not likely to do well when trying to make an out of sample prediction. As with regression, it's possible to fit our existing data perfectly but have terrible predictions for new data.

To show this, fit a tree in R using rpart() with maximum depth maxdepth = 30, complexity parameter cp = 0, minimum number of observations in each leaf minbucket = 1, and the minimum number of observations in a leaf for a split to be attempted minsplit = 1. Calculate the **root mean squared prediction error** in the training data and the test data.

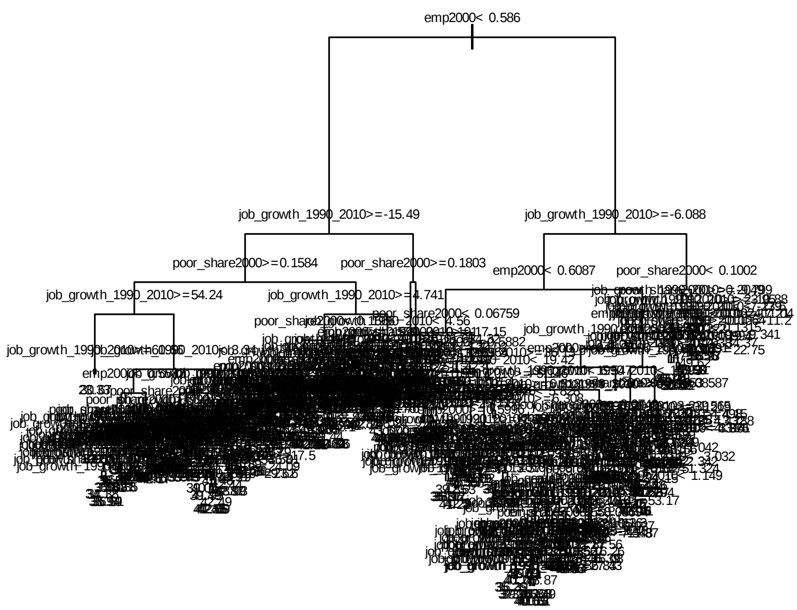
[6]:
```
# QUESTION 6 Code

#Estimate large tree
big_tree <-rpart(kfr_pooled_pooled_p25 ~ job_growth_1990_2010 + emp2000 +␣
 ↪poor_share2000,
             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

#Save figure
dev.copy(png,'figure2.png')
dev.off()

#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
print(paste("The Root Mean Squared Percentage Error (RMSPE) for the test␣
 ↪dataset is:", rmspe_test_big_tree))
```

```
print(paste("The Root Mean Squared Percentage Error (RMSPE) for the training␣
↪dataset is:", rmspe_train_big_tree))
```

png: 3

png: 2

```
[1] "The Root Mean Squared Percentage Error (RMSPE) for the test dataset is:
7.77424406754692"
[1] "The Root Mean Squared Percentage Error (RMSPE) for the training dataset is:
0"
```



Question 6 Answer

The modified code above runs a big/overfit tree on the same three vairables. As expected, the training RMSE falls to 0 since the tree is overfit to this data and predicts accurately, whereas the testing error increases to 7.72 which is higher than both OLS and the smaller decision tree.

7. Which of the three models – the linear regression, the small decision tree, or the big decision tree – performs best on the training sample? What about the test sample?

[7]:
```
# QUESTION 7 Code

#Compare RMSPE for the three models in-sample
#Compare RMSPE for the three models in the pseudo out-of-sample
# Create a data frame to hold the RMSPE values
rmspe_table <- data.frame(
  Model = c("Big Tree", "Tree", "OLS"),
  RMSPE_Test_PsuedoOOS = c(rmspe_test_big_tree, rmspe_test_tree,␣
↪rmspe_test_ols),
  RMSPE_Train_IS = c(rmspe_train_big_tree, rmspe_train_tree, rmspe_train_ols)
)

# Print the table
print(rmspe_table)
```

```
     Model RMSPE_Test_PsuedoOOS RMSPE_Train_IS
1 Big Tree             7.774244       0.000000
2     Tree             6.124329       4.898806
3      OLS             6.068430       5.130563
```

Question 7 Answer

As shown above, the table compares the RMSPE for training data (in sample) and testing data (psuedo out of sample) for the tree models - overfit big tree, simple tree, and OLS.

8. Create an annotated/commented do-file, .ipynb Jupyter Notebook, or .R file that can replicate all your analyses above. This will be the final code that you submit on Gradescope. The motivation for using do-files and .R files is described on page 4, which has been adapted from training materials used by Innovations for Poverty Action (IPA) and the Abdul Latif Jameel Poverty Action Lab (J-PAL).

Final Submission Checklist for Lab 6

If you're working with R

If you're working with Stata

Lab 6 Write-Up:

PDF of your answers. For graphs, you must save them as images (e.g., .png files) and insert them into the document.

Lab 6 Code:

.R script file, well-annotated replicating all your analyses;OR

.ipynb file

Lab 6 Write-Up:

PDF of your answers. For graphs, you must save them as images (e.g., .png files) and insert them into the document.

Lab 6 Code:

do-file, well-annotated replicating all your analyses;AND

log-file, not a .smcl file, with the log showing the output generated by your final do-file.

If you're working with an .ipynb notebook

It is likely that your .ipynb file will be greater than 1 MB in size. Therefore, for this assignment please submit both your *well-annotated* **.ipynb file** and **a .PDF version of this file**. The notebook should replicate all your analyses for Lab 5 (with enough comments that a principal investigator on a research project would be able to follow and understand what each step of the code is doing).

1.3 How to submit your assignment

1.4 What files to submit

If you're using Python Notebook to write your R code, and a document editor to write your answers
If you're using a Python Notebook to write your R code AND to write your answers

1.5 WHAT ARE DO-FILES AND .R FILES AND WHY DO WE NEED ONE?

Let's imagine the following situation - you just found out you have to present your results to a partner– all the averages you produced and comparisons you made. Suppose you also found out that the data you had used to produce all these results was not completely clean, and have only just fixed it. You now have incorrect numbers and need to re-do everything.

How would you go about it? Would you reproduce everything you did for Lab 1 from scratch? Can you do it? How long would it take you to do? Just re-typing all those commands into Stata or R in order and checking them would take an hour.

An important feature of any good research project is that the results should be reproducible. For Stata and R the easiest way to do this is to create a text file that lists all your commands in order, so anyone can re-run all your Stata or R work on a project anytime. Such text files that are produced within Stata or linked to Stata are called do-files, because they have an extension .do (like intro_exercise.do). Similarly, in R, these files are called .R files because they have an extension of .R. These files feed commands directly into Stata or R without you having to type or copy them into the command window.

An added bonus is that having do-files and .R files makes it very easy to fix your typos, re-order commands, and create more complicated chains of commands that wouldn't work otherwise. You can now quickly reproduce your work, correct it, adjust it, and build on it.

Finally, do-files and .R files make it possible for multiple people to work on a project, which is necessary for collaborating with others or when you hand off a project to someone else.

1.6 DATA DESCRIPTION, FILE: mobility.dta

The data consist of $N = 741$ Commuting Zones. Commuting zones are geographical aggregations of counties that are similar to metro areas but cover the entire U.S., including rural areas. Commuting zones are meant to consist of local labor markets where people both live and work. For more details on the construction of the variables included in this data set, please see Chetty, Raj, John Friedman, Nathaniel Hendren, Maggie R. Jones, and Sonya R. Porter. 2018. "The Opportunity Atlas: Mapping the Childhood Roots of Social Mobility." NBER Working Paper No. 25147.

TABLE 1

Variable Definitions

Variable

Description

Obs.

Mean

St. Dev.

Min

Max

(1)

(2)

(3)

(4)

(5)

(6)

(7)

1

cz

Five-digit 1990 commuter zone code

741

n/a

n/a

n/a

n/a

2

cz_name

String variable consisting of the name of the commuting zone.

741

n/a

n/a

n/a

n/a

3

kfr_pooled_pooled_p25

Absolute Mobility at the 25th Percentile

741

42.99

5.994

23.33

66.63

4

bowl_per_capita

Bowling Alleys per 100,000 residents

741

3.928

5.661

0

70.50

5

singleparent_share1990

Share of Single-Headed Households with Children 1990

741

0.197

0.0503

0.0433

0.441

6

singleparent_share2000

Share of Single-Headed Households with Children 2000

741

0.268

0.0563

0.105

0.547

7

singleparent_share2010

Share of Single-Headed Households with Children 2006-2010 ACS

741

0.315

0.0687

0.109

0.573

8

hhinc_mean2000

Mean Household Income 2000

741

65,137

12,755

38,817

122,288

9

mean_commutetime2000

Average Commute Time of Working Adults in 2000

741

21.97

4.548

7.383

40.24

10

frac_coll_plus2000

Fraction of Residents w/ a College Degree or More in 2000

741

0.181

0.0639

0.0488

0.481

11

frac_coll_plus2010

Fraction of Residents w/ a College Degree or More in 2006-2010 ACS

741

0.204

0.0695

0.0764

0.481

12

foreign_share2010

Share of Population Born Outside the U.S. in 2006-2010 ACS

741

0.0517

0.0622

0.000817

0.722

13

med_hhinc1990

Median Household Income in 1990

741

24,973

6,371

12,097

51,112

14

med_hhinc2016

Median Household Income in 2016

741

48,983

10,936

26,645

103,043

15

poor_share2010

Share Below Poverty Line 2006-2010 ACS

741

0.160

0.0541

0.0500

0.442

16

poor_share2000

Share Below Poverty Line 2000

741

0.145

0.0569

0.0540

0.460

17

poor_share1990

Share Below Poverty Line 1990

741

0.165

0.0703

0.0515

0.505

18

share_white2010

Share White 2010

741

0.758

0.197

0.0286

0.986

19

share_black2010

Share Black 2010

741

0.0846

0.123

0.00151

0.697

20

share_hisp2010

Share Hispanic 2010

741

0.0999

0.145

0.00249

0.957

21

share_asian2010

Share Asian 2010

741

0.0132

0.0328

0.000414

0.428

22

share_black2000

Share Black 2000

741

0.0773

0.118

0

0.646

23

share_white2000

Share White 2000

741

0.795

0.188

0.0365

0.991

24

share_hisp2000

Share Hispanic 2000

741

0.0755

0.133

0.00186

0.948

25

share_asian2000

Share Asian 2000

741

0.0107

0.0313

0.000244

0.455

26

gsmn_math_g3_2013

Average School District Level Standardized Test Scores in 3rd Grade in 2013

741

3.190

0.652

-0.661

4.960

27

rent_twobed2015

Average Rent for Two-Bedroom Apartment in 2015

741

704.3

185.9

336.0

1,652

28

traveltime15_2010

Share of Working Adults w/ Commute Time of 15 Minutes Or Less in 2006-2010 ACS

741

0.450

0.143

0.152

0.991

29

emp2000

Employment Rate 2000

741

0.578

0.0639

0.323

0.756

30

mail_return_rate2010

Census Form Rate Return Rate 2010

741

79.82

5.205

47.80

88.98

31

popdensity2010

Population Density (per square mile) in 2010

741

109.4

284.0

0.106

5,636

32

popdensity2000

Population Density (per square mile) in 2000

741

101.3

271.0

0.0833

5,506

33

job_growth_1990_2010

Job Growth Rate 1990-2010

741

13.54

21.56

-36.52

148.2

34

ann_avg_job_growth

_2004_2013

Average Annual Job Growth Rate 2004-2013

741

-0.001

0.0134

-0.0827

0.107

35

job_density_2013

Job Density (in square miles) in 2013

741

50.28

133.6

0.0425

2,595