HKS SUP-135 Lab 7: Predicting Social Mobility using Cross Validation and Random Forests

Matt Khinda

4/10/2023

```r
# Set random seed
HUID <- 41531460
set.seed(HUID)

# Store predictor variables
pred_vars <- colnames(atlas_train[,grep("^[P_]", names(atlas_train))])

# Create a training set with only predictors and kfr_pooled_pooled_p25
training_set <- subset(atlas_train, training == 1, pred_vars)
training_set$kfr_pooled_pooled_p25 <- atlas_train[atlas_train$training==1,]$kfr_pooled_pooled_p25
```

Question 1: For Loops and Steady States

```r
# Initialize variables for regression
gens <- seq(1,7,1)
alpha <- 36.14
beta <- 0.26
parent_rank <- 36.17

for (j in gens){
  kid_rank = alpha + beta * parent_rank
  cat(paste0("In generation ", j, ", parent_rank = ",
            parent_rank, ", child_rank = ", kid_rank), sep = "
    ")
  parent_rank <- kid_rank
}
```

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

Using the rank-rank regression for economic outcomes of Hispanic children from Chetty et al. (2020), with the average starting parent rank of the 36.17th percentile, the stable state after 7 generations appears to be around the 48.83th percentile for an improvement of about 12 percentiles from the current average.

Question 2: The Purpose of Cross-Validation

Cross-validation helps address the issue of overfitting by dividing the training data randomly into folds, which are iteratively used to evaluate trees of varying depths that have been trained on the other folds. This creates psuedo out-of-sample tests that help determine the appropriate depth of a tree without the risk of overfititng the models to the noise in the full dataset. Practically speaking, the appropriate tree depth is based on the combined root mean squared prediction error for all of trees of that depth across all iterations of the folds.

Question 3: Five-Fold Cross Validation

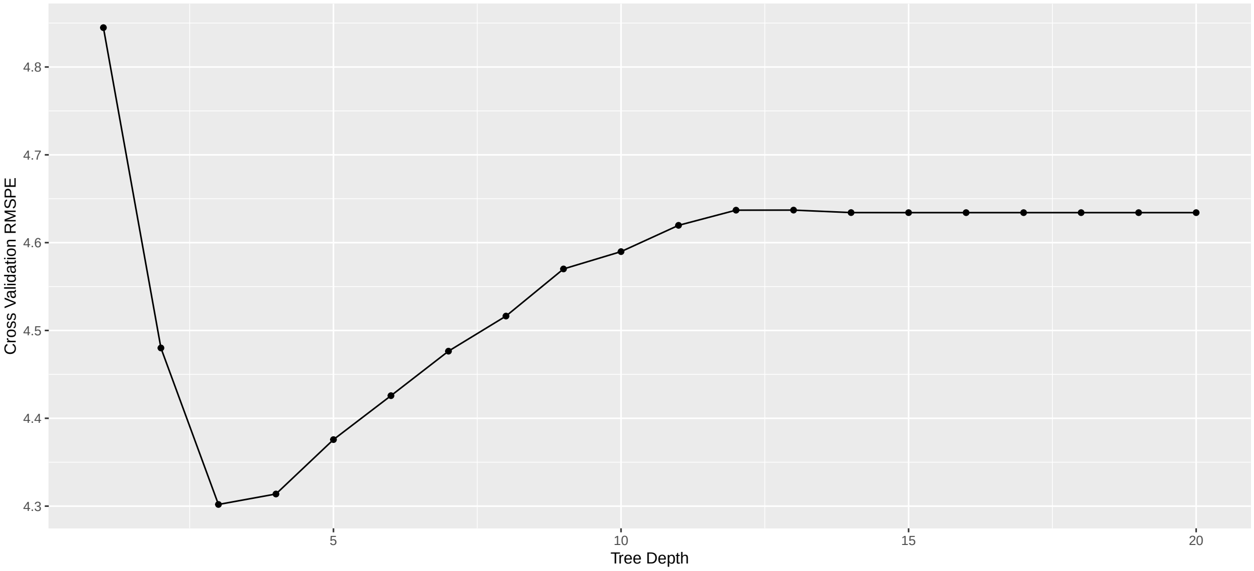
```r
n <- nrow(training_set)
K <- 5 # num folds
B <- seq(1,20,1) # Test tree depths from 1 to 20

# Copy training data for use inside the loop
cv <- training_set

# Randomly assign fold memberships to each row
cv$fold_id <- rep(1:K,each=ceiling(n/K))[sample(1:n)]

# Create an empty data frame to store future results
OOS <- data.frame(fold=rep(NA,K*length(B) ),
                  squarederror=rep(NA,K*length(B) ),
                  maxdepth=rep(NA,K*length(B) ))

# Start at row 0
row <- 0

for(j in B){
  for(k in 1:K){
    row <- row + 1 #increment row
    cv_train <- subset(cv, fold_id != k) #subset to include all but fold k
    cv_fold <- subset(cv, fold_id == k) #subset to include only fold k

    # Define tree parameters and grow based on training set
    cvtree <- rpart(kfr_pooled_pooled_p25 ~ P_12 + P_47,
                    data = cv_train,
                    maxdepth = c(j),
                    cp=0)

    # Apply trees to fold
    predfull <- predict(cvtree, newdata = cv_fold)

    # Calculate and store prediction errors, max depth, and id for fold k
    OOS$squarederror[row] <- sum((cv_fold$kfr_pooled_pooled_p25 - predfull)^2)
    OOS$maxdepth[row] <- j
    OOS$fold[row] <- k
    }
  }
```

```r
# Display summary results
summary(OOS)
```

```
##       fold    squarederror     maxdepth
##  Min.   :1   Min.   :4394   Min.   : 1.00
##  1st Qu.:2   1st Qu.:4919   1st Qu.: 5.75
##  Median :3   Median :5073   Median :10.50
##  Mean   :3   Mean   :5243   Mean   :10.50
##  3rd Qu.:4   3rd Qu.:5420   3rd Qu.:15.25
##  Max.   :5   Max.   :6965   Max.   :20.00
```

```r


# Calculate combined error across folds and put in dataframe
ssr <- tapply(OOS$squarederror, OOS$maxdepth, sum)
ssr <- as.data.frame(ssr)
ssr$maxdepth <- seq(1,20,1)

# Calculate RMSPE as new variable
ssr$rmse <- sqrt(ssr$ssr / nrow(training_set))

# Plot resulting CV RMSPE by tree depth
ggplot(ssr, aes(x=maxdepth,y=rmse)) +
  geom_point() +
  geom_line() +
  labs(y = "Cross Validation RMSPE",
       x = "Tree Depth")
```

3a: CV RMSE Plot

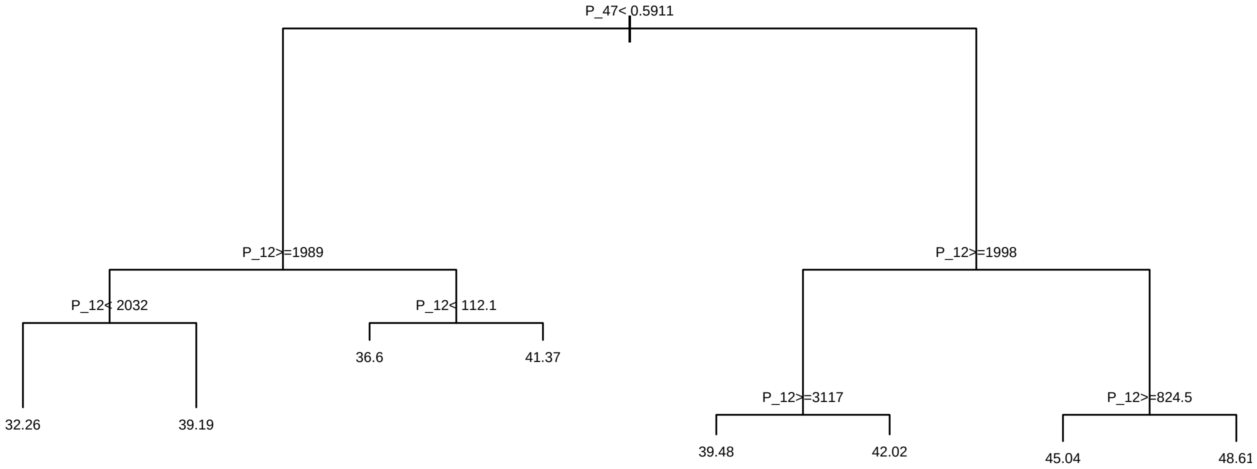


```
cv_optimal_depth <- ssr$maxdepth[ssr$rmse == min(ssr$rmse)]
```

3b: Optimal Tree Depth As shown in the chart above, based on the five-fold cross-validation, the optimal tree depth to reduce the root mean squared prediction error is 3.

```
# Create new tree at optimal depth using the full training set
tree <- rpart(kfr_pooled_pooled_p25 ~ P_12 + P_47,
                      data=training_set,
                      maxdepth = cv_optimal_depth,
                      cp=0)

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

3c: Estimate Tree Using the Full Dataset



```
# Predict values for each row using new tree
y_train_predictions_tree <- predict(tree, newdata=training_set)
```

3d: Predictions in the Training Sample

Question 4: Advantages of Random Forests

Random forests improve on the predictive power of decision tress by growing a much larger number of trees with a much greater depth than we could with a single tree. Each individual tree is trained on a different, randomly generated bootstrap sample that mimics the variability we would expect in a real-world out-of-sample test. By bagging, or taking the average prediction from the various decision trees, the risk of overfitting is eliminated while preserving the low bias from the larger tree depths. Equally, in wide datasets, to avoid over-correlation between random trees and to ensure the usefulness the average prediction, a technique called input randomization is used whereby the predictor variables in each tree are randomly selected among the full set.

Question 5: Random Forest with 1000 Trees (2 predictors)

```
# Generate forest of 1000 trees on two variables
small_forest <- randomForest(kfr_pooled_pooled_p25 ~ P_12 + P_47,
                              ntree=1000,
                              mtry=2,
                              data=training_set)

# Use small forest to generate predictions within training set
y_train_predictions_smallforest <- predict(small_forest,
                                            newdata=training_set,
                                            type="response")

small_forest
```

```
##
## Call:
##  randomForest(formula = kfr_pooled_pooled_p25 ~ P_12 + P_47, data = training_set,      ntree = 1000,
##                Type of random forest: regression
##                      Number of trees: 1000
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 19.83495
##                     % Var explained: 25.75
```

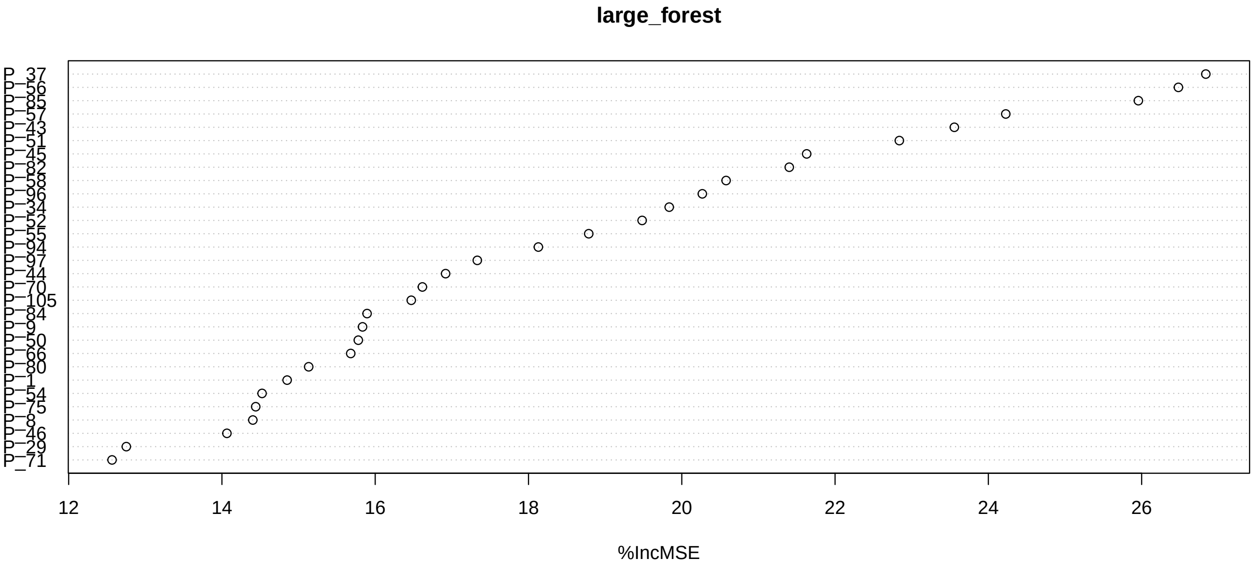
Question 6: Random Forest with 1000 Trees (All predictors)

```
# Generate forest of 1000 trees on all variables
large_forest <- randomForest(kfr_pooled_pooled_p25 ~ .,
                              ntree=1000,
                              mtry=40,
                              importance = T,
                              data=training_set)

# Use large forest to generate predictions within training set
y_train_predictions_largeforest <- predict(large_forest,
                                            newdata=training_set,
                                            type="response")

large_forest
```

```
##
## Call:
##  randomForest(formula = kfr_pooled_pooled_p25 ~ ., data = training_set,      ntree = 1000, mtry = 40
##                Type of random forest: regression
##                      Number of trees: 1000
## No. of variables tried at each split: 40
##
##           Mean of squared residuals: 4.726496
##                     % Var explained: 82.31
```

Question 7: Identifying Important Predictors in a Random Forest

```
varImpPlot(large_forest, type=1)
```



Based on the table and plot above, the three most important predictor variables are P_56 (Average number of mentally unhealthy days per month for people over 18 years old), P_85 (Fraction of the population that is Roman Catholic), and P_37 (Share of the population identifying as Black in 2000).

Question 8: RMSPE and Model Comparison

```
RMSPE <- matrix(0, 3, 1)
RMSPE[1] <- sqrt(mean((training_set$kfr_pooled_pooled_p25 -
                          y_train_predictions_tree)^2, na.rm=TRUE))
RMSPE[2] <- sqrt(mean((training_set$kfr_pooled_pooled_p25 -
                          y_train_predictions_smallforest)^2, na.rm=TRUE))
RMSPE[3] <- sqrt(mean((training_set$kfr_pooled_pooled_p25 -
                          y_train_predictions_largeforest)^2, na.rm=TRUE))

# Display summary table of results
knitr::kable(data.frame(RMSPE, method = c("Tree", "Small RF", "Large RF"))
             %>% select(method, RMSPE))
```

method	RMSPE
Tree	4.1625995
Small RF	2.1680658
Large RF	0.8652163

As shown in the summary table, the large random forest is the model that predicts economic outcomes with the highest accuracy based on the available variables. This is almost certainly because of its ability to incorporate all of the predictor variables to produce its forecasts rather than just the two that were used in the single tree and small random forest.

Question 9: Evaluating Models Against Lock Box Data

```r
atlas <- left_join(atlas_lockbox, atlas_train , by="geoid")
test <- subset(atlas, training==0)

#Get predictions for lock box data using 3 models
y_test_predictions_tree <- predict(tree, newdata=test)
y_test_predictions_smallforest <- predict(small_forest,
                                          newdata=test, type="response")
y_test_predictions_largeforest  <- predict(large_forest,
                                           newdata=test, type="response")

Test_RMSPE <- matrix(0, 3, 1)
Test_RMSPE[1] <- sqrt(mean((test$kfr_actual -
                              y_test_predictions_tree)^2, na.rm=TRUE))
Test_RMSPE[2] <- sqrt(mean((test$kfr_actual -
                              y_test_predictions_smallforest)^2, na.rm=TRUE))
Test_RMSPE[3] <- sqrt(mean((test$kfr_actual -
                              y_test_predictions_largeforest)^2, na.rm=TRUE))

# Display summary table of results
knitr::kable(data.frame(Test_RMSPE, method = c("Tree", "Small RF", "Large RF"))
             %>% select(method, Test_RMSPE))
```

method	Test_RMSPE
Tree	4.202734
Small RF	4.242246
Large RF	2.196632

When applied to the lock box data, the random forest remains the best predictive model of the three. Interestingly, in this case, the tree predicts slightly better than the small random forest unlike in the training dataset, though the difference is fairly small. Across all three models the error is significantly higher when applied to the test dataset as opposed to the training set, which is to be expected.