**STAT 387 – Project 1 Report**

**Predictive Modeling in Higher Education  
Winter 2024**

[**Haziel Garcia Sanchez**](mailto:haziel@pdx.edu) **&** [**Vix Talbot**](mailto:vix@pdx.edu)

**Summary**

For this project we were using the college dataset contained in the ISLR package. The dataset, collected in 1995, comprises measurements of 18 different characteristics from 777 colleges and universities across the United States. The dataset contained a variety of categorical and numerical characteristics which included tuition rates, number of Phds on the teaching faculty, and whether the college or university was public or private. In this dataset, Apps represented the number of applications that a college or university received. Using this information, our goal was to predict the number of applications that a college or university would receive by using a variety of ensemble methods.

A computer screen shot of a program

Description automatically generated

**Problem Statements**

Note that at this point we had split our data into a 50/50 training testing set, and were working with all of the features available to us.

**Fit a linear model using least squares and report the LOOCV estimate of the test error.**

First we fit a linear model using least squares and computed the LOOCV estimate of the test error. We got an MSE of 1527149.4258536 which translates to an RMSE of 1235.778874, the RMSE can be interpreted as our model's prediction is accurate within this number of college applications. In this case approximately 1,236 college applications. Our data set comes with a highly varied response variable. Our minimum value for total college applications received is 81, and the maximum value is 48,094 applications. We can tell that this simple linear prediction is useful for larger colleges but not necessarily for smaller ones.

**Fit a tree to the data. Summarize the results. Unless the number of terminal nodes is large, display the tree graphically and explicitly describe the regions corresponding to the terminal nodes that provide a partition of the predictor space (i.e., provide expressions for the regions R1; ...; RJ). Report its MSE.**

We used a decision tree approach and found that our tree had 8 terminal nodes. Decision trees work by considering our dataset and splitting it into two separate sets based on a certain threshold in each predictor. Then it measures the node purity of each split data set and selects the splits that yield that highest node purity.

A diagram of a tree

Description automatically generated

4 of these nodes were determined by Accept and the other 4 were determined by Top10perc, again confirming that these were the most significant predictors. We then created a table to explicitly explain the region space that corresponds to each leaf of the original decision tree model.

A white rectangular object with black text

Description automatically generated

The MSE for this decision tree was 2,277,347.9. This translates to an RMSE of 1509 college applications. This result is actually a bit worse than the RMSE from our simple linear model that we explored previously.

**Use LOOCV to determine whether pruning is helpful and determine the optimal size for the pruned tree. Compare the pruned and un-pruned trees. Report MSE for the pruned tree. Which predictors seem to be the most important?**

We wanted to see if pruning our tree would improve our prediction so we used Leave One Out Cross Validation (LOOCV), the cross validation curve graph below shows that our tree had the smallest error rate when it had 8 terminal nodes.

A graph with a line graph

Description automatically generated

This means that our MSE for the “pruned” decision tree was identical to our first decision tree, 2,277,347.9. This translates to an RMSE of 1509 college applications. In a future section we will explore what happens when we fit a larger decision tree.

**Use a bagging approach to analyze the data with B = 500 and B = 1000. Compute the MSE.  
Which predictors seem to be the most important?**

We built a bagging model using the same data that we’ve been using this entire time. The bagging approach involves creating multiple decision trees from bootstrapped samples of our original data. Note that by bootstrapping our data, we get decision trees that aren’t correlated. The B, in this context is the number of trees we want our model to create and consider when aggregating a final answer. As asked we fit a bagging model where B = 500, the MSE for this model was 1,047,360.81 which translates to an RMSE of approximately1028 college applications.

Bagging models are a low bias, high variance model which means that increasing B doesn’t necessarily lead to overfitting. Since each decision tree is built for a bootstrapped subset of data the answers from various trees will tend to vary quite a bit. The advantage of these decision tree ensemble methods is that they are low bias meaning that they don’t make a lot of assumptions about the relationship between the response and predictor variables.

With this in mind we fit another bagging model but this time with B set to 1000. This resulted in an MSE of 1,053,600.79 which translates to an RMSE of approximately 1,012 college applications.

These results are an improvement over the single decision tree that we fit but not as large of an improvement as we were expecting to see.

**Repeat (d) with a random forest approach with B = 500 and B = 1000, and m ≈ p = 3.**

Now we move on to the random forest approach. The random forest approach is similar to the bagging approach in that it is an ensemble method that involves fitting multiple decision trees that are using bootstrapped subsets of the dataset. B still refers to the number of trees we’re asking our model to generate, however in bagging we allow each decision tree to consider all of the predictors when deciding which splits yield the highest node purity. In random forests we restrict our trees to only consider a small set of the predictors which are randomly chosen. We call this restriction m, in this section we set our m = 3.

For our random forest model where B = 500 and m = 3 we got an MSE of 1,039,048.14 and an RMSE of approximately 1,019 college applications. This RMSE is just slightly worse than the bagging model.

**Compare the results from the various methods. Which method would you recommend?**

A graph with green dots

Description automatically generated

Here we show a comparison of the RMSE from each of the models discussed in the original project assignment. Based on these findings we would have to recommend the simple linear model approach, it’s computationally simple and yielded the best results for our model prediction accuracy measured in MSE. However we weren’t satisfied with these results. The final RMSE would mean that our model could only predict how many college applications a university would receive to within about 992 applications. For our smaller colleges this is all but useless.

**Additional Methods**

**Data Manipulation**

We began by creating a training and testing dataset with a 70/30 split from the original dataset. Using this data, we ran a linear regression model using least-squares and found that the most significant predictors were Accept and Top10perc. These predictors represent the number of applications that a college or university accepted and the percentage of new students that were in the top 10% of their high school class, respectively. Since many of the predictors were insignificant, we ran a reduced linear model and found that Accept and Top10perc were the most significant predictors again. This makes sense, as we can take the Accept predictor to mean how many applications a given university takes, which will be a direct result of how large a college is. Then the Top10perc predictor can be interpreted to mean the colleges that are attracting the top performing students. We put those two pieces of information together to mean that larger, more prestigious schools were receiving a higher volume of applicants.

Decision trees typically are good at taking in a lot of predictors without suffering from model complexity but in the interests of curiosity we decided to move forward with the reduced data set, and our new 70/30 split of our data set.

**Additional Models**

As promised we went back and fit a tree with more terminal nodes. When researching how to achieve this in R we found that the argument mincut = n could be passed into the tree() function. The mincut argument specifies the minimum number of observations allowed in a terminal node. This doesn’t guarantee a larger tree but passing a smaller value for n increases the likelihood. Setting mincut all the way down to 2 we were able to produce a single decision tree with 9 leaves. However when we ran LOOCV it showed that the MSE started to go back up if we increased the number of terminal nodes from 8, confirming that our original work was the best choice.

Next we fit a BART model to the data. BART is another ensemble decision tree method. Bagging and random forest approaches will sum up the final results from all of their trees and report an average for regression problems or take a majority vote in the case of classification. BART models, on the other hand, are based on a Bayesian framework. Instead of averaging the predictions of individual trees, BART models combine the predictions using a Bayesian approach, where the uncertainty associated with each prediction is quantified using posterior distributions. BART models use a sum-of-trees model structure, where the ensemble is formed by combining multiple regression trees in a Bayesian framework.

To our dismay our BART method did even worse than a single decision tree. At this time our assumption is that it’s too sensitive to noise, and potentially overfit our data set.

We re-ran our original models and didn’t find significant differences in our MSE. However we did start to notice how wildly our results could vary based on the seed that we set in our R code. We went back and tried to really understand our data set. It’s at this point that we discovered that our response variables had such a huge gap. We also began to understand that our data had a pretty natural linear relationship.

Next we decided to explore a gradient boosting model. The GBM is still an ensemble method, however it differs significantly from our previous approaches in that it is what we call a “slow learner.” Instead of creating a large number of trees and summing them up, instead the GBM makes one split typically referred to as a decision stump, or a weak learner. For each weak learner, the model stops to apply a loss function which calculates the discrepancy between the model's predicted values and the actual target values. Its primary purpose is to quantify the error or loss incurred by the model's predictions. After that, it recalibrates as it goes making improvements along the way.

A graph with green dots

Description automatically generated

When comparing our models we found that our boosting model had the best performance with a RMSE of 933. This means that we were able to improve beyond a simple linear model through machine learning techniques, but not by much.

Our next thought was to transform our response variable in order to make our methods scalable for smaller universities. We tried a simple logarithmic transformation. After we transformed the response variable, we re-ran all of our code and got some interesting results.

A graph with green dots

Description automatically generated

Suddenly our decision tree models are performing much better than before.

Our boost model ended up with an adjusted RMSE of just 1.21. We would have to do further investigation of how practical our transformation was, as prediction accuracy to within 1 or 2 college applications seems a bit too good to be true. However our conclusion is that exploring further data manipulation got us a much better result than the initial methods we tried.