Kaggle Competition Report

The fields of Data Mining and Machine Learning are home to a variety of powerful mathematical and statistical algorithms/models that are capable of great displays of predictive power. In this Kaggle competition we were challenged to predict a continuous variable Y for each corresponding ID given 15 numerical predictors labeled "X1", "X2", …, "X15". In order to find the best RMSE we performed regression methods ranging from linear/polynomial regression to bagging trees. Overall, we discover that random forests allow us to beat the best benchmark model by almost a factor of RMSE = 0.02.

First, since we are only given one set of data that not only has the predictors but also the outcome (this is the training dataset) we first split this data into a localized training and test dataset. Therefore, I split the original training dataset into a sample dataset with 70% of the observations and a sample test set with 30% of the observations. In doing so, I am able to develop models using the sample train set and then find a test MSE using the sample test set. This gives me a better idea of the RMSE I would get when submitting to Kaggle without necessarily wasting attempts. Next, since this is a regression problem, simple linear regression was the obvious starting point. Before performing this I decided to analyze the correlations between all our variables to see if there were any that had a high correlation with the dependent variable, Y. As shown in Figure 1, most of the correlations between the first variable, Y, and the rest of the predictors are very small with the largest only being X4 with corr = -0.19. Given there was no obvious best set of predictors from this analysis, I preceded to perform linear regression with the full model and got $R_{adj}^{2} = 0.08236$, too low to consider for a Kaggle submission.

The next best way to improve the model was using shrinkage methods because we could reduce the effects of the unimportant variables and reward the best predictors. It is less flexible

than linear regression and will give better predictions since its increase in bias is less than its decrease in variance. I started with Ridge Regression since this form of shrinkage does not completely remove the effects of less important variables—unlike that of Lasso Regression—and I was unsure if still having small effects of these less important variables might result in better test MSEs. I also performed Lasso Regression immediately afterwards to compare their test MSEs. In both cases I used standardization of the predictors and performed cross validation on their respective lambdas. The results of the cross-validation can be seen in Figure 2 and Figure 3 for ridge and lasso respectively: $\lambda_{ridge,\,min} = 0.2848$ and $\lambda_{lasso,\,min} = 0.0403$. With these lambdas, we get the following test MSE, 1.9335 for ridge regression and 1.859 for lasso regression. Clearly, lasso had a better test MSE and model than ridge regression, even though it did not set any coefficients to zero as I had hypothesized. I submitted the predictions from the lasso modele on the kaggle competition and found that it only beat the first two benchmark models; it was clear I had to try more sophisticated approaches.

At this point I diverted my attention to tree based algorithms and started with an unpruned deep regression tree as shown in Figure 4. From this figure, we can see that there are 11 splits—with the most important predictors being towards the top: X4, X12, X2, etc— and 12 terminal nodes; the resulting test MSE was 1.5564, a significant drop to the previous approaches. However, before submission I attempted to prune the tree using cross-validation to determine the optimal tree size. The results shown in Figure 5 tell us that the optimal size of this tree is three. Using this height I built the regression tree shown in Figure 6 with just two splits for X4 and X12; the resulting test MSE was 1.8620. At this point I was at an impasse on which of the two models to submit since the deep tree gave a much better MSE but was very likely overfitting,

while the pruned tree gave a simpler model that could have been underfitting. In submitting both models, I was able to surpass the third benchmark model but not the last benchmark model.

After this attempt, I thought about how deep trees tend to have low bias but high variance while small trees have high bias but low variance. In realizing this, I recalled that "Bagging" is a method used to deal with low bias but high variance, solving the issue when using deep trees. Therefore, I decided to use bagging; in doing so, I had to determine the best lambda that gave the best testing MSE. As shown in Figure 7, this lambda value is close to zero. After a little experimentation with the hyper parameters, I developed a model that had $MSE_{test} = 1.6601$, using 5,000 trees and 0.001 for the shrinkage value. After submitting this to Kaggle, I discovered that I was about 0.01 RMSE off from beating the best benchmark model.

Finally, to beat the last benchmark model I had to make a small improvement over my boosted trees model. This brought to mind random forests which reduce variability by only allowing a random subset of m predictors to be used for the forest of trees used in bagging. First I used the "out-of-bag" method to find the best value for m; as seen in Figure 8, this was m = 2. Now, all I had left to do was determine the best value for the number of trees and the best set of predictors for the final model. I found the best predictors by performing a random forest on the full set of predictors and then using the importance function to find the most influential independent variables. The results of this function showed that "X2", "X4", "X12", "X13", "X14", and "X15" were the best predictors. Finally, I tested different values for the number of trees to get the best model. The optimal number of trees turned out to be 1,000 allowing my random trees method to reach a test MSE of 0.375—the biggest drop in error we have seen. I submitted my predictions to Kaggle with a finalized test MSE of 1.2408, beating the last benchmark model.

Appendix

Figure 1:

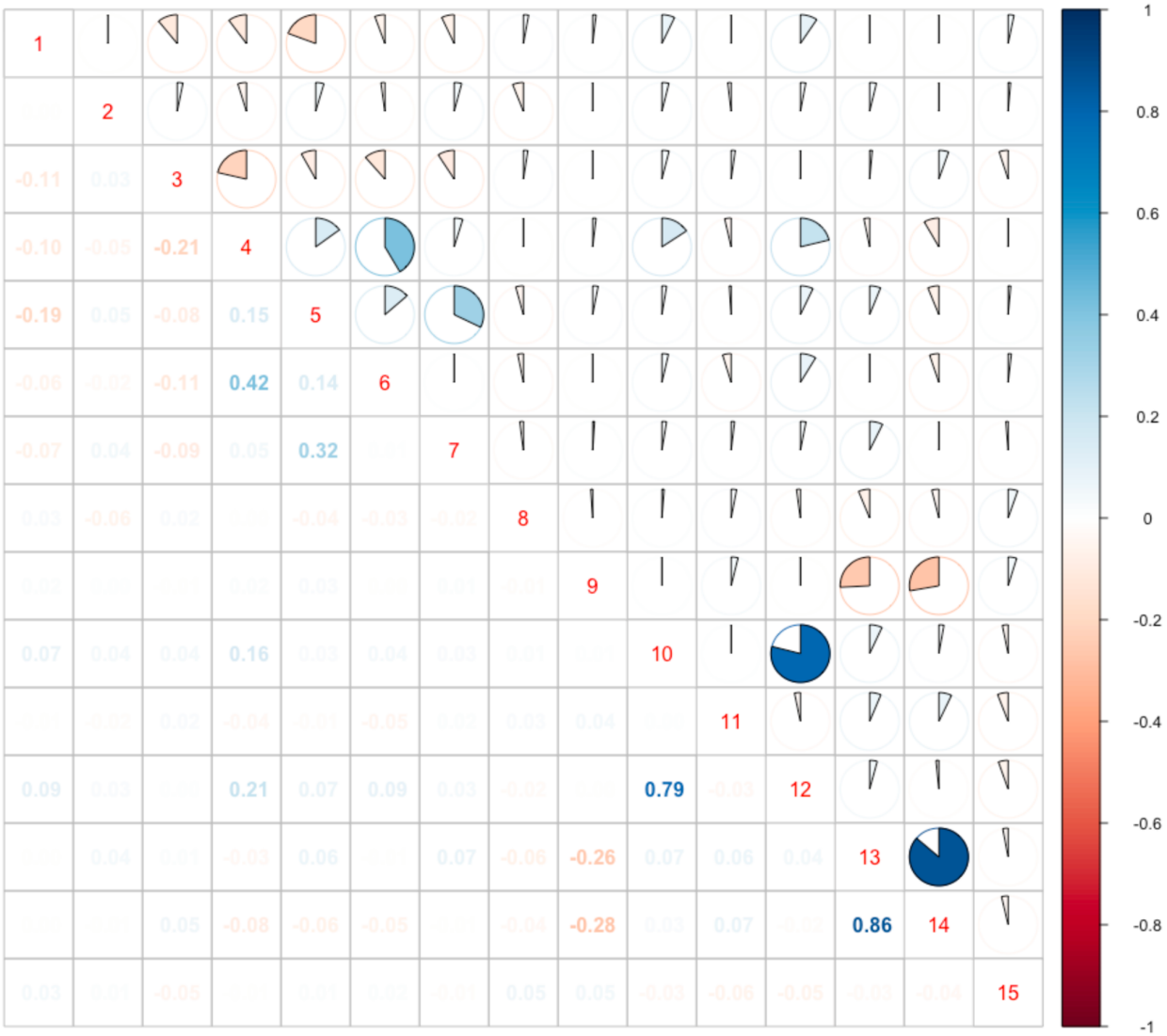


Figure 2:

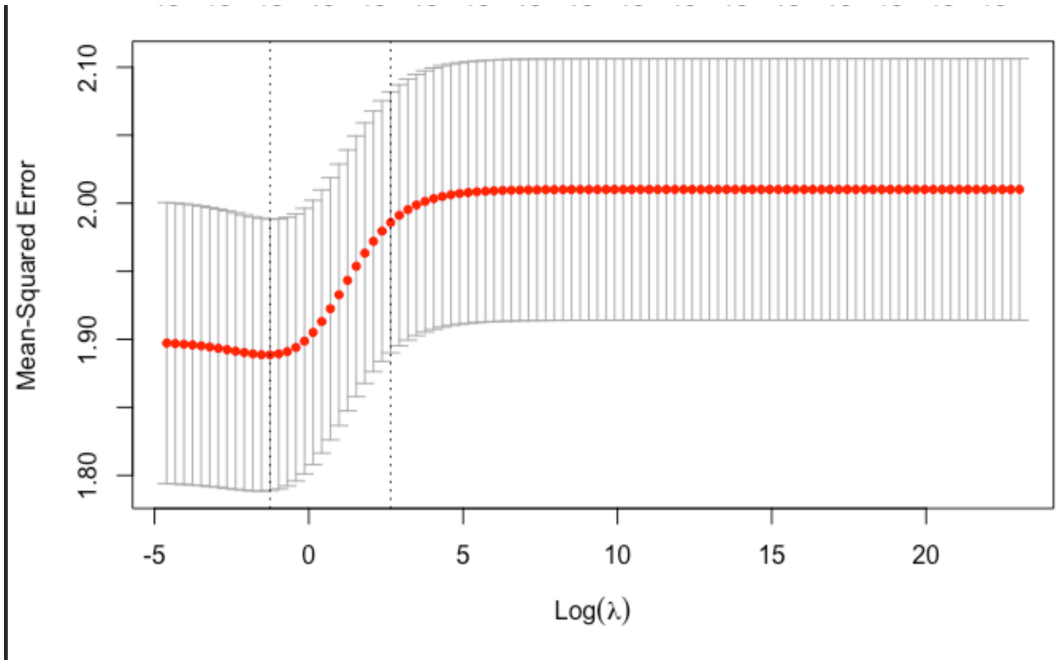


Figure 3:

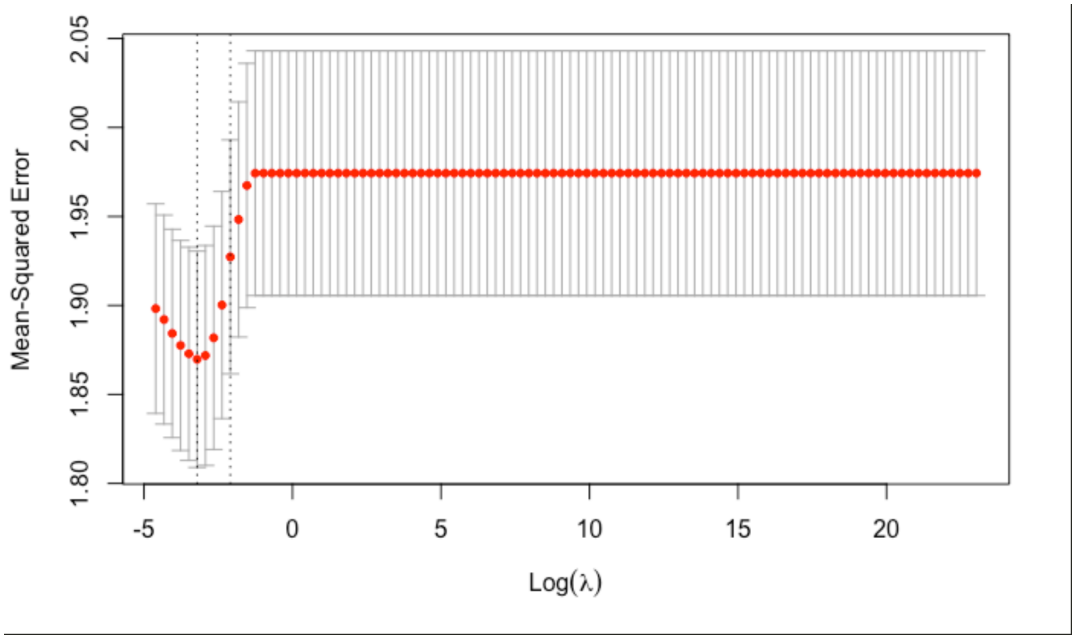


Figure 4:

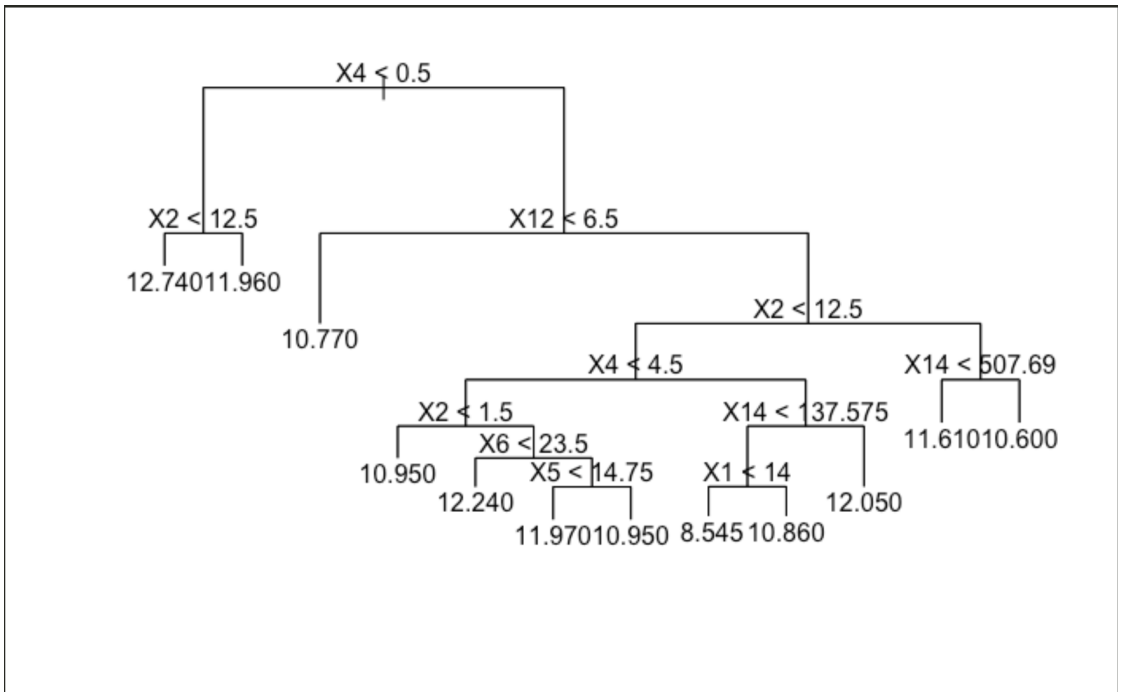


Figure 5:

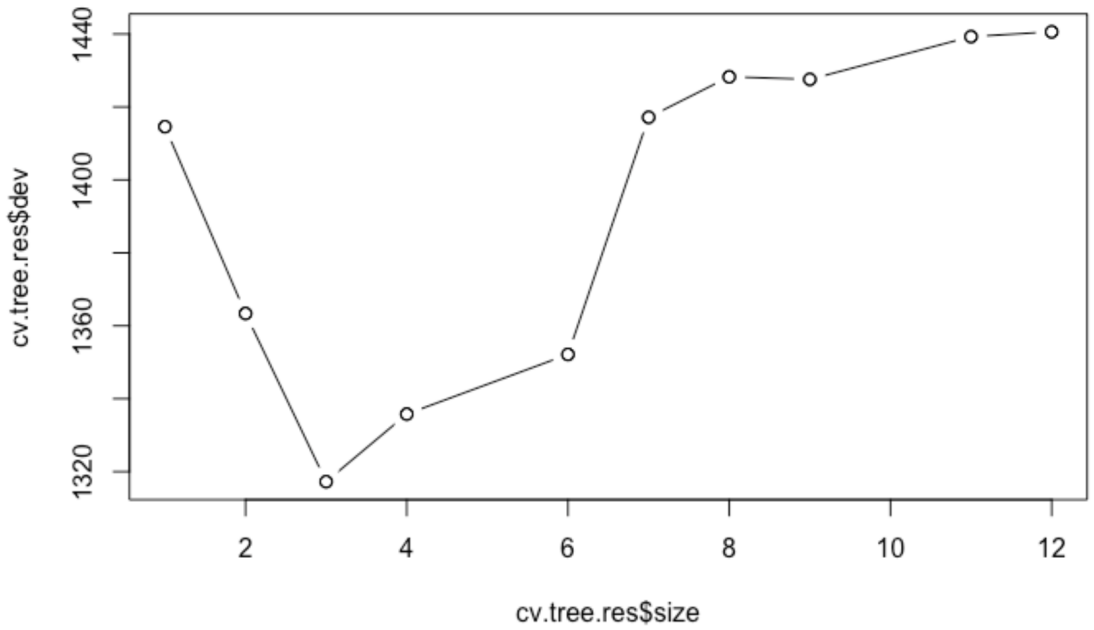


Figure 6:

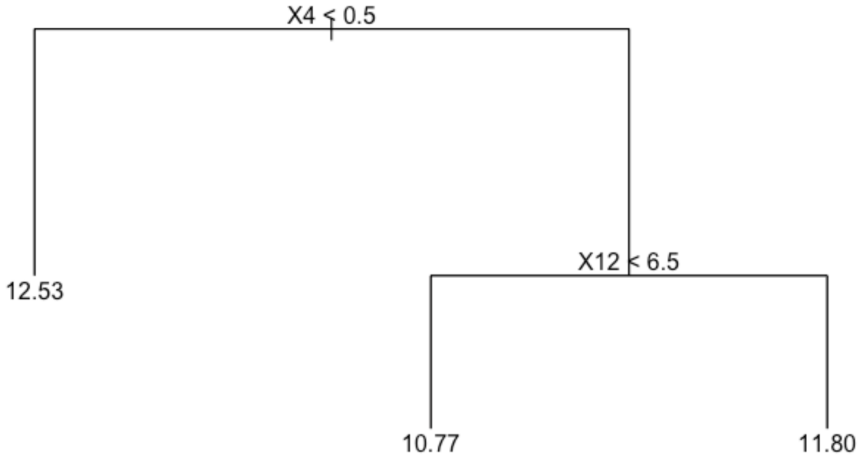


Figure 7:

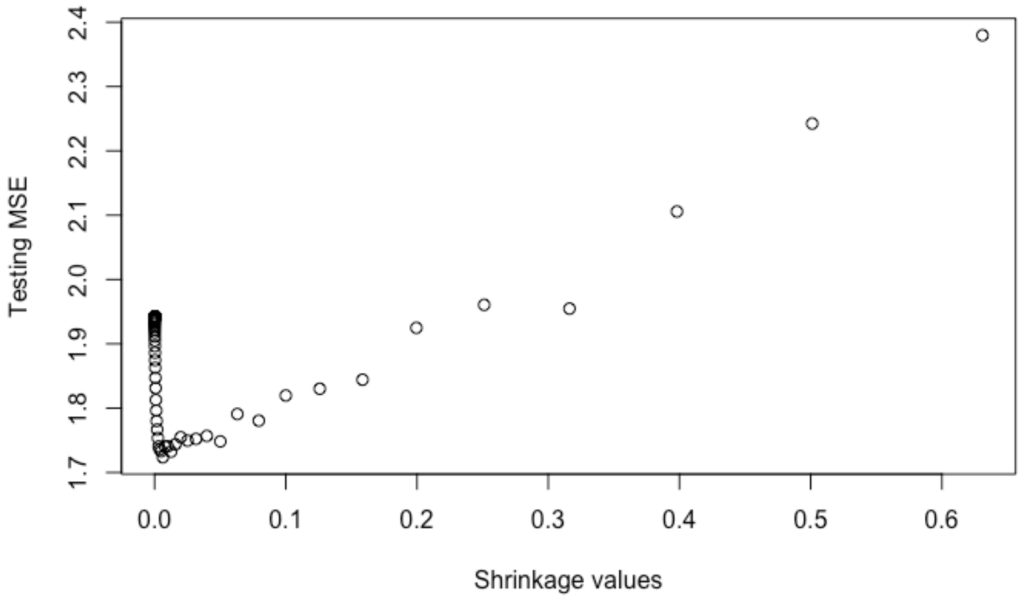
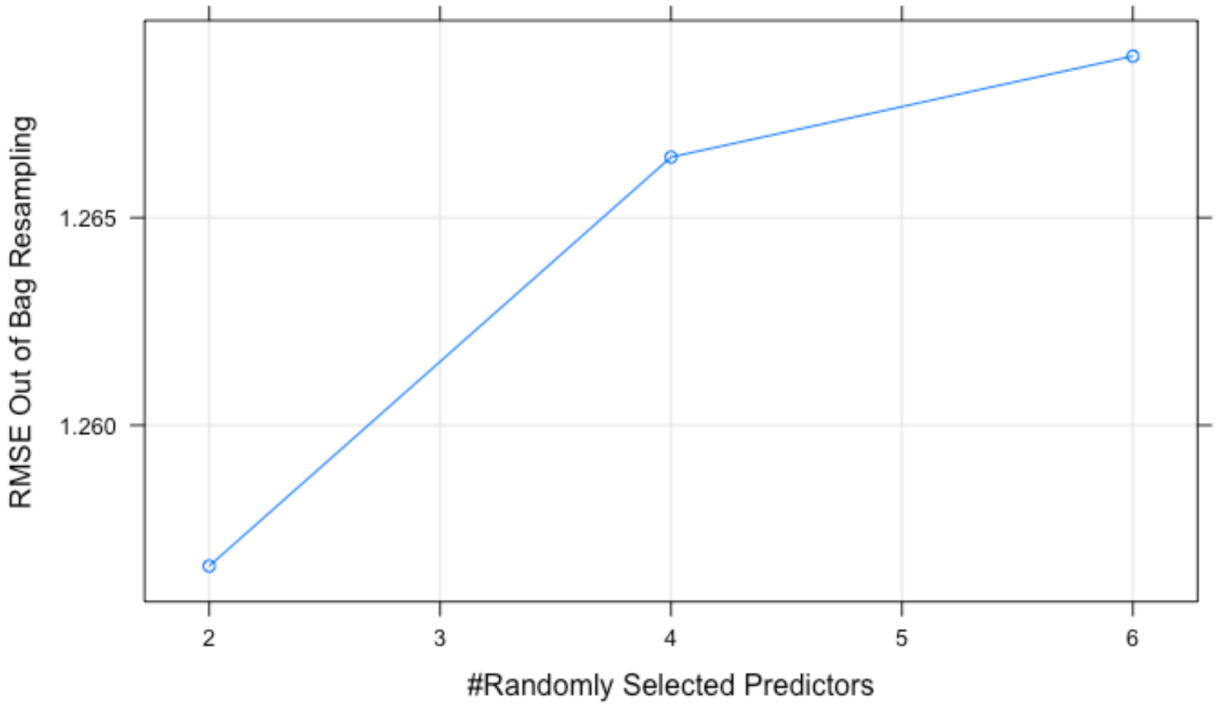


Figure 8:


RMSE Out of Bag Resampling
1.265
1.260
2
3
4
5
6
#Randomly Selected Predictors