WORCESTER POLYTECHNIC INSTITUTE

Predicting the Lifetime of Grinding Stones on Trains

Valerie Tuzel, ML Tlachac, Umesh Nair, Mo Cheng

Group 5

DS 502 Fall 2017

Grinding stones on grinding trains are used to maintain railroad tracks. These grinding stones wear out as they grind the track and must be replaced. As the stones must be replaced when the train is at rest, predicting how long these stones will last is important in planning when to replace these stones. Using data obtained from the company Loram, we investigated many methods to predict the lifespan of stones. We randomly split our data into test and train datasets and after training the models on the train dataset we used the test dataset to make our predictions. We used the root mean squared error (RMSE) as our comparison guide. In this report, we talk in detail about the methods we explored and compare the test RMSE results that lead us to the best model which resulted in the lowest test RMSE.

Contents

1. Introduction

1.1 Background

When many trains run over a railroad track, the track begins to crack. If these cracks are allowed to grow, it is unsafe to use the track and that segment of the track needs to be replaced. This is both time intensive and expensive. To prevent the need to replace the segment of a track, a grinding train is sent over the track when the cracks are small. Loram, a leading supplier of railway track maintenance machinery and services in North America, operates many of these grinding trains [1]. Loram's grinding trains contains four grinding cars, each with thirty grinding stones. When the grinding train runs over the tracks, these grinding stones grind against the track which repairs the track.

Figure 1. A grinding train operated by Loram [1].

1.2 Motivation

These grinding stones wear out as they grind the track and must be replaced. As the stones must be replaced when the train is at rest, predicting how long these stones will last helps to plan when to replace these stones.

1.3 The Objective

Our goal is to predict the lifespan of a grinding stone based on distance, angle, amperage, location, and month. In what follows, we discuss our experiments with a variety of methods encompassing linear regression, elastic net, dimensionality reduction, trees, and ensemble methods, and how we determined the method that is the most effective at predicting the lifetime of a grinding stone.

2. The Data

The data for the grinding stones was provided by Loram. Each row of the data contains information about a single grinding stone. Covering a year for a single grinding train, the dataset contains information for 11,280 grinding stones. We have been informed that no prior analysis has been done on any of the grinding stone data Loram has ever collected.

2.1 The Variables

The data from Loram contains 52 total variables. We use a continuous numeric variable Lifetime as the response variable. As the stones were replaced at different levels of wear, we only considered time and the distance variables that were calculated as if each stone was fully used. Additionally, there were some variables that were the same for every grinding stone, such as the train and year, which we removed from the dataset. We also discovered that the variable Speed in the data was calculated manually from the distance and time and hence was not an observed value. For this reason, this variable was removed from the dataset as well. After cleaning the data, we had 22 variables remaining.

The 21 predictor variables involve each stone's distance traveled in miles, angle, amperage, location on the train, and month. Specifically, seven of the variables involved the percent of time at an angle and seven of the variables involved the programmed amps at those angles. There are also three variables referring to the position of the stone, two of which we considered categorical.

2.2 Correlations

Before building predictive models, we explored the correlations among the target variable and the predictor variables. The only predictor variable that had a notable correlation with the Lifetime was the distance variable, shown in Figure 2.

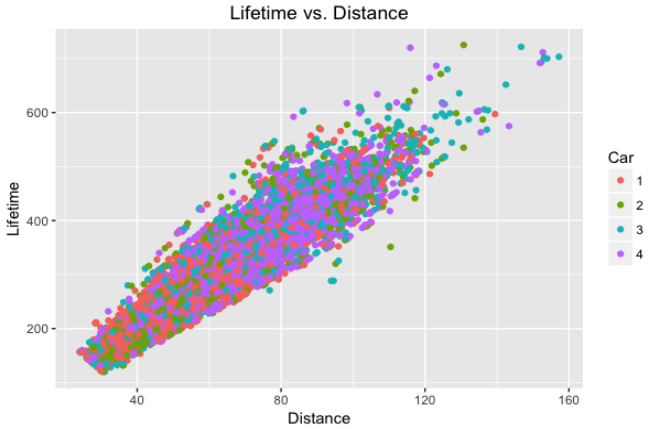

Figure 2. Plot of Lifetime vs. Distance.

3. Methods and Results

We first split our data as train and test data. 65% of the randomly chosen rows were labeled as the train data and the rest was labeled as test data to obtain the root mean squared error (RMSE) to compare each of the methods we used. We calculated the test RMSE of each model's prediction on the test data using the following formula

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2}$$

We explored several methods to decide which method provided us with the best predictions of grinding stone's lifetime. In the following we talk about these methods we explored, and the test RMSE results we used to compare.

3.1 Linear Regression

We started exploring several linear regression models by first looking at a single regression model with the distance variable as there is a clear relationship between the distance and Lifetime as seen in Figure 2. This model had an adjusted R^2 of 83.94%. Test RMSE for this model was 37.095. Figure 3 shows the plot of the actual Lifetime versus the corresponding predictions made on the test dataset.

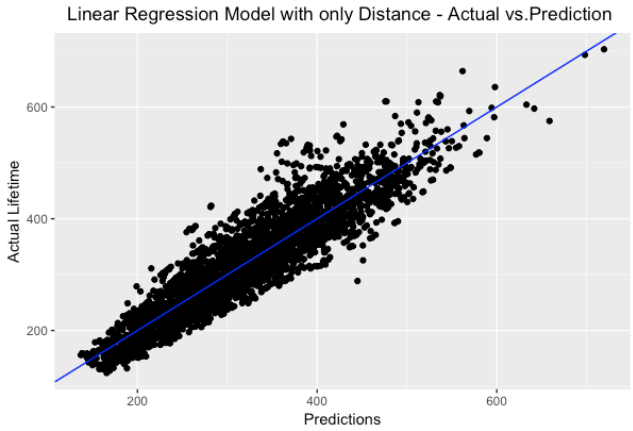

Figure 3. Actual vs. predictions of simple linear regression with the distance variable.

Next, we explored a multiple regression model where all the predictors were considered. This model had an adjusted R^2 of 85.85% and the test RMSE resulted in 34.914 which was an improvement from the previous model discussed. Figure 4 shows the plot of the actual Lifetime versus the predictions.

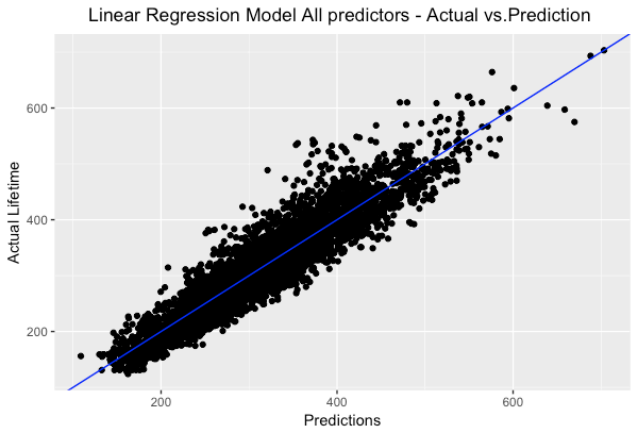

Figure 4. Actual vs. predictions of multiple linear regression with all the predictors.

Then we wanted to see if the best subset model would give us an improved model. Before we applied the best subset method, we first took out one of the predictor columns that had 6220 zeros in it and hence was deemed not very useful and also ignored the categorical variables since the best

subset method considers each category to be a variable as well. Using 10-fold cross-validation and by comparing the mean cross-validation error (MSE=$(RMSE)^2$), we found that the best subset with 16 variables had the best result, with a mean cross-validation error of 1312.310, as shown in Figure 5.

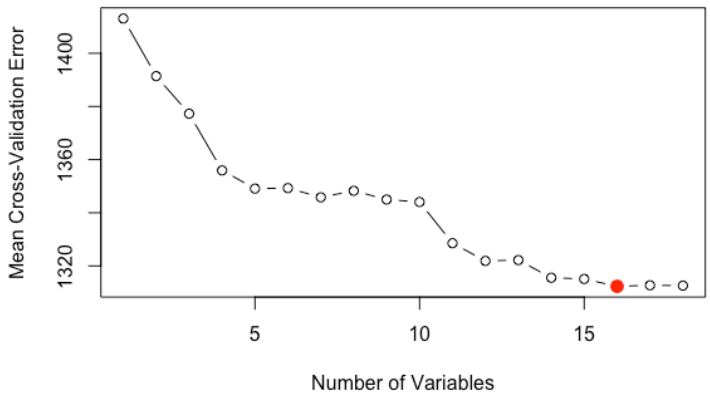

Figure 5. 10-fold cross-validation results for the best subset method.

After obtaining a list of these variables names we built a new a linear regression model using those variables. The resulting test RMSE using this model was 35.793. Next, we added one of the categorical variables to see if that would improve the results and that yielded test RMSE of 35.0248. Upon investigation, we decided to add one more numerical variable to this list to make it an 18-variable subset and our test RMSE improved to 34.898 compared to the previous test RMSE of 34.914 that we obtained using all the predictors in the model. Furthermore, the adjusted R^2 for the best subset model with 18 variables was 85.79%. Corresponding plot of the actual Lifetime versus predictions is shown in Figure 6.

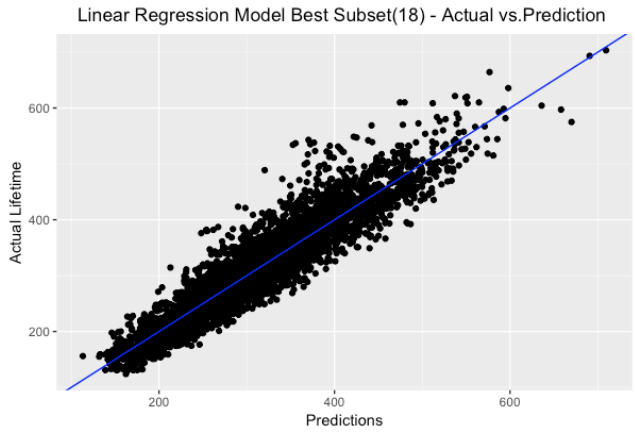

Figure 6. Actual vs. predictions of linear regression with 18 variables using best subset.

Next, we incorporated interaction terms to see whether having interaction terms between variables would improve these results further, and after some trial and error, a linear regression model with 18 variables and an interaction term between two of the variables resulted in an adjusted R^2 of 85.95% and a test RMSE of 34.777. Figure 7 shows the corresponding plot of the actual Lifetime versus predictions.

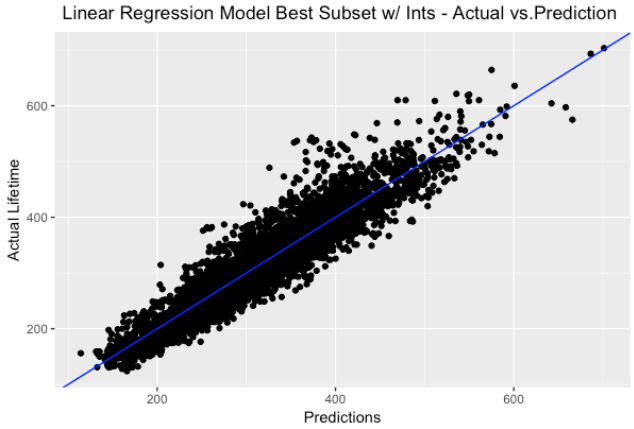

Figure 7. Actual vs. predictions of linear regression with 18 variables with interactions.

3.2 Elastic Net

We then explored the combination of the lasso and the ridge regression using elastic net [2]. In elastic net, the penalty term applied to minimize the sum of the squared residuals is a linear combination of the penalties used for the lasso and the ridge as follows

$$\sum_{i}^{p} \alpha|\beta_j| + (1-\alpha)\beta_j^2$$

We used 10-fold cross-validation for α=0,0.1,0.2, … ,0.9,1 and chose the one that resulted in the smallest RMSE. The smallest RMSE was obtained for when α=1, meaning lasso, and the corresponding test RMSE was 34.943. We also provided R with our own custom grid β values and applied 10-fold cross-validation for α=0,0.1,0.2, … ,0.9,1 again. This time, using these manually supplied β values the smallest RMSE was obtained for when α=0, meaning ridge regression, and the corresponding test RMSE was 34.920. Figures 8 and 9 show the corresponding actual Lifetime versus prediction results.

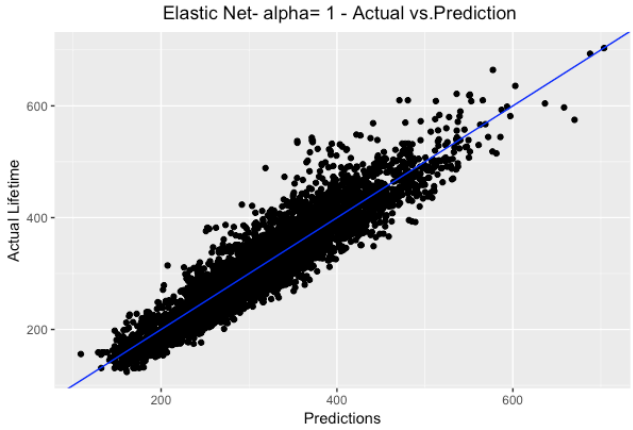

Figure 8. Actual vs. predictions of elastic net for when α=1.

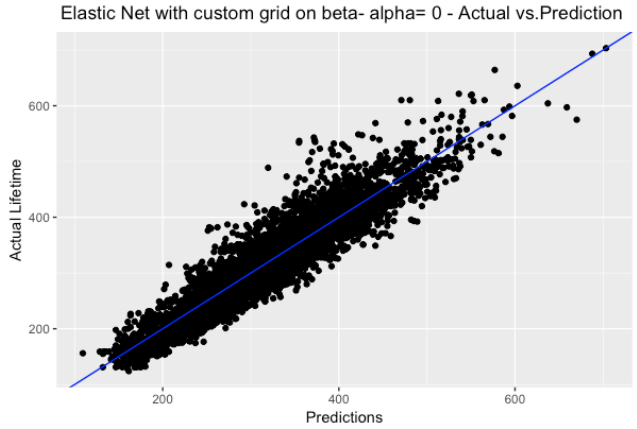

Figure 9. Actual vs. predictions of elastic net with custom grid for β values, for when α=0.

3.3 Dimensionality Reduction

3.3.1 Principal Component Regression

Next, we explored principal component regression (PCR). In order to use PCR, we took out all the categorical variables along with two of the numerical variables. We used cross-validation to find the number of variables that resulted in a lower RMSE while explaining good amount of variation in the response variable—Lifetime. As shown in Figure 10, a low RMSE value was attained around 10 components and upon investigation, we found that 10 components explained 84.74% of the variation in the response variable. Using this result, we recomputed the PCR in R with 10 components and obtained a test RMSE of 36. We did not expect this test RMSE, as it did not improve the results. Figure 11 shows the corresponding actual Lifetime versus predictions plot.

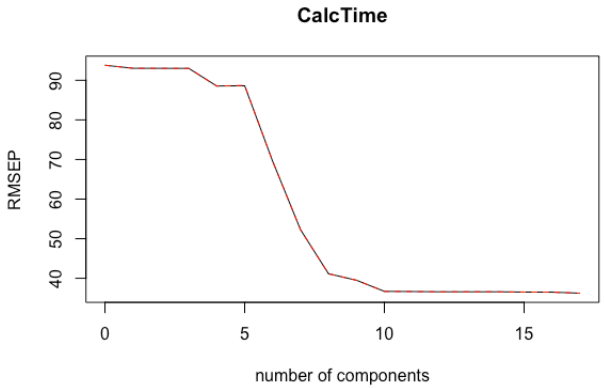

Figure 10. Cross-validation plot for the principal component regression.

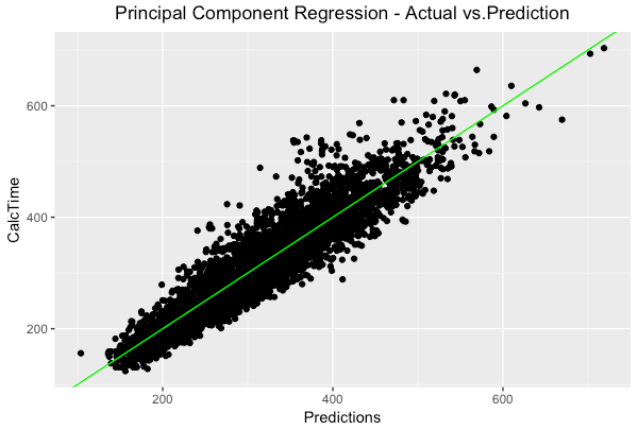

Figure 11. Actual vs. predictions of principal component regression with 10 components.

3.3.2 Partial Least Squares

We repeated the same analysis using partial least squares (PLS). By only removing the categorical variables, we sought dimension reduction using cross-validation. 4 components resulted in a lower RMSE as shown in Figure 12, while explaining 84.90% of the variation in the response variable. We then recomputed the PLS in R with 4 components and obtained a test RMSE of 36. This was again a test RMSE value we did not anticipate to obtain, as it did not improve the results. Figure 13 shows the corresponding plot of the actual Lifetime versus predictions.

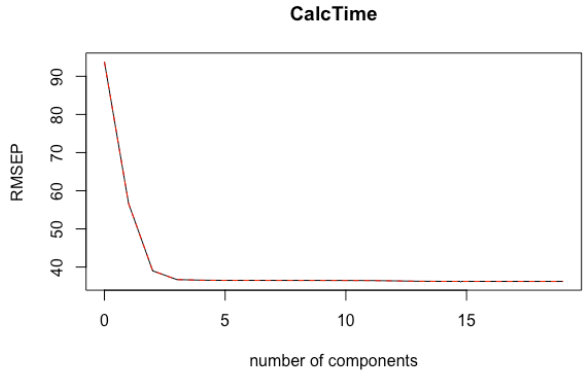

Figure 12. Cross-validation plot for the partial least squares.

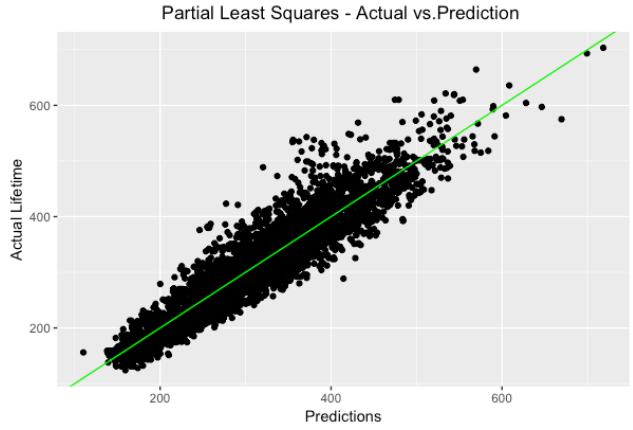

Figure 13. Actual vs. predictions of partial least squares with 4 components.

3.3.3 Principal Component Analysis

Since these RMSE results were not better than the previous ones as expected, we decided to apply principal component analysis (PCA) on our data set manually, using prcomp() function in R. In order to do this, we first took out the categorical variables. Looking at the summary of the results, we observed that the first 10 principal components explained 87.70% of the variation. Figure 14 shows the proportion of the variance versus the number of components. Next, we combined this transformed data with our original dataset and applied a multiple linear regression using these first 10 principal components and one of the categorical variables. The result was a really low test RMSE of 18.352, almost half of the previously obtained lowest test RMSE. The corresponding plot of the actual Lifetime versus predictions is shown in Figure 15.

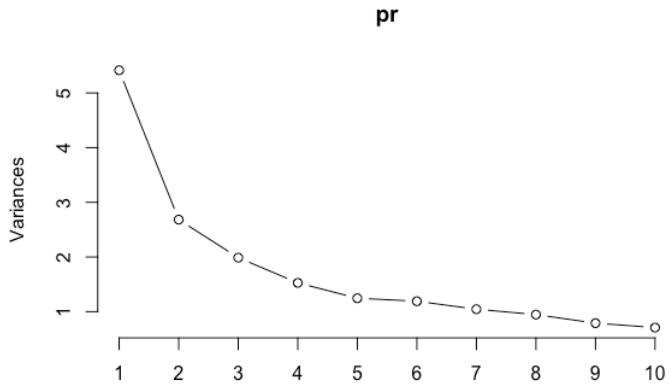

Figure 14. Proportion of the variance versus number of components.

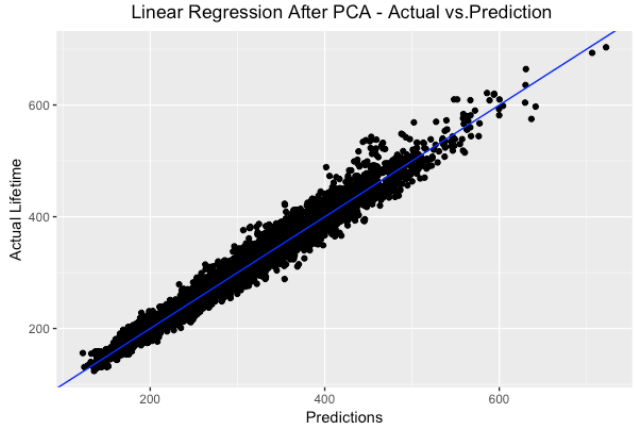

Figure 15. Actual vs. predictions of linear regression after treating the data with PCA.

3.4 Decision Trees

The next method we explored was decision tree method. When we applied the tree() function in R we obtained the following tree shown in Figure 16. The figure shows that all the splitting decisions were made on just one variable—distance. Since this tree model resulted in a very shallow tree, without even needing to prune it, we ended up getting pretty course-grain predictions with a test RMSE of 39.257, which was one of the worst test RMSE we had obtained.

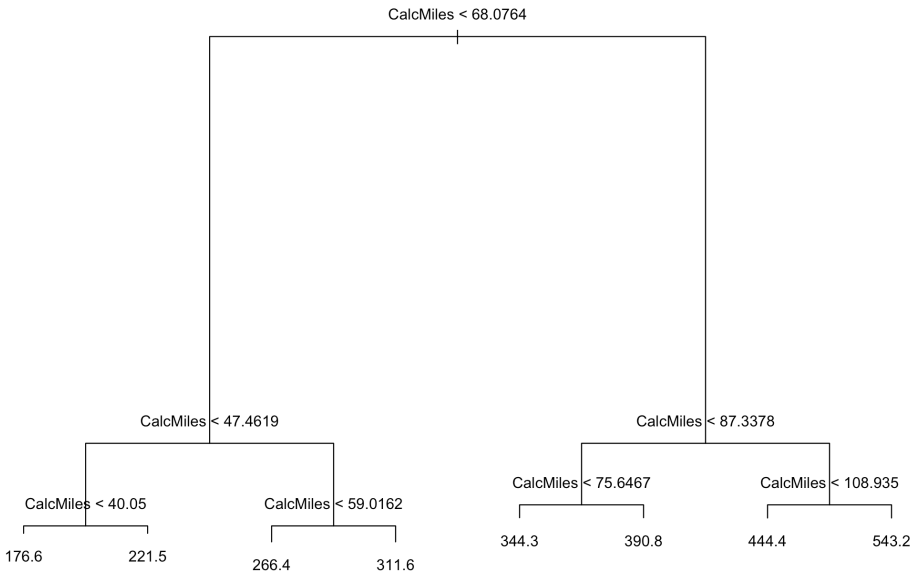

Figure 16. The un-pruned tree obtained on the training data using the tree() function in R.

Since these predictions were the worst among the methods we had tried so far, we decided to use rpart() in R to see whether we could tune the cp parameter in rpart() function to get a deeper tree with better predictions. Figure 17 shows the resulting tree from rpart with cp = 0.0004. The lowest possible test RMSE was attained only with this choice of cp value. Resulting plot of actual Lifetime versus predictions is shown in Figure 18. Corresponding test RMSE was 36.111. Even though this was an improved test RMSE from 39.257 it still was not better than the lowest RMSE we had obtained so far. This led us to explore other options like growing more than one tree using ensemble methods.

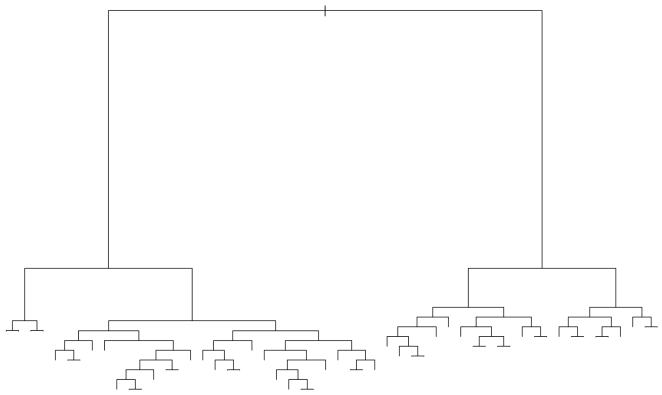

Figure 17. Tree obtained on the training data using rpart() function in R for recursive partitioning.

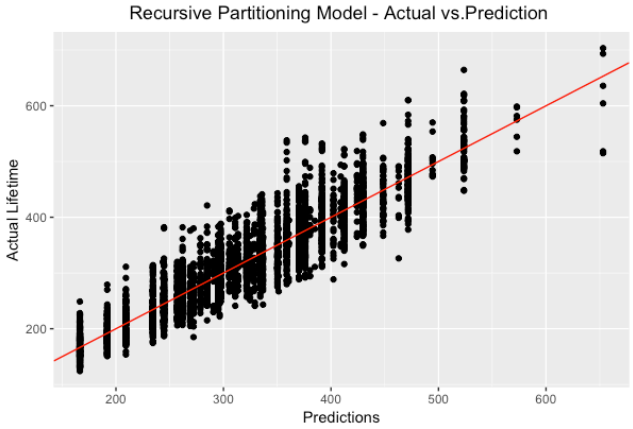

Figure 18. Actual vs. predictions of tree obtained by recursive partitioning with rpart() function in R.

3.5 Ensemble Methods

First method we explored among ensemble methods was bagging, where all the predictors (mtry=22) were considered at each splitting point. The resulting test RMSE from this method was 32.595—a good improvement from the rpart result we obtained from growing just one tree. Then, we applied the random forest method where only 10 predictors were considered at each splitting node and got a slightly improved test RMSE of 32.152. We wanted to check these results with a different random forest package called ranger. With ranger, we were able to obtain a test RMSE of 32.146 using mtry=11. Even though there is not a big difference between these two random forests test RMSE's, we deemed the ranger version to be the best of all ensemble methods we had tried so far due to the low RMSE. Corresponding actual Lifetime versus predictions plot for the random forest using the ranger package is shown in Figure 19.

Next, we wanted to see whether the PCA treated data would give better predictions as it did with the linear regression. We used the first 10 principal components and one of the categorical variables in the formula to build the random forest—just as we did while applying the linear regression model. We also set mtry to be 6 so that only 6 of these variables were considered at each splitting node decisions. Using the ranger package, we obtained a test RMSE of 18.881, which was the best among all the ensemble methods we explored. Figure 20 shows the plot of the actual Lifetime versus predictions using PCA treated data.

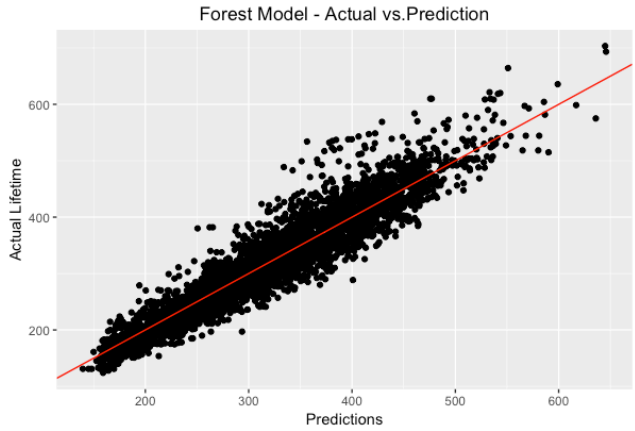

Figure 19. Actual vs. predictions of random forest obtained by using the ranger package in R.

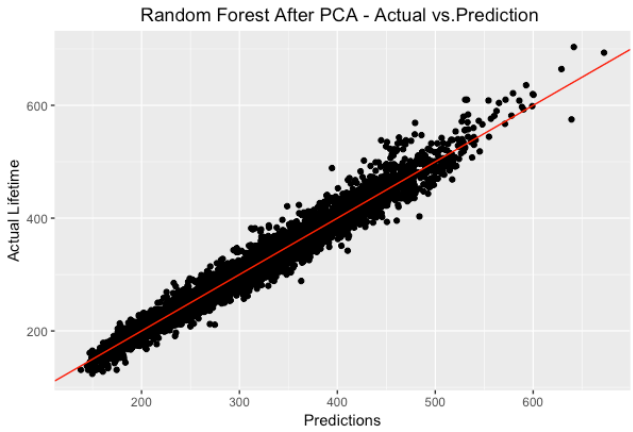

Figure 20. Actual vs. predictions of random forest obtained by using the ranger package in R after treating the data with PCA.

4. Conclusion

After exploring a variety of methods from linear regression to random forest, we found that the best results were obtained after applying the principal component analysis on our data manually using prcomp function. Even though there is not much difference between the random forest's and the linear regression's test RMSEs obtained using the PCA treated data, for the sake of choosing the one that has resulted in the lowest test RMSE, we concluded that the linear regression predictions yielded the best results. However, one could also argue that using random forest on a PCA treated data performs just as good in predictions.

After employing preliminary analysis on feature engineering and outlier removal on the single linear regression model where only the distance variable was used as the predictor, as discussed in Section 3.1, we found that whether the stone was on the left or the right of the train was inconsequential. Furthermore, condensing the percent of time at an angle into percent of time at a negative angle was similarly unhelpful. We also considered the snow potential in a particular month and incorporating this feature into the aforementioned model yielded a reduced test RMSE indicating that snow may impact the lifespan of a grinding stone.

In addition, the dataset contained the variable OverSetPoint which is the actual amps divided by the programmed amps. If OverSetPoint value is far from one it indicates equipment issues which may be considered as outliers. For this reason, we explored removing the points for which the OverSetPoint value was furthest from one in either direction. As with the engineered features, the impact of removing these outliers was tested on the single linear regression model where only the distance variable was used as the predictor, as discussed in Section 3.1. However, the removal of these outliers was inconsequential, likely due to the large volume of data or OverSetPoint outliers are not actually outliers in the data.

A further investigation of outlier removal and feature engineering could be done by applying principal component analysis to this data to see whether these would provide any improvements. Additionally, in the future, we could explore whether models like gradient boosting—which is another ensemble method that uses the previous tree model's residuals to build next trees in the forest, or K-nearest neighbors would yield better results.

References

[1] Loram.com
[2] “Regularization and variable selection via the elastic net.”, Zhou H., Hastie T., R. Statist. Soc. B **67**, Part 2, pp. 301–320 (2005).