The MSE minimization Problem

Ender Erimhan

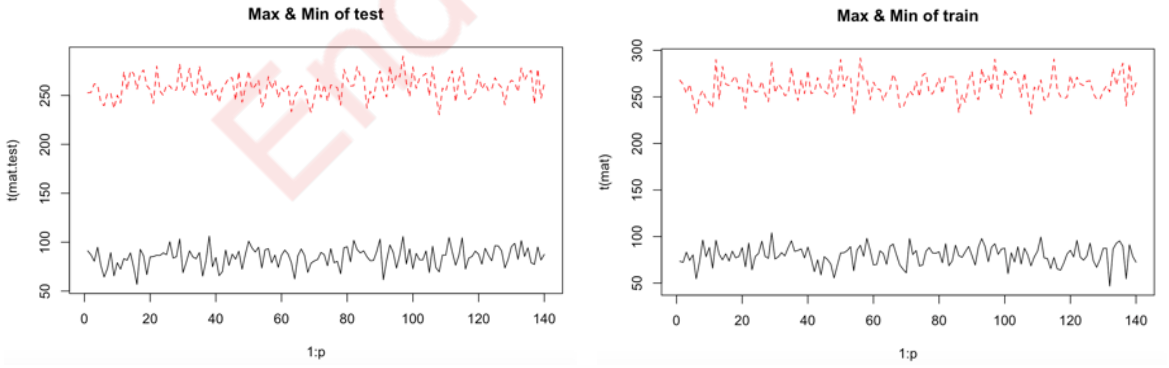
Abstract

The goal of this paper is to find a prediction algorithm that results a test mean square error (MSE) far less than the training MSE for a linear regression model. The paper begins by exploring the training data set and the test data set. We then explore different prediction algorithms with the training data to find a small enough training MSE without overfitting. We finally conclude that the best model that minimizes the test MSE is a 4^{th} degree polynomial.

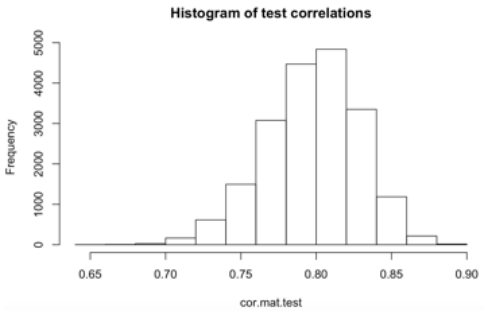
Exploratory Data Analysis

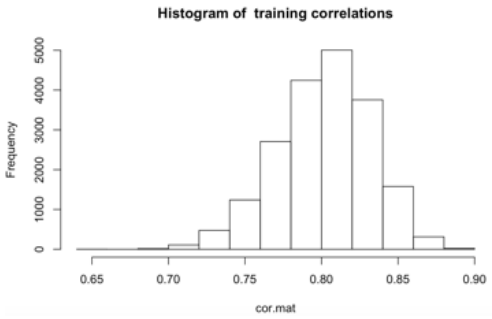
In this project, we are given a training data set and a test data set. We are not given any information on what the data means. We are given the task to find a test MSE that is "significantly less" than 300,000 using only the training data set. One of the fundamental rules of statistical machine learning is the relationship between test MSE and train MSE. As train MSE decreases, so does test MSE up to a certain point, then the test MSE will gradually increase again. This is because if the train MSE becomes too low, we overfit the training data by "memorizing" it. This causes bad predictions for the test data and in turn a higher test MSE. The challenge here is to decrease the train MSE as much as possible without overfitting.

We begin by loading in both files into R and exploring both sets. Looking at the data is always the first step in the art of Data Science. Data.train.CSV is a 5200 by 141 data frame with all numeric observations. Data.test.CSV is a 5200 by 140 data frame with all numeric observations as well. Note that the test data frame does not include the Y-column. This is something we need to find and report. Let's first look at the minimum and maximum for each predictor in both data sets:

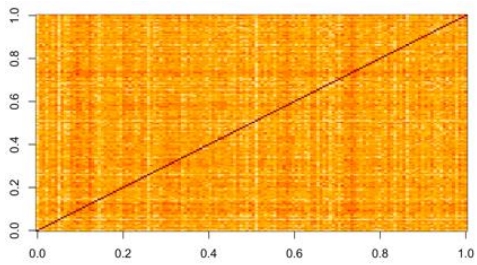


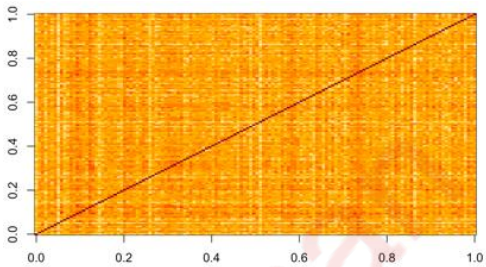
Both data sets seem to have similar maximum and minimum values. Both the data sets maximum hovers around 250-300 and there minimum around 50 to 100. We now explore the correlations between each predictor.





Both Data sets seem to have moderate to high correlation among predictors with none that are severe. Below is an image of the correlation matrix for both data sets. The colors measure correlation intensity:

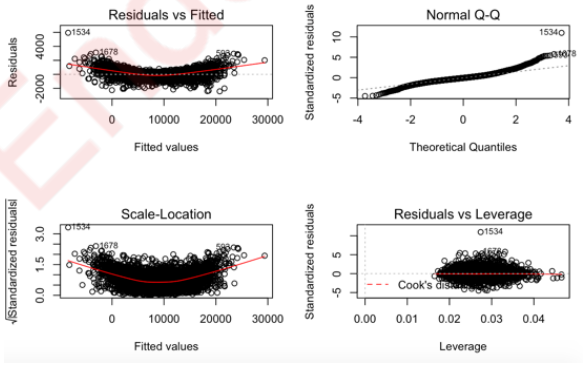




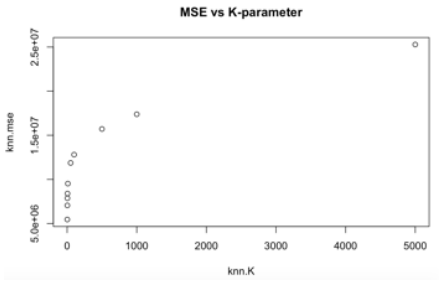
Both images seem to inhibit the same grid-like patterns. In conclusion, both data sets seem to be very similar to each other.

Regression Methods

For most models, we will calculate the train MSE to determine the predictive power of the model. This is because the train MSE creates a lower bound for our test MSEs. We first fit a multiple linear regression model as a potential prediction algorithm. We fit the model with the training data set and the results are clear.

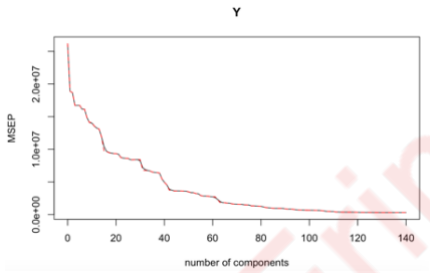


Not only is the train MSE 292,068 but we have the Normality and Constant Variance assumptions violated. Let's take a look at the residual's vs fitted plot above. The scatterplot seems to have a slight curvature to it. We will investigate this further in the section below. We throw away the linear regression model immediately and we turn to K-Nearest Neighbors Regression. Using an R-loop, we fit multiple K-nearest neighbors regression models with different K-values, and we plot the MSE. Below are the results:



At K=0 we not only have an overfit, but by definition, the results are the data set again. This is why the train MSE is 0. We ignore this K-value and look at K=2. The train MSE is 5,455,227 which is way worse than the linear regressions value. With increasing K, the train MSE increases significantly as well. None of the KNN regression models seem to be of interest to us. We now turn to Lasso Regression. We use a Cross-Validation method to find the optimal λ which turns out to be $\lambda = 0.93$. It turns out that the test MSE is 299,307. This is way better than the K-nearest neighbors but performs slightly worse than Linear Regression. We now turn to Principal Components Regression.

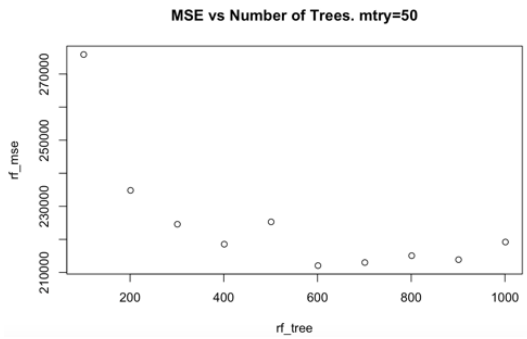
We use a validation plot on the PCR model to determine how many principal components we should consider. The graph is below:



The train MSE is best when all principal components are considered, however the train MSE results in 292,068 which is the same as the linear regression model. We temporarily end our investigation of regression models and we turn to Random Forest Models.

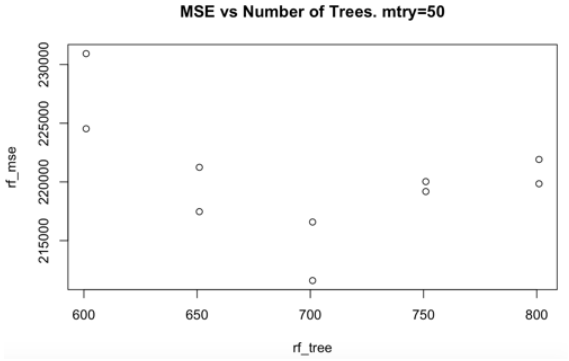
The Random Forest Model

We begin our investigation into random forest by varying the number of trees. We keep the number of variables randomly sampled at each split (mtry) fixed at 50. The graphs are done with an R-loop that calls the random forest function at each level of the parameter. The idea is to hyper tune the random forest parameters to minimize test MSE without overfitting. The minimum node size is kept at 20 to speed up time complexity without impacting the random forest fit by too much.

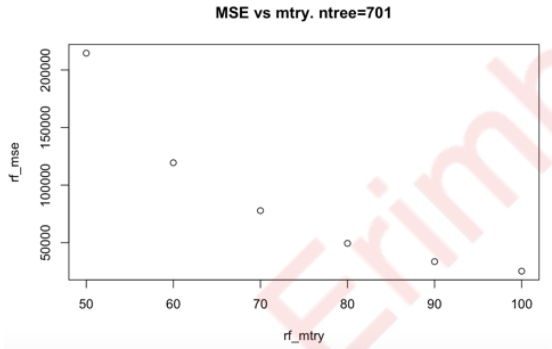


Ender Erimhan

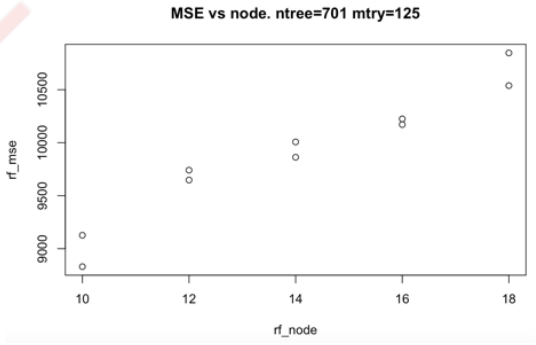
The train MSE decreases considerably from 151 trees to 851 trees before slightly increasing at 1001 trees. Lets take a closer at the number of trees between 601 and 801. For each tree, we fit a random forest model twice to get a better idea of where the train MSE will lie.



701 trees are a great candidate to move forward with our model. We now tune the mtry parameter while keeping the number of trees fixed at 701.



Let P be the number of parameters in the training data set. Increasing mtry from P/3 to P/2 creates a noticeable difference. The train MSE goes from above 200,000 to around 25,000. This does considerably better than our initial random forest model. We now keep mtry fixed at 125 and ntree fixed at 701. We now tune the node parameter. We do not want to overfit the training data, thus we choose nodesize = 14 to keep the training MSE around 10,000.



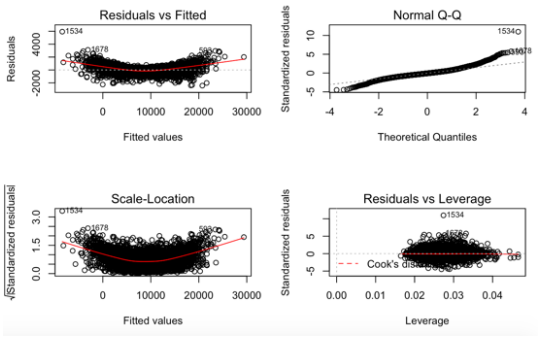
We choose our final random forest model with parameters ntree=701, mtry=125 and nodesize = 14.

Ender Erimhan

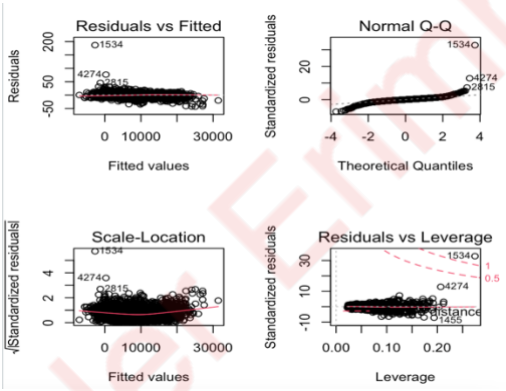
Polynomial Regression Models

Let's take a look at our diagnostic plots from the linear regression model above:

.



The Residuals vs Fitted plot seems to indicate that higher-order polynomials should be fit to capture the curvature. We begin with Quadratic Regression by extending the training data frame to include squared features. We then fit a Linear Regression model on the original and squared features. The diagnostic plot is below:



The curvature of the Residuals Vs Fitted plot seems to disappear. The training MSE decreases significantly to just 42.269. Quadratic Regression seems to do way better than the random forest model. However, is this an overfit? Let's use cross-validation to determine our test MSE. 4-fold cross validation gives a test MSE of 56.5 which indicates that Quadratic Regression does not overfit our data. Using a similar approach, we fit a 3rd , 4th , 5th and 6th degree polynomial and we record the test and train MSE below:

```
> trainvalues
           Quadratic    Cubic  Quartic  Quintic 6-degree
1 Train MSE  42.26868 3.986436 3.634196 3.507091 3.383086
```

As the degree of the polynomial increases, the train MSE decreases. This is to be expected because higher order polynomials tend to "memorize" the data. When we take a look at the test MSEs, the Bias-Variance trade off becomes abundantly clear:

```
> sapply(X = values , FUN = mean)
Quadratic     Cubic   Quartic   Quintic  6-degree
56.352751  5.530639  4.934439  6.241355 15.070066
```

As the degree of the polynomial increases, the test MSE decreases at first, then begins to increase again. We conclude that the best model that minimizes both the test and train MSE is a quartic polynomial.

Conclusion

In this project, we learned that while it is important to minimize train MSE, minimizing it too much will cause the model to have low bias and high variance. This means that the model overfits the training data set, causing any test data set to produce larger test MSEs. It is important to balance bias and variance and anyone who can master this, masters the art of machine learning. We conclude that the best model that fits the training data without overfitting is a Quartic Polynomial Regression model.

Ender Erimhan