st538 Project 2 Week 8

Philip Ourso

5/22/2022

## Project #2 - Interim Report Week 8

### Introduction

For this project we chose dataset 5, the Kaggle Product Line Performance Challenge. Bosch's challenge to Kaggle users is to predict component failure based on measurements made throughout the manufacturing process. Each component, or observation, includes a large amount of numerical, categorical and date data. These features are anonymized, but identified by a feature number, as well as the line and station ID on which the measurement was made. The date features indicate the date on which the measurement was made.

The components are uniquely identified and labelled as to whether they failed quality control. Given that a successful manufacturing company would naturally minimize failures, it is expected that the data would be highly imbalanced.

### Feature extraction and description

As the specifying the exact features would likely reveal trade secrets, features were anonymized and represented in the format:

> L<Line #>\_S<Station #>\_F<Feature #>

The precise meaning of "line", "station" and "feature" is not provided but given the manufacturing context it is likely that features refer to specific measurements, stations refer to specific tools and lines refer to the overall sequence of operations. It would not be surprising for there to be redundancy in the lines and stations: each line might produce the same component and hence consist of the same types of stations. Furthermore, each line would likely have redundant stations to enable increased throughput.

Assuming redundancies in lines and stations, then the feature labels themselves *could* serve as features themselves: which specific station processed a component is useful information, as the individual station performance would vary.

The datasets were provided in several CSV files: training and test data were separated and presumably no response data was provided for test data. Training data was exclusively used for this project and was split across three files: numerical, categorical and datetime data. The datetime data indicated when the numerical and categorical data was collected for each observation.

There were nearly 1,200,000 observations, with 968 numerical features, 2140 categorical features and their associated datetime data.

**Building classifiers and results**

Given the sheer volume of data, it was necessary to reduce the dataset to something that could be tackled on a personal computer. Using the full dataset would require access to massively parallel cloud computing resources. While the entire numerical data set could be loaded into memory, it was impossible to do so for the categorical data, so only 200,000 rows was read from the numerical and categorical CSVs. Datetime data was not used.

Once in memory, a train-test split of the 200,000 observations in a ratio of 3:1 was written back to disk, with categorical and numerical data joined. Even 150,000 observations is infeasible, so a subset of 50,000 observations was loaded and split 4:1 to serve as training and validation data.

Features and response were separated; defective components accounted for ~0.55% of the training data, confirming the highly imbalanced nature of the dataset. A classification regression tree was fit to the full feature set and 6 variables were considered for the first split. Of the 6 features, L1\_S24\_F1846 was found to be the most important and a split of:

> L1\_S24\_F1846 >= -0.3005

was used to assign observations to the defective component class. This was found to improve upon the null model of just guessing the majority class (non-defective), but over 37,000 observations were missing data for this feature, so it was not a very relevant split.

The misclassification rate on training data was ~0.547% with the below table indicating a non-trivial number of bad parts were predicted to be good. This is clearly not an acceptable trade-off: while it is expensive to fail good parts, it is likely much more expensive to sell bad parts, considering both the cost of the customer return and the impact to brand image.

Table Classification Tree Training Results

|  |  |  |
| --- | --- | --- |
|  | Predicted Good | Predicted Bad |
| Good | 39786 | 5 |
| Bad | 214 | 8 |

The poor performance of the single decision tree was reflected in the validation dataset results, while the misclassification rate was ~0.43 %, this was primarily due to classifying defective parts as good.

Table Classification Tree Validation Results

|  |  |  |
| --- | --- | --- |
|  | Predicted Good | Predicted Bad |
| Good | 9944 | 2 |
| Bad | 41 | 0 |

The next attempt at fitting a classifier used the random forest ensemble method. Fitting this model took several hours, likely due to the large amount of missing data. In order to successfully fit a model at all NAs needed to be handled, so a coarse solution used na.roughfix. This imputed missing numerical data with the corresponding column median and categorical data with the most common factor value.

Prediction accuracy deteriorated, with a misclassification rate of ~0.64 %. More concerningly, the number of bad parts predicted good increased slightly. Due to the large amount of time required to train, validation misclassification rate is pending.

Table Random Forest Training Results

|  |  |  |
| --- | --- | --- |
|  | Predicted Good | Predicted Bad |
| Good | 39751 | 40 |
| Bad | 216 | 6 |

Extreme gradient boosting produced much better results on the training dataset, but only numerical data could be retained, so categorical data was dropped. The misclassification rate dropped to 0.04 % and more importantly ~95 % of defective parts were correctly classified as such. However, this is likely due to overfitting; misclassification rate on validation data was 0.53% and less than 5 % of bad parts were identified.

Attempting to increase generalizability by train a boosting classification tree with an early exit showed a more modest improvement in the misclassification rate, ~0.51 % with only ~7.5% of defective parts correctly identified. Validation set performance did improve to 4.5%, but no defective parts were identified.

Table Extreme Gradient Boosting Training Results

|  |  |  |
| --- | --- | --- |
|  | Predicted Good | Predicted Bad |
| Good | 39785 | 6 |
| Bad | 11 | 211 |

Table Extreme Gradient Boosting Validation Results

|  |  |  |
| --- | --- | --- |
|  | Predicted Good | Predicted Bad |
| Good | 9932 | 14 |
| Bad | 39 | 2 |

Table Extreme Gradient Boosting (Early Exit) Training Results

|  |  |  |
| --- | --- | --- |
|  | Predicted Good | Predicted Bad |
| Good | 39791 | 0 |
| Bad | 205 | 17 |

Table Extreme Gradient Boosting (Early Exit) Validation Results

|  |  |  |
| --- | --- | --- |
|  | Predicted Good | Predicted Bad |
| Good | 9942 | 4 |
| Bad | 41 | 0 |

**Appendix: R code**

