Reduce Manufacturing Failures

Machine Learning Engineer Nanodegree

Chandrashekar Gunashekaran
September 25th, 2016

Definition

Project Overview

Manufacturing Failures System are to help manufacturing companies, has an imperative to ensure that the recipes for the production of its advanced mechanical components are of the highest quality and safety standards. Parts of doing so is closely monitoring its parts as they progress through the manufacturing processes. Because companies records data at every step along its assembly lines, they have the ability to apply advance analytics to improve these manufacturing processes. If the companies like Bosh use the manufacturing failures system it intimates the failure during the manufacturing process.

In this project, we will build a manufacturing failure systems to predict internal failures using thousands of measurements and tests made for each component along the assembly line. This would enable Bosh to bring quality products at lower costs to the end user.

Problem Statement

We got data which represents measurements of parts as they move through Bosh's production lines. Each part has unique Id. The goal is to predict which parts will fail quality control (represented by a 'Response' = 1).

The dataset contains an extremely large number of anonymised features. Features are named according to a convention that tells you the production line, the station on the line, and a feature number. E.g. **L3_S36_F3939** is a feature measured on **line 3**, **station 36**, and **feature number 3939**.

On account of the large size of the dataset, we have separated the files by the type of feature they contain: numerical, categorical, and finally, a file with date features. The date features provide a timestamp for when each measurement was taken. Each date column ends in a number that corresponds to the previous feature number. E.g. the value of **L0_S0_D1** is the time at which **L0_S0_F0** was taken.

File Descriptions:

- **train_numeric.csv** - the training set numeric features (this file contains the 'Response' variable)
- **test_numeric.csv** - the test set numeric features
- **train_categorical.csv** - the training set categorical features.

For training the model we consider **train_numeric** which has the numeric features of each station. We got 1,183,000 numeric data for training, since the dataset is really large, we are not going to use all numeric for training.

Metrics

We need to estimate the parts which will fail quality control represented by a Response = 1.

The evaluation for this task is Matthews correlation coefficient (MCC) between the predicted and the observed response. The MCC is given by:

$$\mathrm{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Where TP is the number of true positives, TN the number of true negatives, FP the number of false positives, and FN the number of false negatives.

The Matthews correlation coefficient is used in machine learning as a measure of the quality of binary (two-class) classifications. It takes into account true and false positives and negatives and is generally regarded as a balanced measure which can be used even if the classes are of very different sizes. The MCC is in essence a correlation coefficient value between -1 and +1. A coefficient of +1 represents a perfect prediction, 0 an average random prediction and -1 an inverse prediction.

Since the dataset has 99.4% positive response (response = 0), only 0.6% negative response (response = 1) and also MCC is not affected by class size. MCC would be the best metrics for this project.

Analysis

Data Exploration

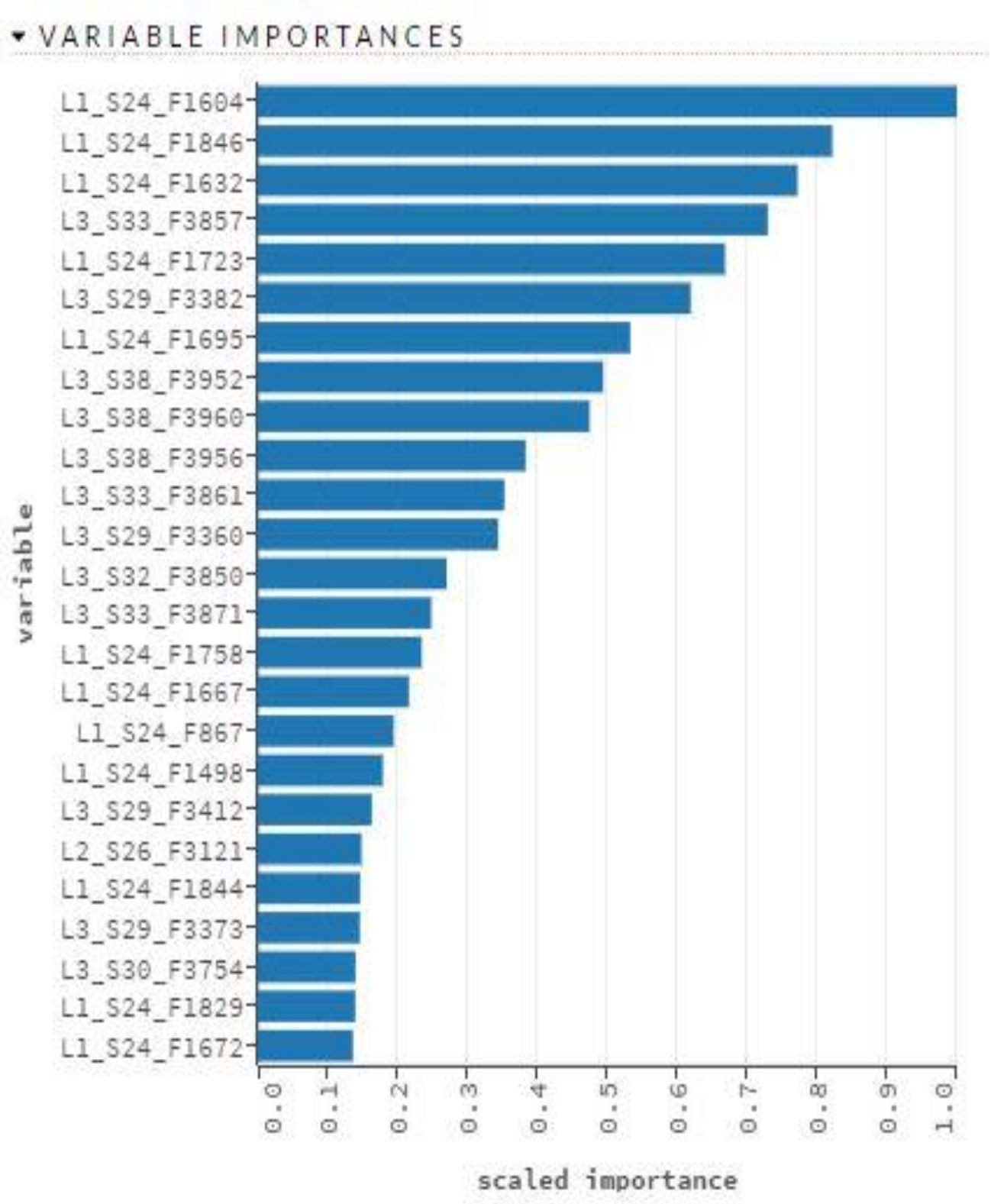
The train numeric data is from train_numeric.csv. The dataset has 1183000 observations, 968 features and missing values which do not seem to be missing at random, the response we are playing with an imbalance ratio of 1:172(172 negatives for 1 positive) 99.4% accuracy.

▾ COLUMN SUMMARIES

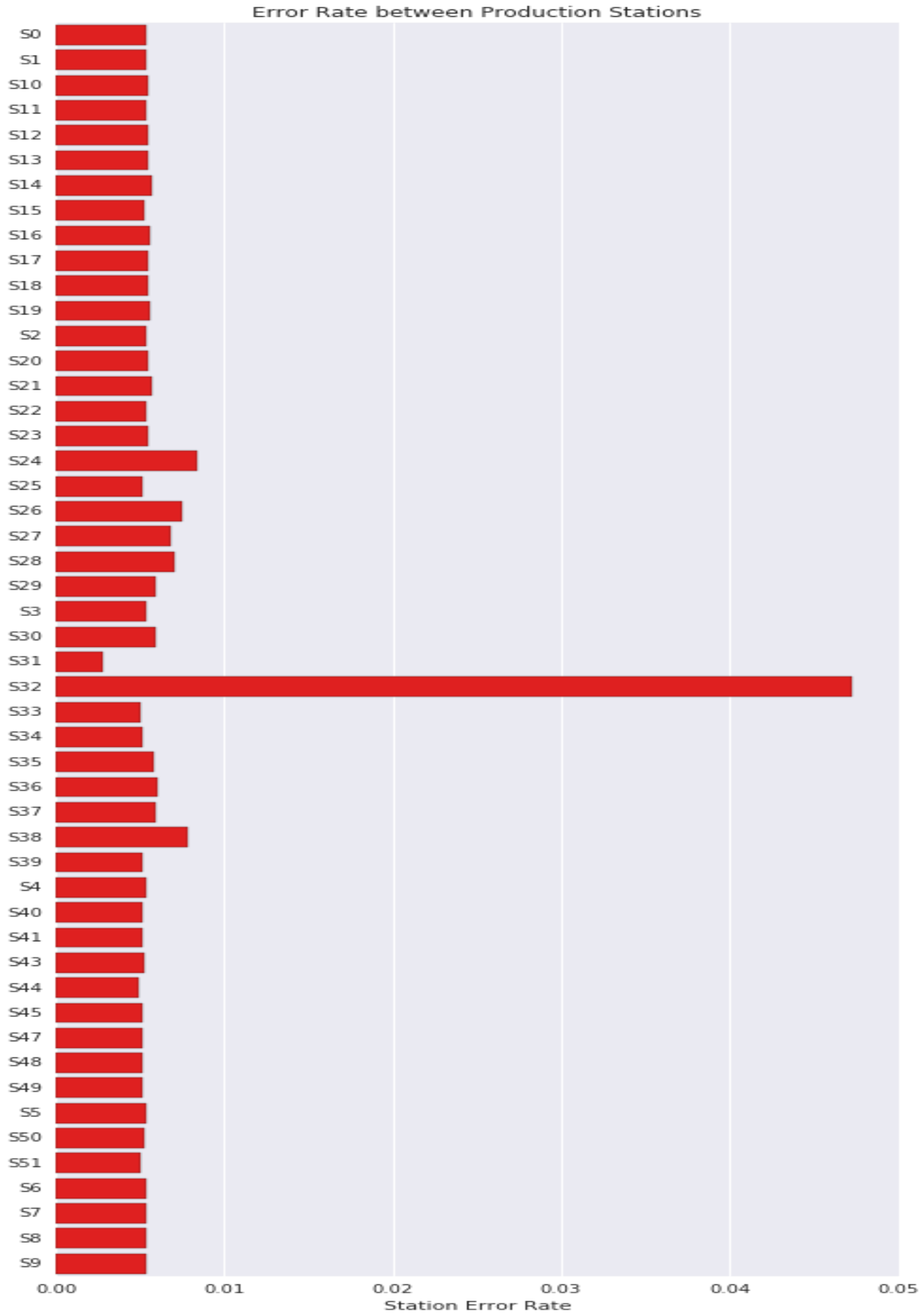
label	type	Missing	Zeros	+Inf	-Inf	min	max	mean	sigma	cardinality	Actions
Id	int	0	0	0	0	1.0	2367494.0	1183446.2581	683439.5519	·	Convert to enum
L0_S0_F0	real	509244	0	0	0	-0.7160	0.2840	-0.0	0.0805	·	·
L0_S0_F2	real	509244	0	0	0	-0.6980	0.2950	-0.0001	0.0930	·	·
L0_S0_F4	real	509244	0	0	0	-0.4150	0.5850	0.0	0.2116	·	·
L0_S0_F6	real	509244	0	0	0	-0.4160	0.5840	0.0001	0.2117	·	·
L0_S0_F8	real	509244	0	0	0	-0.4470	0.5530	-0.0001	0.0944	·	·
L0_S0_F10	real	509244	0	0	0	-0.6120	0.2520	0.0001	0.1644	·	·
L0_S0_F12	real	509244	84126	0	0	-0.0520	0.3700	-0.0	0.0194	·	·
L0_S0_F14	real	509244	0	0	0	-0.2720	0.7280	-0.0002	0.1044	·	·
L0_S0_F16	real	509244	12142	0	0	-0.4590	0.5000	-0.0	0.1149	·	·
L0_S0_F18	real	509244	0	0	0	-0.4310	0.5590	-0.0	0.1127	·	·
L0_S0_F20	real	509244	874	0	0	-0.3410	0.6590	-0.0003	0.2029	·	·
L0_S0_F22	real	509244	874	0	0	-0.3410	0.6590	-0.0003	0.2029	·	·
L0_S1_F24	real	509224	0	0	0	-0.3710	0.5960	-0.0	0.1032	·	·
L0_S1_F28	real	509221	2616	0	0	-0.4080	0.5920	0.0002	0.1073	·	·
L0_S2_F32	real	844356	0	0	0	-0.2130	0.7870	-0.0001	0.0543	·	·
L0_S2_F36	real	844356	667	0	0	-0.3080	0.6920	-0.0004	0.1946	·	·
L0_S2_F40	real	844356	0	0	0	-0.1920	0.8080	-0.0002	0.0715	·	·
L0_S2_F44	real	844356	296	0	0	-0.3730	0.6270	-0.0006	0.2655	·	·
L0_S2_F48	real	844356	37990	0	0	-0.9840	0.0150	0.0	0.0099	·	·

Exploratory Visualization

Here are some visualizations of features:



From the histogram of variable importance, we found the important feature which has the scaled importance at the range 1 – 0.1.


Error Rate between Production Stations
S0
S1
S10
S11
S12
S13
S14
S15
S16
S17
S18
S19
S2
S20
S21
S22
S23
S24
S25
S26
S27
S28
S29
S3
S30
S31
S32
S33
S34
S35
S36
S37
S38
S39
S4
S40
S41
S43
S44
S45
S47
S48
S49
S5
S50
S51
S6
S7
S8
S9
0.00
0.01
0.02
0.03
0.04
0.05
Station Error Rate

Error rate between production stations shows us station 32 has 4.7% error rate compared to the mean error rate 0.6%. It only has one feature L3_S32_F3850, which has a high error rate. From the above two visualisations, we can see that only three stations impact the production line and other stations play the equal and low impact in the production.

Algorithms and Techniques

I plan to spot check a series of classification learning algorithms and pick that performs fairly well with parameters to tune further.

Before committing to a model for this data, I did an experiment with a series of classification models available in sklearn to see out of the box which one scored well with default parameters.

Each model are tested with 50,000 data of train_numeric, measured the accuracy of every classification model. From the measured score, the algorithm that performed above and beyond the other classifications was Extra Tree Classification. This is a process where the class implements a meta-estimator that fits a number of randomised decision trees on various subsamples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.

This algorithm has several hyper-parameters that I tuned.

n_estimators: This is the number of trees in the forest.

max_features: The number of features to consider when looking for the best split. The search of the split does not stop until at least one valid partition of the node samples is found, which controls the over-fitting of the model.

For initial training I just used a series of default parameters to see how the model did out of the box, then the tuning is done with respect to the performance of the model. Tuning max_depth, min_samples_split and min_samples_leaf parameters affect the model by under-fitting. Thus tuning n_estimators and max_features were well performed for prediction.

Based on the evaluation in data exploration and visualising the importance of the features, we get to see 46% of the features played the important role for prediction using extra tree classification model. Before using the data we have to fix the missing values of the dataset where the dataset has a numerical reading ranging from -1 to 1. For this dataset -9 would be the best missing value which will be helpful for prediction.

Benchmark

The primary benchmark here would be the accuracy of prediction. During our data exploration, our classifier was able to predict with baseline accuracy of 0.0926. Since we are using the data from kaggle, from analysing the leader board form 600 kagglers the benchmark score would be 0.1580, which would be correct accuracy score for predicting the value of data, we are hoping to get better performance from the model better than the baseline accuracy.

Methodology

Data Preprocessing

According to the data exploration, we have to deal with the missing values of the dataset in train_numerics where the values of the dataset are at the numerical range of -1 to 1. For this dataset -9 would be the best missing value which will be helpful for prediction.

▾ COLUMN SUMMARIES

label	type	Missing	Zeros	+Inf	-Inf	min	max	mean	sigma	cardinality	Actions
Id	int	0	0	0	0	1.0	2367494.0	1183446.2581	683439.5519	·	Convert to enum
L0_S0_F0	real	509244	0	0	0	-0.7160	0.2840	-0.0	0.0805	·	·
L0_S0_F2	real	509244	0	0	0	-0.6980	0.2950	-0.0001	0.0930	·	·
L0_S0_F4	real	509244	0	0	0	-0.4150	0.5850	0.0	0.2116	·	·
L0_S0_F6	real	509244	0	0	0	-0.4160	0.5840	0.0001	0.2117	·	·
L0_S0_F8	real	509244	0	0	0	-0.4470	0.5530	-0.0001	0.0944	·	·
L0_S0_F10	real	509244	0	0	0	-0.6120	0.2520	0.0001	0.1644	·	·
L0_S0_F12	real	509244	84126	0	0	-0.0520	0.3700	-0.0	0.0194	·	·
L0_S0_F14	real	509244	0	0	0	-0.2720	0.7280	-0.0002	0.1044	·	·
L0_S0_F16	real	509244	12142	0	0	-0.4590	0.5000	-0.0	0.1149	·	·
L0_S0_F18	real	509244	0	0	0	-0.4310	0.5590	-0.0	0.1127	·	·
L0_S0_F20	real	509244	874	0	0	-0.3410	0.6590	-0.0003	0.2029	·	·
L0_S0_F22	real	509244	874	0	0	-0.3410	0.6590	-0.0003	0.2029	·	·
L0_S1_F24	real	509224	0	0	0	-0.3710	0.5960	-0.0	0.1032	·	·
L0_S1_F28	real	509221	2616	0	0	-0.4080	0.5920	0.0002	0.1073	·	·
L0_S2_F32	real	844356	0	0	0	-0.2130	0.7870	-0.0001	0.0543	·	·
L0_S2_F36	real	844356	667	0	0	-0.3080	0.6920	-0.0004	0.1946	·	·
L0_S2_F40	real	844356	0	0	0	-0.1920	0.8080	-0.0002	0.0715	·	·
L0_S2_F44	real	844356	296	0	0	-0.3730	0.6270	-0.0006	0.2655	·	·
L0_S2_F48	real	844356	37990	0	0	-0.9840	0.0150	0.0	0.0099	·	·

Since this is for predicting the manufacturing defect of spare parts, each and every feature plays an important role for prediction. Even normalising the numerical value of the feature will not give any advantage for prediction.

Implementation

Before committing to a model for this data, I did an experiment with a series of classification models available in sklearn to see out of the box which one scored well with default parameters.

Each model are tested with 50,000 data of train_numeric. The accuracy of the models are:

Model	Accuracy
Random forest classifier	0.06
Support vector machine classifier	0
Gaussian naïve bayes classifier	0.0119
Ada boost classifier	0
Decision tree classifier	0.0599
Bagging classifier	0.0734
Knearest neighbours classifier	0.0808
Stochastic gradient descent classifier	0.0448
Extra tree classifier	0.0926

From the accuracy scores of the models, we can see knearest neighbours classifier and extra tree classifier performs well than other classification models. When tuning the knearest neighbours classifier the highest accuracy is 0.136 which is not a good performance improvement.

The next better performing classification model is extra tree classifier by tuning the model we achieved the highest performance accuracy of 0.1909 which is a better performance with respect to the benchmark accuracy.

The major problem faced during coding with imbalance positive and negative response data is while tuning the model, when there is slight difference in tuning the model gets over-fitted or under-fitted. Further explanation on tuning the model given in refinement.

Refinement

In order to find the hyperparameters for the Extratree Classification model with the best accuracy score, I tuned the n_estimators, max_features, max_depth, min_samples_split and min_samples_leaf, the scored accuracy are:

Parameters	Training Accuracy	Test Accuracy
n_estimators = 500, max_features = 250, max_depth = 5	0.0893	0.06
n_estimators = 500, max_features = 450, max_depth = 5	0.1522	0.0985
n_estimators = 500, max_features = 250, max_depth = 10, min_samples_split = 3 min_sample_leaf = 2	0.34	0.1123
n_estimators = 518, max_features = 450	0.9447	0.1909
n_estimators = 500, max_features = 250	0.9474	0.1766

From these parameters tuning at n_estimators = 518 and max_features = 450 the model gets the best accuracy score of 0.1906. Even though the accuracy score obtained is not near 1, because the dataset has 99.4% positive response and only 0.6% negative response, so 0.1906 is the best accuracy score.

Results

Model Evaluation and Validation

Compared the performance of two models:

1. KNearest Neighbours Classification: The most efficient model, very fast, but the performance is not good as expected. The Extratree Classification model is better than the Knearest neighbours classification model, the best performance of the knearest neighbour classifier is 0.136.
2. Extratree Classification: The model is slower than the knearest neighbours, but the accuracy of the model is 0.1906 which is higher than knearest neighbors.

Extratree classifier implements a meta estimator that fits a number of randomized decision trees on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. Knearest neighbours classifier does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbors of each point, a query point is assigned the data class which has the most representatives within the nearest neighbors of the point. Since Extratree Classifier takes the meta estimator form the randomized tree, where knearest neighbours classifier just vote the majority of nearest neighbors instead of predicting the value from dataset, that makes Extratree classifier the best model for prediction.

Compared to Benchmark:

To demonstrate the result of a final model at the parameter setting n_estimators = 518 and max_features = 450, I consider running the code at 3 different sizes of the dataset.

Training data size	Test data size	Accuracy score
70000	30000	0.1796
350000	150000	0.1684
560000	240000	0.1909

We can see that as the model as the model gets trained with a large dataset, the benchmark performance score of the model gets increased at the selected parameter of the model.

Justification

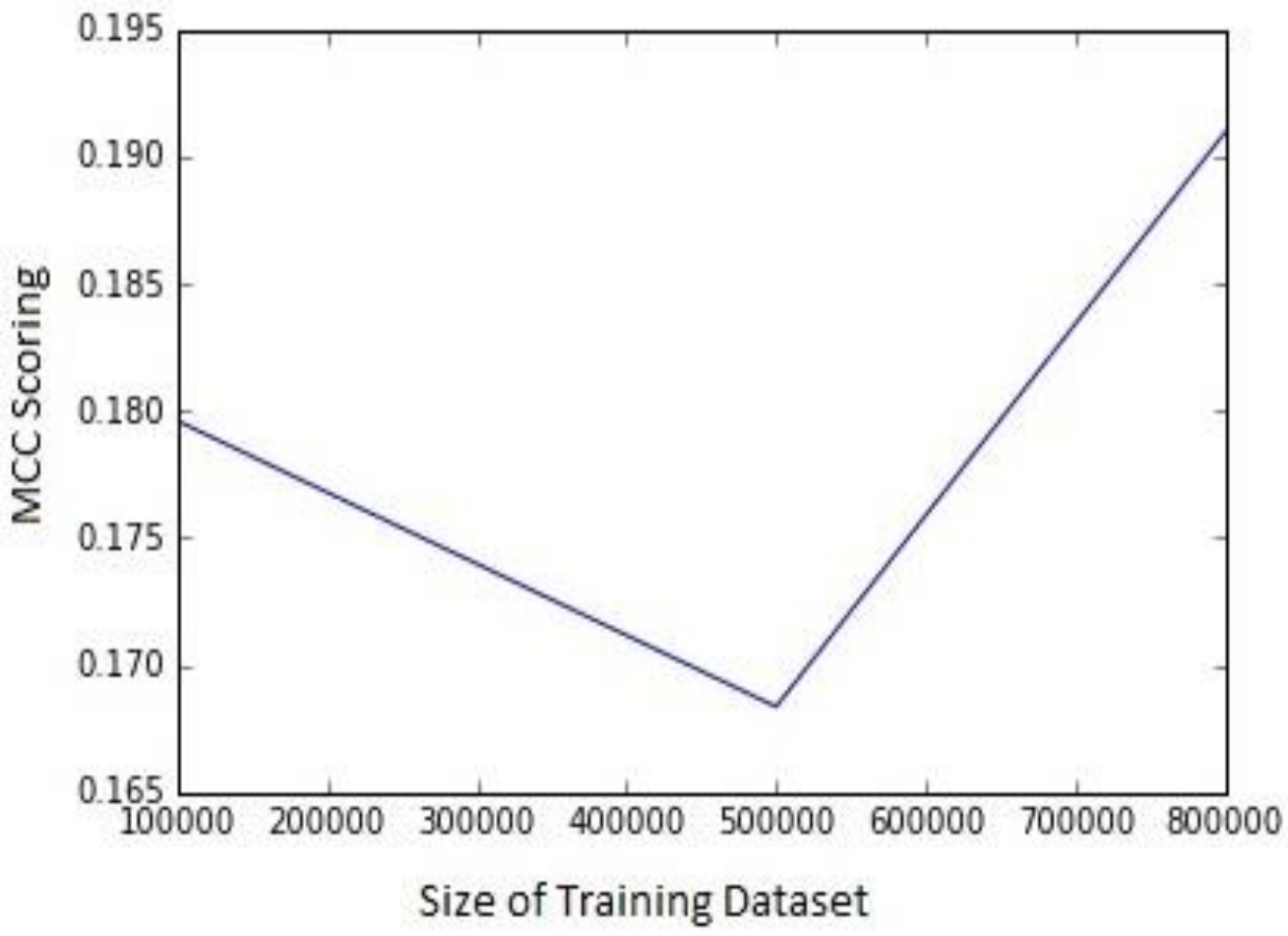
Finally, I decided to use Extratree Classification model to do the prediction, the result of the test set is 0.1906 which is higher than the accuracy we got in benchmark score (0.1580).

But the accuracy got by just testing on 240000 test data set, which is not large enough, so the reliability of the accuracy should be doubted. But still we have reason to justify that the Extratree classification model works better than the baseline accuracy and since the dataset has 99.4% positive response and only 0.6% negative response, this indicates to me that based on the constraints the Extratree classification model is sufficient to predict the manufacturing failure.

Conclusion

Free-Form Visualization

The Extratree Classification model is trained by three different size of training sets and tested using the three different size testing set, which are then evaluated using Matthews Correlation Coefficient (MCC) between the predicted and the observed response.



When the classifier is trained by 70,000 data and tested using 30,000 data the MCC score is 0.1789 which seems to be better performing which compared baseline score the model makes good performance improvement.

Now the model is trained by 350,000 data and tested using 150,000 data the MCC score is 0.1684 which is not a good improvement with respect to the 70,000 test data.

Finally when the model is trained without changing the parameters by 560,000 data and tested by 240,000 data now the MCC score make a sudden jump to 0.1909.

Even though the model accuracy score falls when trained by 350,000 data and increase when trained by 560,000 data, where there may be any missing details in 350,000 data for the model to perform better. Thus when the model is trained further the model's performance may increase.

Reflection

This project is to reduce manufacturing failures, we are given 1183000 Bosh manufacturing dataset which has the numerical record of the tools at each station. Most of the time is spent on understanding the data and handle the missing values, where the numerical values are in the range -1 to 1, where the missing value is handled by replacing with -9 which helped in model prediction.

Another interesting thing of the dataset is the importance of features for prediction. As mentioned in analysis few station has high error rate particularly station 32 has 4.7% error rate by viewing 25,000 datasets but as we increase the analysis of the dataset the error rate keeps changing with the station, this analyzation shows the equal importance of features. Thus training the model with 46% of the features would help in predicting the manufacturing failure.

Based on the baseline score Extratree Classification model performance best in baseline without tuning the parameters scoring 0.0926. The tuning parameters for extratree classification are n_estimators, max_features, max_depth, min_samples_split and min_samples_leaf, where max_features helps in controls over-fitting the model. Tuning the max_features with 150, 250 and 450, when max_features at 150 and 250 the model performance score is not at big improvement from baseline score, but when max_features is at 450 the performance of the model is notably increasing. While introducing max_depth, min_samples_split and min_samples_leaf the model gets under-fitting. Thus for final prediction the parameters n_estimators and max_features are set to 518 and 450, which make the scoring of 0.1909 by training with 560,000 data.

Improvement

The first and most obvious way to improve this model is to start taking in more data. I only made use of Bosh manufacturing reading of each station provided in kaggle, by training the model with large dataset may improve prediction model. Some cleaning would have to be done by normalising or feature selecting by looking the large data.

It's also possible that Extratree classification model isn't the right way for the dataset and in reality, the metrics of using accuracy may not be a good idea in practice, we should consider assigning additional weight to negative instances, for example, F_1 score.