Capstone Project:

Bosch Production Line Performance

Machine Learning Engineer Nanodegree

Joseph Durago

09/19/2016

I. Definition

Project Overview

Bosch, a company which manufactures automotive components, industrial products and building products, shared production line data with Kaggle. The data consists of measurements from 4 production lines and 50 stations. The measurements follow a part as it goes the the production line and is a mixture of numerical, categorical, and time data. Bosch and Kaggle have started a data science competition in hopes to predict whether a part will fail or pass Bosch's quality control.

Problem Statement

The goal of this project is to accurately predict if a part will pass or fail Bosch's quality control using the training and test data provided by Bosch. I will be implementing an XGBoost (Extreme Gradient Boosting) algorithm, which is a supervised learning technique based on decision trees. The XGBoost algorithm was utilized as it has been used in previous winning Kaggle competitions. Also, XGBoost has comparatively good runtime performance versus other machine learning algorithms. This is important since

the data set is quite large. To implement the XGBoost algorithm I will need to do the following:

1. Identify machine that can process the large dataset
2. Explore the dataset to determine important and unimportant features
3. Implement XGBoost algorithm based on important features
4. Tune algorithm to maximise correct predictions of passing/failing parts

Metrics

Submissions into the Bosch Production Line Performance competition are evaluated on the 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.

MCC returns a value between -1 and +1. A coefficient of +1 represents a perfect prediction, 0 no better than random prediction and −1 indicates total disagreement between prediction and observation. [1] MCC is a good metric in assessing the accuracy of the production line data as it takes into account true positives, true negatives, false positives and false negatives in determining model performance.

II. Analysis

Data Exploration

The dataset consists of 7 files and is a mix of categorical, numerical, and date data:

Filename	# Rows	# Columns
test_categorical.csv	1183749	2141
train_categorical.csv	1183748	2141
test_date.csv	1183749	1157

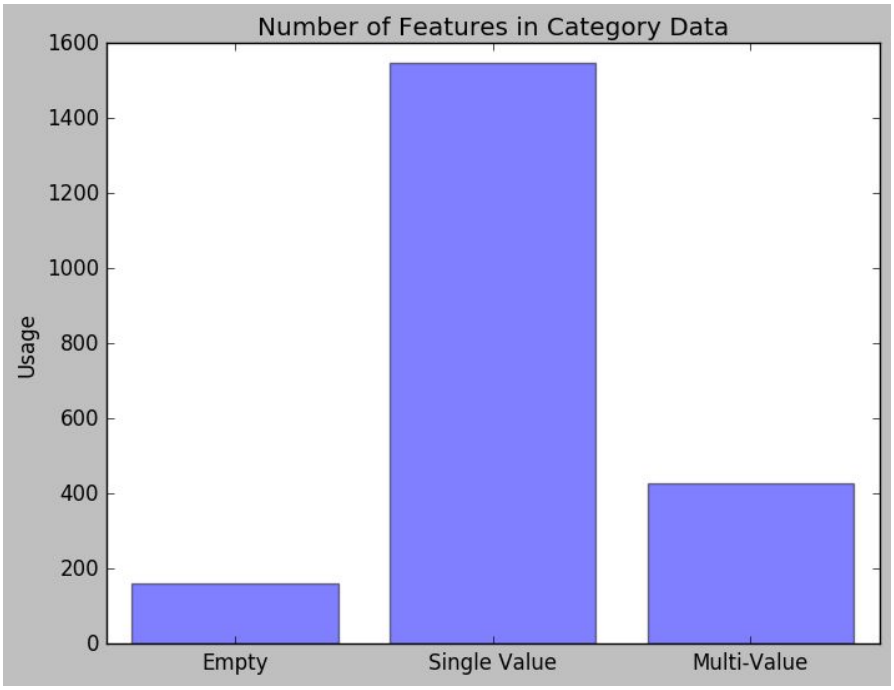
train_date.csv	1183748	1157
test_numeric.csv	1183749	969
train_numeric.csv	1183748	970
sample_submission.csv	1183749	2

https://www.kaggle.com/c/bosch-production-line-performance/data

The dataset is 17 GB. Working with a large dataset is a challenge, especially one with a sizeable number of columns. I'll be using a r3.xlarge instance on Amazon Web Services, which has 30.5GB of memory and 160GB of SSD storage to aid in data processing.

The dataset contains an extremely large number of features that have been anonymized (likely for intellectual property reasons). Features are named according to a convention that tells 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 is feature number 3939. 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.[2]

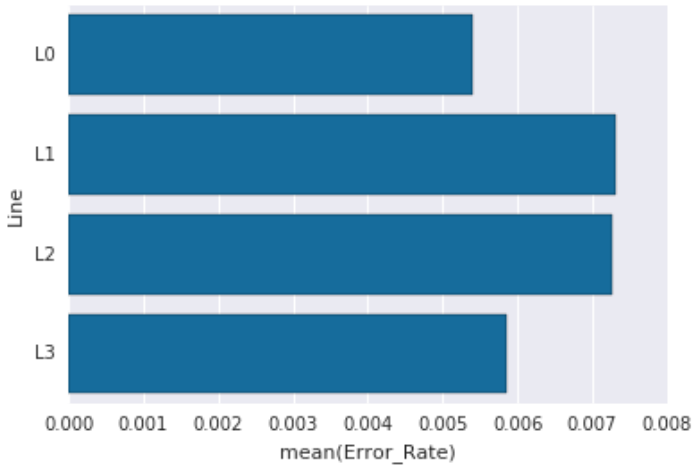
The categorical data set is primarily composed of single value data points. See chart below showing the distribution between no entries, single value entries, and multi-value entries.



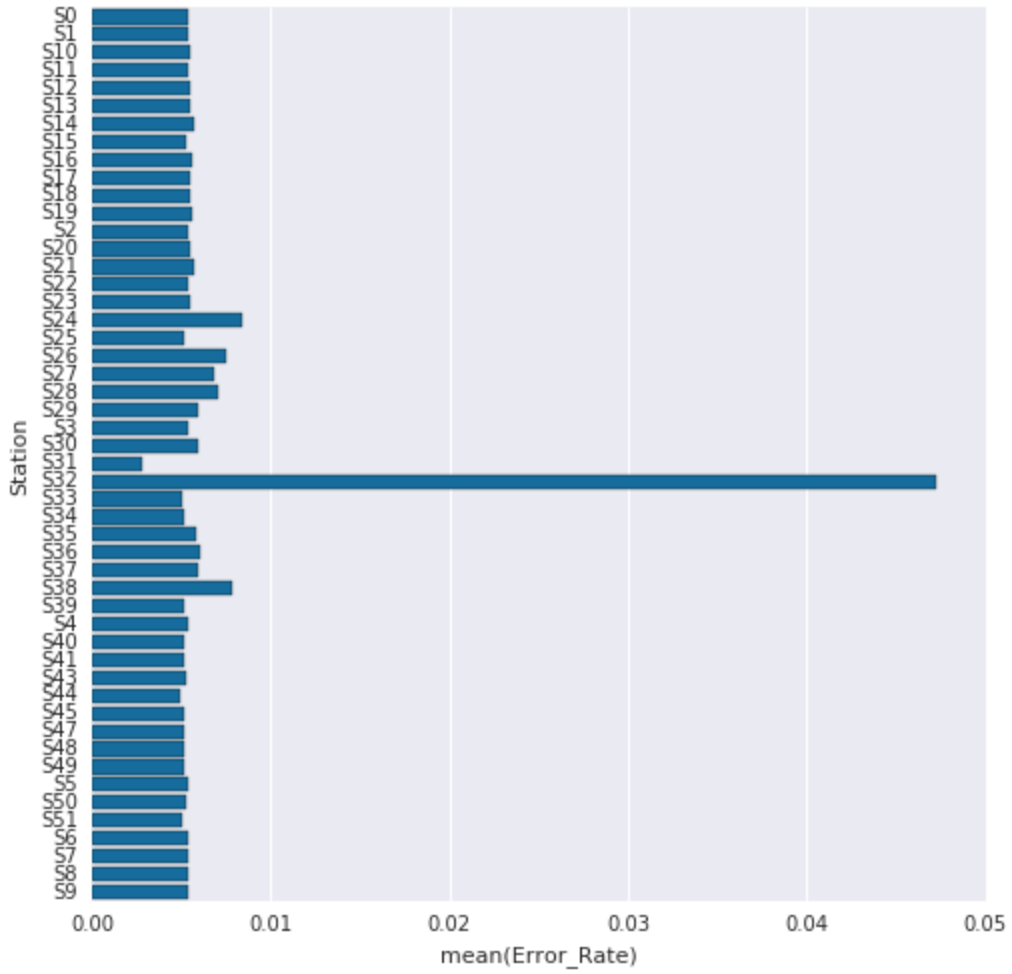
XGBoost requires categorical data to be encoded for proper classification. However, with such a large data set, one hot encoding all of the categorical data from Bosch will make the data set very large and difficult to compute on. To save space, I plan on using an encoding technique called, leave-one-out encoding [3].

Exploratory Visualization

The chart below compares the error rate of each of the 4 production lines. The error rates between the production lines are relatively similar.



The chart below compares the error rates of the various stations. Station 32 has a much higher error rate compared to the other stations. I would speculate that the station could be a rework station where faulty components go to be refinished. Given the high error rate associated with this station, I would expect the XGBoost algorithm to incorporate it in its model for predicting failed components.



Algorithms and Techniques

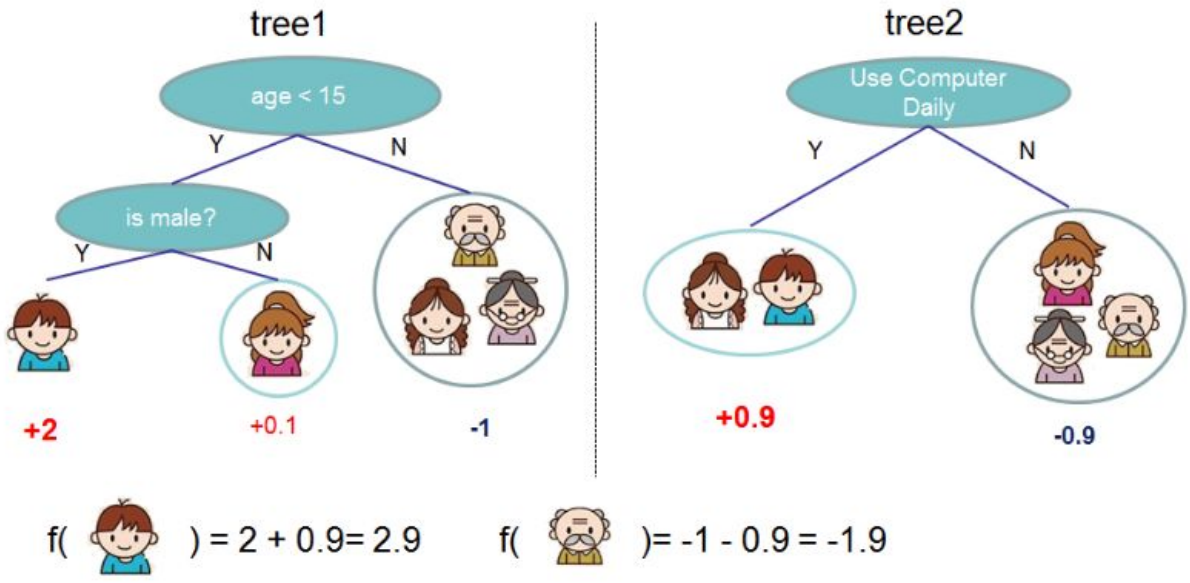
An XGBoost algorithm will be used in developing a model from the training data. The algorithm is based off of a decision tree. There are various decision trees algorithms and each apply different metrics to form the tree. For example the ID3 algorithm attempts to minimize entropy, or the amount of uncertainty in a data set, while maximizing information gain. XGBoost is based on an ensemble of trees and its objective is to minimize training loss and place higher value on simpler models.

$$Obj = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

Training loss Complexity of the Trees

[4]

XGBoost is an ensemble of trees. During training, a new tree is added with each prediction. Each of the trees attempts to optimize the objective function above. Given an input a prediction is made by taking the weights of each of the trees in the ensemble. Individually each tree is a weak classifier. However, the ensemble of trees together create a strong classifier.



The benefits of using an XGBoost algorithm is that it is invariant to inputs so normalization is not needed, high order interactions can be learned between features, its scalable, and is used widely in industry. Together, these benefits make XGBoost an ideal candidate to use for the Bosch production line data set. [4]

Benchmark

As a benchmark, I plan on implementing the XGBoost algorithm with the default parameters. The parameter values I plan on manipulating when tuning the algorithm are listed below [5]:

learning rate [default=0.3, alias: learning_rate]

- step size shrinkage used in update to prevents overfitting. After each boosting step, we can directly get the weights of new features. and eta actually shrinks the feature weights to make the boosting process more conservative.

max_depth [default=6]

- maximum depth of a tree, increase this value will make the model more complex / likely to be overfitting.
- range: [1,∞]

subsample [default=1]

- subsample ratio of the training instance. Setting it to 0.5 means that XGBoost randomly collected half of the data instances to grow trees and this will prevent overfitting.
- range: (0,1]

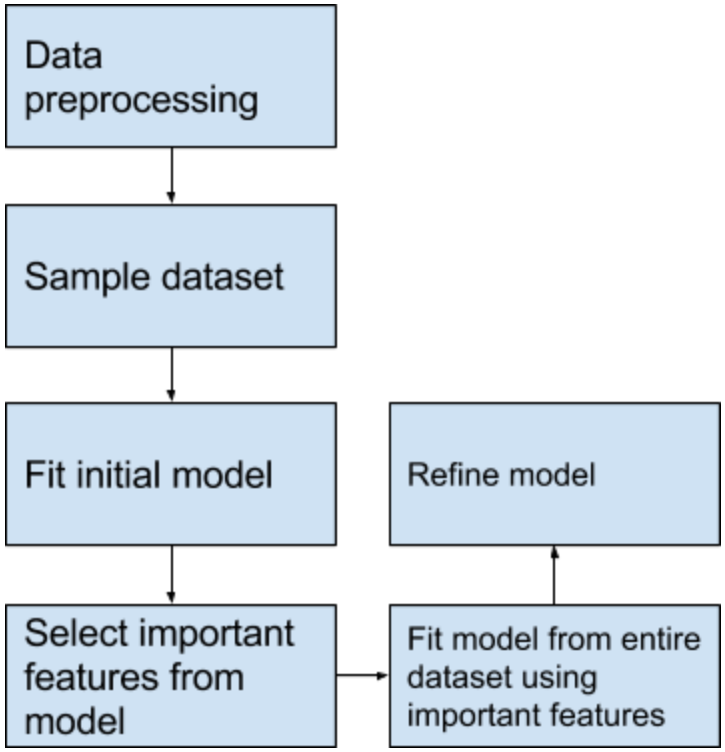
colsample_bytree [default=1]

- subsample ratio of columns when constructing each tree.
- range: (0,1]
-

With these default parameters I achieve an MCC score of 0.20118 and I would have been on the 891st place on the leaderboard. My goal is to tune the XGBoost algorithm to perform better than the benchmark.

III. Methodology

The flowchart below shows the methodology used in developing the XGBoost algorithm on the Bosch production line dataset and will be discussed further in subsequent sections.:



Data Preprocessing

The baseline algorithm with default parameters discussed in the Benchmark section provides an MCC score of 0.209. Incorporating all of the data and encoding all the categorical data would require lots of processing power. To reduce the amount of

processing power needed, I filtered out insignificant features and used a leave-one-out encoding technique. I had also processed the date data into weekly and daily bins. This way the XGBoost algorithm could potentially find pass/fail patterns that could occur on a particular day of the week or on a particular time of day.

Implementation

After the data was preprocessed, I sampled 5% of the data and fit the XGBoost classifier to it. Sampling the data was used so that I could quickly identify important features. The full dataset was then filtered to include only the previously identified important features and a XGBoost classifier was then fitted to the new data set. Sampling the data allowed for a reduced amount of processing power. I had attempted to fit the entire data set and not sample the data. This was unmanageable, took lots of processing time and I used up all the available 244 GB of memory on my r3.8xlarge machine [6]

Refinement

After preprocessing the data and training the XGBoost algorithm. I identified the important features that should be used in creating a model. Initially I trained on all the data, however I found that none of the categorical features were important, when using the leave-one-out encoding technique. Also incorporating the processed date data, which binned the data into a weekly and daily format didn't provide a more accurate prediction. To save processing time I did not use any of the categorical data on subsequent rounds of training and I relied on the original raw date data instead of the pre-processed binned date data.

IV. Results

Model Evaluation and Validation

I tuned the XGBoost algorithm using a grid search approach. The parameters I tuned with the various values were:

learning_rate: [0.05, 0.1, 0.3]
max_depth: [6, 9, 12]
subsample: [0.9, 1.0]
colsample_bytree: [0.9, 1.0]

The final model had the final values:

Learning_rate = 0.05
Max_depth = 9
Subsample = 0.9
Colsample_bytree = 0.9

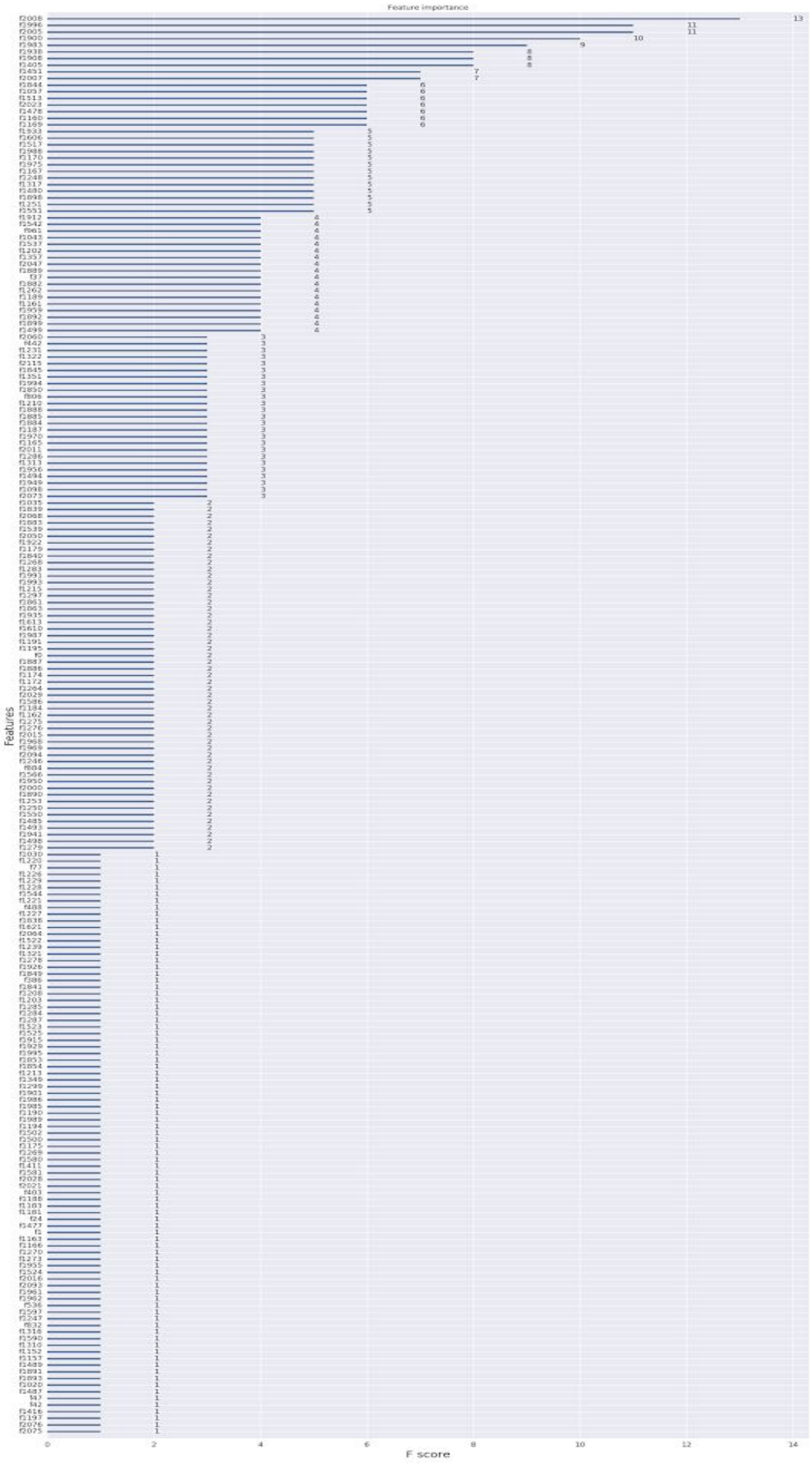
I achieved a MCC score of 0.21353 and I would have been on the 849th place on the leaderboard. This is an improvement of 42 places on the leaderboard from the benchmark model.

Justification

Compared to the MCC benchmark score of 0.209, I had achieved a score of 0.21353 after tuning the algorithm. This was a marginal improvement from the benchmark. I think the model I had developed is a good starting point, however there is much improvement to be made before my machine learning model should be deployed in a production setting.

V. Conclusion

Free-Form Visualization


Feature importance
Features
F score
0
2
4
6
8
10
12
14

The chart above shows the F-score for important features identified by the XGBoost algorithm. Overall the features have relatively low f-scores. The highest value being 13 and the majority of features having f-scores below a value of 3. The shows that a single feature does not strongly correlate with a passing or failing part. Instead, a combination of these low scoring features contribute to predicting whether a part passes or fails.

Reflection

The Bosch production line data set was large. It was interesting working with a large dataset that was a mixture of numerical, date, and categorical data. It provided several interesting challenges that I had not encountered before. I had initially attempted to fit the XGBoost algorithm on my local laptop, but quickly realized it did not have enough processing power. I then used more capable machines on AWS, but even with the added processing power I still needed to find creative solutions to working with the large amount of data. After refining the algorithm I was able to achieve a reasonable MCC score. However, there is lots of improvement to be made before the machine learning system I had developed in predicting a passing/failing part would be used in a production environment.

Improvement

Further improvements that can be made are:

- Determine alternative ways of including categorical data
- Add parallel processing of data to increase speed of fitting XGBoost algorithm
- Use different learning algorithms for categorical, date and numerical data

I attempted to incorporate the categorical data and found that it wasn't an important feature in determining if a part would pass or fail in the production line. I think there are

other methods that can be used to add the categorical data that can be explored when attempting to improve upon the program I had developed.

Adding parallel processing of the data across multiple machines would improve processing time. It would also allow me to use all the training data, instead of taking a sample of it, thereby improving accuracy of the algorithm.

An additional approach I could use to predicting a passing/failing part would be to use 3 separate learning algorithms for the numerical, date and categorical data. Each of these algorithms could produce its own prediction. The final response would be a weighted calculation of each of the predictions. This method would allow for all the data to be incorporated and would allow for a more tuned approach to looking at each of the data types, instead of using a single monolithic approach to the entire dataset.

Appendix I.

References

1.https://en.wikipedia.org/wiki/Matthews_correlation_coefficient

2.https://www.kaggle.com/c/bosch-production-line-performance/data

3. http://www.slideshare.net/OwenZhang2/tips-for-data-science-competitions

4.http://homes.cs.washington.edu/~tqchen/pdf/BoostedTree.pdf

5. http://xgboost.readthedocs.io/en/latest/python/python_api.html

6. https://aws.amazon.com/ec2/instance-types/

7.https://www.kaggle.com/joconnor/bosch-production-line-performance/python-xgboost-starter-0-209-public-mcc

8.http://www.slideshare.net/ShangxuanZhang/kaggle-winning-solution-xgboost-algorithm-let-us-learn-from-its-author