Predicting Rare Failure Events using Classification Trees on Large Scale Manufacturing Data with Complex Interactions

Jeff Hebert, Texas Instruments
Maine Fab, Surface Prep Process Engineering
South Portland, ME
Jeffrey.Hebert@ti.com

***Abstract*—Manufacturers collect large amounts of information during production. This data can be hard to interpret because of the presence of many unknown interactions. The goal of this study is to demonstrate how machine learning methods can uncover important relationships between these parameters. This paper describes several techniques to predict rare failure events. Single parameter methods are used to explore performance over time. Tree based classification methods are used to identify deep interactions that cannot be identified by linear methods. Tree methods are shown to identify non-linear relationships.**

I. Introduction

This paper describes several techniques to identify relationships related to failing parts in The Bosch Production Line Performance challenge hosted by Kaggle [1]. Section I gives a brief introduction to the data and methods used in this paper. Section II introduces a method to visually examine yield patterns. Section III introduces a method to identify individual parameters that correlate with product quality. Section IV describes the Random Forest classification model [2]. Section V describes Boosted Classification Trees [3] and the XGBoost model [4]. Section VI describes cross validation as a method to improve model performance. Section VII discusses some of the implications of these methods. Section VIII concludes with findings.

Bosch provided anonymized data with no information on the kind of factory that produced the data. The competition challenge was to accurately predict which parts will fail at the end of the production line. The data is separated into three different types: Numeric data, categorical data, and time data. Only the training data is considered in this analysis. Training data has labels that indicate the outcome of each row. Testing data is used in the Kaggle competition to evaluate model performance. The testing data has no outcome so it is not used in this work.

The analysis in this paper was performed using the R programming language [5]. Code for the visualizations used in this paper are posted in the Bosch Production Line Performance Kaggle forums.

There are 1,183,747 rows (observations) in each training data file. There are 4,264 variables in the dataset: 968 numeric variables, 2140 categorical variables, and 1,156 time variables. The variables are described as scaled and anonymized. The data is sparse. In each row, more than half of the variables are missing. The variable names have a consistent structure of L#_S#_*. The last characters in the variable name increment sequentially.

The Bosch data has extreme class imbalance. The average failure rate is just 0.58%. There were 6,879 failing parts produced in this data. The overall failure rate is calculated by dividing 6,879 failing parts by the total of 1,183,747 parts produced. This means there are 172 passing parts for each failing part. This poses additional challenges to classification. Classification algorithms can simply predict the majority case to achieve high accuracy. With imbalanced data, performance can be improved by penalizing false positives [6]. There are many metrics that are highly sensitive to classification errors [7]. The Kaggle competition uses the Matthews correlation coefficient (MCC) which penalizes false positives.

It is necessary to make some assumptions about the data. First and foremost, missing data is not missing by accident. Missing data indicates the part was not processed at that station. Variables are presented in groups with prefixes like L0_S14. The second assumption is that all variables with the same prefix are associated with the same station. For example, there are 36 variables with the prefix L0_S14_. The variable L0_S14_F358 is a numeric variable for the station, L0_S14_F359 is a categorical variable for the station, and L0_S14_F360 is a date variable for the station. Third, each row represents an independent part such that parts processed together at one station may be processed separately at other stations. Finally, all timestamp data is normalized to integer values. The last assumption it that the time scale is linear such that any gaps represent actual elapsed time when a station is not active.

II. Process Yield and Stability

One method to identify production problems is to calculate average failure rate for each station. For example, Station 14 processed 120,625 parts with 681 parts failing. This give an average failure rate of 0.56%. This is slightly lower than the overall process failure rate of 0.58%. A quality engineer would perform a full analysis of variance to

978-1-4673-9005-7/16/$31.00 ©2016 IEEE

determine if the 0.02% yield difference is significant. This analysis is performed on each station to determine if failures are associated with a specific tool. However, this failure rate analysis does not determine if the tool is stable over time.

Station failure trend is a simple way to identify problems that appear temporarily. Figure 1 shows station activity along the X axis and the part outcome on the Y axis. Failing parts are marked in red. The station in Figure 1 shows many gaps in production. This station appears to run many parts in batches, and then has short breaks. The failure rate for this station appears to mimic the total run rate.

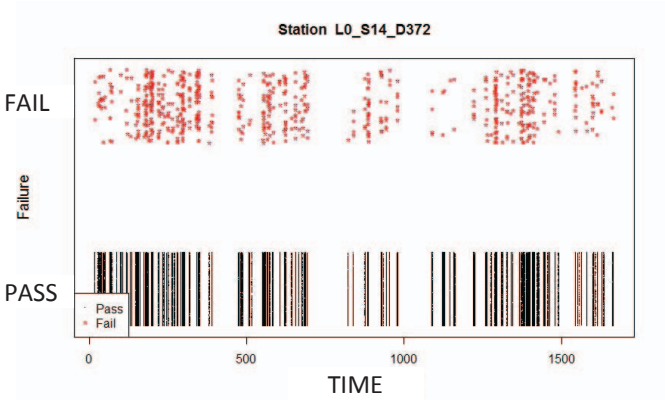

Figure 1 – Station Failure Trend. Each point represents a part that was processed at the station. Black points indicate the part passed at the end of production. Red points indicate that the part failed at end of production. vertical bars show that many lots are processed at the same time with white space indicating the station is idle.

III. Station Performance

Individual parameters provide insight on station performance over time. The numeric variables may represent tool settings, or product measurements. These variables are likely to influence product quality. Increasing variation or shifts in the process average are common manufacturing problems.

A time plot of tool performance can be used to identify periods of instability. Figure 2 shows the performance of parameter L0_S5_F114. It was relatively consistent prior to time 700. After a period of station inactivity from time 700 to 800, the parameter shifted to a lower value. After another period of inactivity from time 1000 to 1100, the parameter became unstable. These changes in behavior may indicate a change in station performance.

The overall station failure rate does not help when comparing a station against itself. Tool performance trend is one method to determine if station performance stable over time. This analysis is performed by cutting the data into equal time slices and calculate failure rate for each slice. Figure 2 shows an example of station yield changing when the tool parameter changes. Between time 200 and time 700, the tool had a relatively high failure rate with stable tool performance. After time 1100, the failure rate started low, however the failure rate increased with increasing variation in the L0_S5_F114 parameter.

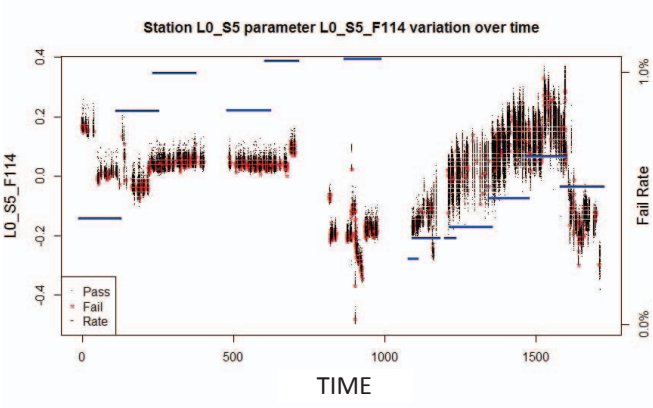

Figure 2 – Failure Rate and Parameter Trend. Each point represents the value of parameter L0_S5_F114 when the part was processed at station L0_S5. The relative density of black (passing) and red (failing) parts is indicated as yield (blue lines). Increasing variation after time 1000 correlate with increasing failure rate.

IV. Random Forest Classification

Random Forest classification is a powerful tool for uncovering interactions between variables [8][9]. This tool is a collection of decision trees that are each very simple. Each tree describes a small part of the process variation. When hundreds or thousands of these decision trees are used together, they can detect deep interactions. This is specifically useful to manufacturing where difficult problems may be the result of interactions between several stations.

Random Forest classification use random subsets of the independent variables to reduce the risk of over-fitting. Two important tuning parameters are the number of variables to select for each tree and the number of trees to construct. For classification problems, we use the square root of the number of features to use when constructing each tree. In the Bosch numeric data, there were 968 variables so we set the algorithm to select 32 variables to construct each tree. We constructed 500 trees for the final model. This ensures there are enough trees to use each variable multiple times.

Variable importance in Random Forest models can be calculated using the Gini Index [10]. Table I shows important variables identified by the Random Forest model. The Gini Index measures impurity between different classes in a population. Lower GINI index indicates better separation of the classes. At each split in the table, the average GINI index of the children is slightly lower than the GINI index of the parent. Variable importance is calculated by adding up the average decrease in GINI at each decision point where the variable is used in all the trees. Parameters with higher mean decrease in Gini index are more important than other variables. To calculate importance of interactions, we calculate the change in GINI explained by two or more connected nodes. We can calculate feature importance by

calculating the importance of each node (split), and interaction importance as the importance of each edge (path).

TABLE I. RANDOM FOREST VARIABLE IMPORTANCE

Parmeter	MeanDecreaseGini
L0_S2_F64	0.250
L3_S30_F3774	0.200
L3_S29_F3342	0.167
L1_S24_F1514	0.164
L3_S30_F3499	0.160
L1_S24_F1728	0.159
L3_S29_F3401	0.152
L1_S24_F1494	0.151
L3_S30_F3784	0.149
L3_S29_F3345	0.147

The parameters with high Mean Decrease in GINI may not have direct influence over failure rate. Figure 3 shows that parameter L0_S2_F64 has no apparent relationship with failure rate. However, the model indicates that L0_S2_F64 explains much of the variation in the data, so it most likely interacts with other parameters. Other methods are required to investigate deeper relationships.

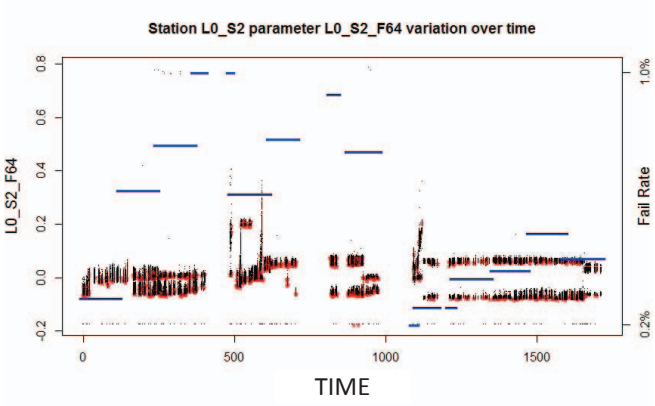

Figure 3 – Failure Rate and Parameter Trend for L0_S2_F64 which was identified as an important variable by Random Forest. On its own, this parameter does not provide useful information about passing or failing parts.

Random Forest classification can identify non-linear interactions. Figure 4 shows one decision tree in the forest. It shows that variable L0_S2_F64 interacts with several other variables to predict whether a part is passing or failing. Keep in mind that this is just one of many trees. There may be hundreds or thousands of branches where any given parameter determines the outcome. Each tree identifies a combination of parameter interactions that predicts part of the classification.

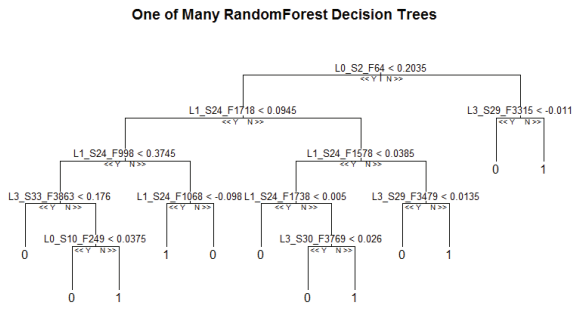

Figure 4 – One of the decision trees in the Random Forest model. This tree shows interactions between many variables.

V. BOOSTED CLASSIFICATION TREES

There are many variants of Boosted Classification Trees[9][11]. Boosted Classification Trees are similar to Random Forest classifiers in that they are constructed with many decision trees. However, the algorithm for generating new trees is different. Both methods use random samples of variables to construct each tree. Boosted Classification Trees adjust the weight of misclassified observations specifically to reduce model error remaining after previous trees are evaluated [4]. Boosted classification trees can also select a subset of the observations used for constructing each tree. Finally, Boosted Classification Trees typically have shallower trees than Random Forest classifiers.

Boosted Classification Trees have multiple tuning parameters which help to prevent over-fitting. Tuning parameters include the number of variables selected for each tree, the number of observations selected for each tree, the learning rate used to adjust the weight of misclassified observations, the maximum tree depth, and the number of boosting rounds among other parameters. In this analysis, we selected 80% of the variables and 80% of the observations to construct each tree. We used a maximum depth of 4, a learning rate of 0.1 and 300 boosting rounds.

XGBoost [3][4] is an implementation of Boosted Classification Trees commonly used on Kaggle. It is flexible and efficient. XGBoost calculates importance using increase in accuracy. We Table II shows several importance measures calculated on the full set of numeric observations. Parameters with higher Accuracy Gain are more important than other variables. Cover describes the fraction of observations that are evaluated with that parameter at all the decision points. This table indicates that the most important parameters are used to explain variation in less than 10% of the observations.

Figure 5 shows some interactions between variables identified by XGBoost. These two-way interactions help to explain conditions that lead to some failures. However, there are many failures that have the same conditions as passing parts. Just like Random Forest classifiers, XGBoost requires many trees to identify underlying relationships.

TABLE II. XGBOOST VARIABLE IMPORTANCE

Parmeter	Accuracy Gain	Cover
L3_S32_F3850	0.0494	0.0930
L3_S33_F3865	0.0335	0.0156
L1_S24_F1723	0.0189	0.0883
L3_S33_F3857	0.0189	0.0040
L3_S29_F3407	0.0166	0.0231
L3_S33_F3859	0.0123	0.0115
L1_S24_F1846	0.0116	0.0811
L3_S33_F3855	0.0116	0.0007
L3_S30_F3809	0.0112	0.0018
L3_S30_F3754	0.0111	0.0017

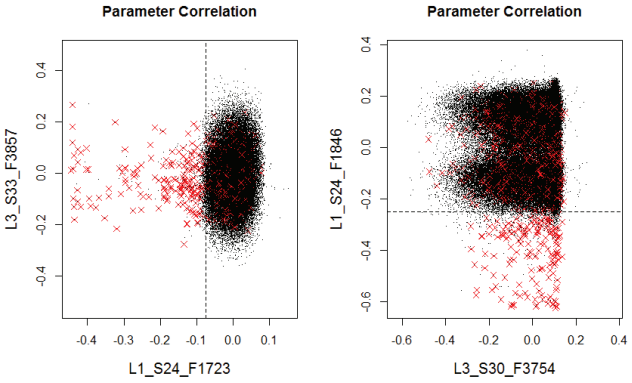

Figure 5 – Interactions between numeric variables identified by XGBoost classification. Each point represents an interaction between two variables. Failing parts are colored red. Dashed lines indicate separation between high failure conditions and low failure conditions.

VI. MODEL VALIDATION

These machine learning methods are very powerful. They can easily over-fit data. One method to minimize over-fitting is to validate the model with hold out data. In machine learning, it is common to hold out a portion of the training data to use later for validating the model. After building a model using training data, the remaining data (the holdout set) is used to ensure the model produces similar results with new data. If the holdout data gives poor results, this indicates that the original model is over-fit to the data [7]. In this analysis, we used a holdout set of 100,000 observations. The remaining 1,083,747 observations were used to construct models.

Three models were built using the same 1,083,747 observations. Validation was performed using the same holdout set of 100,000 observations. The confusion tables in Table III compare the performance of each method. The logistic regression model properly identified 31 failing parts, however it falsely predicted 48,582 passing parts would be failing parts. Random Forest and XGBoost performed similarly on this data. Random Forest performed slightly better by properly classifying 38 failing parts and only misclassifying 3 passing parts. Each of the models misclassified most of the failing parts.

TABLE III. MODEL PERFORMANCE ON 100,000 ITEM HOLD OUT SET

Logistic Regression

	Pass	Fail
PASS	50,838	48,582
FAIL	549	31

Random Forest

	Pass	Fail
PASS	99,417	3
FAIL	542	38

XGBoost

	Pass	Fail
PASS	99,405	15
FAIL	557	23

VII. DISCUSSION

Manufacturing organizations like Bosch have been working with Big Data long before the term was coined. Manufacturers use proprietary tools to analyze their data. The methods developed slowly and often in isolation. This means that manufacturers are often unfamiliar with modern machine learning tools. In recent years, large manufacturing data sets have become available to a much larger population of Data Scientists. There are many machine learning tools that provide useful that provide insight into complex interactions that arise in manufacturing.

Tree based classifiers like Random Forest and Boosted Classification Tree models can detect relationships that are not readily detectable with linear regression techniques. For example, logistic regression is a standard tool for classifying failing parts. Logistic regression is well suited for finding primary relationships. However, to detect second order or third order interactions, the independent variable matrix quickly grows very large. Classification problems like this Bosch classification challenge require hundreds of thousands of observations to solve. Tools like stepwise regression can help to isolate the important variables, but they do not solve the underlying problem. Complex relationships are difficult to model with linear methods. The Tree-based classifiers detect complex relationships and calculates the importance of each input parameter.

The use of random subsets in tree based models helps to minimize the risk of over-fitting a model. Outliers tend to influence a small percentage of the decision trees. Patterns that show up in many trees are likely to describe real behavior of the system. These two phenomena are replicated in each subset used by the algorithm. The repeated use of subsets in these models reduces the risk of over-fitting while identifying true underlying relationships.

There are several methods to determine variable importance. The Random Forest algorithm used here calculates Mean Decrease in GINI to determine variable importance. The XGBoost algorithm calculates Accuracy Gain and Cover. It is important to recognize there are many metrics that can help identify important relationships. Also, note that the most important parameters in the XGBoost model are used to explain variation in less than 10% of the observations. That is, their cover is less than 0.1. This is a key observation that explains why analyzing parameters individually provides little insight. These tools identify important parameters and complex relationships.

Tree based classifiers have several disadvantages. It can also be difficult to interpret the model formula. Each tree is easy to understand. However, there are hundreds of trees in the forest. The combined effect of all the trees is difficult to comprehend. It can also be difficult to understand the interaction between parameters.

There are now many powerful methods available to the public. Modern personal computers have the power to explore and process large data sets. These tools can be readily applied to manufacturing data to provide new insights. The tools described in this paper do not replace existing techniques, but they can be used to accelerate learning. This is just a small taste of the powerful methods used by modern

There are many other tools that can be used to find hidden relationships. Algorithms such as k-nearest neighbor (KNN), support vector machines (SVM), and neural networks can also provide insights. Many models can be combined in sophisticated ensembles which work together to explain different aspects of the problem. These methods may over-fit training data. It is important to use tools like holding out validation data to ensure the model will produce valid results on new data. Modeling techniques improve every day. Manufacturing quality will continue to improve when high volume manufacturing meets modern machine learning

VIII Conclusion

Tree based classification methods find important relationships. Tree methods perform better than logistic regression on the Bosh data set due to the presence of complex interactions. R implementations of Random Forest and XGBoost properly identify conditions that lead to rare failure events in the Bosh data set. The models used here do not identify all conditions that lead to failures. These methods compliment but do not replace existing methods to identify yield problems.

References

[1] https://www.kaggle.com/c/bosch-production-line-performance

[2] A. Liaw and M. Wiener (2002). Classification and Regression by randomForest. R News 2(3), 18--22.

[3] J. Friedman, “Greedy function approximation: A gradient boosting machine.” Ann. Statist. 29 (2001), no. 5, 1189--1232. doi:10.1214/aos/1013203451.

[4] T. Chen and C. Guestrin. “XGBoost: A Scalable Tree Boosting System.” 2016 ArXiv e-prints.

[5] R Core Team (2016). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

[6] A. Owen, “Infinitely Imbalanced Logistic Regression” Journal of Machine Learning Research 8 (2007) 761-773

[7] Powers, David M W (2011). "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation" (PDF). Journal of Machine Learning Technologies. 2 (1): 37–63.

[8] T. Hastie, R. Tibshirani, J. Friedman, “The Elements of Statistical Learning,” Springer-Verlag New York, doi:10.1007/978-0-387-84858-7.

[9] M. Fernandez-Delgado, E. Cernadas, S. Barro, and D. Amorim, “Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?” Journal of Machine Learning Research vol. 15 (2014), pp. 3133-3181, October 2014.

[10] Gilles Louppe, Louis Wehenkel, Antonio Sutera and Pierre Geurts “Understanding variable importances in forests of randomized trees” Proceeding NIPS'13 Proceedings of the 26th International Conference on Neural Information Processing Systems Pages 431-439

[11] Tianqi Chen, Tong He and Michael Benesty (2016). xgboost: Extreme Gradient Boosting. R package version. 0.4-4. https://CRAN.R-project.org/package=xgboost

[12] Abhijit Dasgupta (2014). reprtree: Representative trees from ensembles. R package version 0.6.