Binary Classification of Imbalanced data from Bosch Production Line

P-1: Xi Yang Liang Dong Yeojin Kim

1 Introduction

The big data analysis is one of the challenging problems in modern manufacturing industries[1], since the target datasets generally have complex inheritances, which are large-scale, imbalanced, high-dimensional, with different types of features, and including numerous missing values. In this report, we utilized the dataset from Bosch, which is one of the world's leading manufacturing companies. We focused on processing imbalanced data with different types of features, exploring a best strategy to process a large dataset, and learning a framework to tackle with the data properties mentioned above. The rest of this report is organised as follows: Section 2 describes the brief characteristics of the dataset, Section 3 introduces some previous related works, Section 4 presents the framework we designed for the problem, Section 5 displays some experimental results and raises the analysis, and finally, in Section 6 we discuss contributions and limitations of our study.

2 Data

The data is from a Kaggle competition: Bosch Production Line Performance [2]. To monitor production of each part along the assemble line, Bosch collected measurements of produced parts during each step. They posted these measurements to Kaggle, and launched a contest to predict failure parts through data analysis of published measurements. This is a typical binary classification problem, with the failure parts labeled as 1 and normal parts labeled as 0. However, this problem is pretty challenging due to some characteristics of the dataset: a) large data size, b) complex and high dimension features, c) high missing value rate, d) highly imbalanced classes (the ratio of positive and negative, i.e. failure parts and normal parts is 1:171 (6879: 1176868) in training dataset).

The detailed data size shows in Table 1. The dataset is quite large with 1183747 training and 1183748 testing instances, and it contains 3 different feature types: numeric, categorical and date, with original feature dimension being 986, 1157, 2140, respectively. Since Kaggle, as a contest, does not provide the ground-truth for testing data, we only adopted training dataset in our experiment, and divided it into three parts: training, validation and testing.

Table 1: Description for Original Bosch Dataset

Training Data				Testing Data			
Data Types	Features	Instances	Size (GB)	Data Types	Features	Instances	Size (GB)
Numeric	986	118347	2.14	Numeric	986	118348	2.14
Categorical	2140	118347	2.68	Categorical	2140	118348	2.68
Date	1157	118347	2.89	Date	1157	118348	2.89
Total	4265	118347	7.71	Total	4265	118348	7.71

3 Related Works

Based on the basic characteristics of dataset described Section 2, we reviewed the related works with three categories: missing data handling, imbalanced data, and binary classification regarding such a dataset.

3.1 Missing Data Handling

Missing data commonly occurs either during measuring data or merging different sources of data in real world, and it significantly affects conclusions derived from the data. The general approaches to handle missing data are as follows: deletion, imputation, and using decision models which can handle missing data [3]. According to the current guideline [4], when the missing rate is less than 10%, we can delete the instances including missing values without significant loss of statistical power. In case that the missing rate is between 10% to 50%, we can impute the values using various kinds of imputation methods. On the other hand, when the the missing rate is greater than 50%, imputation is inherently incorrect and we need auxiliary features.

3.2 Imbalanced Data

Binary classification for imbalanced data is an important topic in machine learn practice, in such problems, most of the instances are labeled as one class, while very few instances are labeled as the other class. To improve the balance of imbalanced data, downsampling and upsampling are commonly used. Downsampling aims to balance class distribution through the selection of majority class examples, while upsampling aims to balance class distribution through increasing minor class examples.

3.2.1 Downsampling

Two general methods for downsampling are random downsampling and informed downsampling. Random downsampling is to randomly sample instances from a majority class, and informed downsampling is to sample major instances through learning process (e.g. EasyEnsemble and BalanceCascade algorithms). Its obejective is to overcome the information loss introduced in the traditional random sampling [5].

3.2.2 Upsampling

Two general methods for upsampling are upsampling with replacement and synthetic sampling. Upsampling with replacement randomly sample the instances without modification in minority classes, and synthetic sampling is to synthesize similar instances from minority classes; for example, SMOTE(Synthetic Minority Over-sampling Technique) [6] synthesizes an instance between two minority instances chosen from k-nearest neighbors, and Borderline-SMOTE [7] increases minor instances only near the borderline between majority and minority class.

3.3 Binary Classification

A binary classification problem seems relatively simple, comparing to multi-label problems. However, it is important to understand what the properties of dataset we deal with are, because as the complexity of features increases, the difficulty of binary classification also increases. Considering the property of our dataset, the first problem is to determine which classifier to choose in our framework. We referred to three models: SVM, Random Forest and Gradient Boosting Tree. Random Forest and Gradient Boosting Tree are supposed to achieve good result based on the experience in Kaggle competitions and the property of tree to handle missing values and different types of features. We also include the support vector machine as a baseline here.

3.3.1 Gradient Boosting Tree

Among the machine learning methods used in practice, Gradient Boosting Tree(GBT) is one technique that shines in many applications. The idea of Gradient Boosting comes from Leo Breiman and later developed by Jerome H. Friedman. The basic idea of GBM to fit a base tree learner to pseudo-residuals.

$$r_{im} = -[\frac{\delta F(y_i, F(x_i))}{\delta F(x_i)}]_{F(x)=F_{m-1}(x))}, i = 1, ...n$$

Then compute r_m by solving the following one-dimensional optimization problem:

$$\sum_{i=1}^{n} L(y_i, F_{m-1}(x) + rh_m(x_i))$$

There are many versions of GBT, such as: Scikit Learn GBT, SAS Viya, XGBoost, FastBDT, LightGBM.

3.3.2 Random Forest

Random Forest in another ensemble method of decision trees, introduced by Leo Breiman[10]. The scoring part for Random Forest and GBT is the same, average the result of decision trees. The difference is the training process. The decision trees in Random Forest are independent, and we can build decision trees in the same time in Random Forest. However, building a new decision tree in GBT is based on the residual between previous prediction and target value. Thus GBT is supposed to much slower than Random Forest, especially in parallel computing system. We will use scikit-learn version Random Forest in our experiments.

3.3.3 Support Vector Machines

Support vector machine is invented by Vladimir N. Vapnik and it is reported to work well for high dimensional data and text data. However, it is not designed to handle missing values, while the Bosch data has very large percentage of missing values. Another shortcoming is that the training is too slow. We want to include scikit learn polynomial SVM [11] as the baseline in the paper.

4 Method

4.1 Data Analysis Framework

Our data analysis framework is delineated on Figure 1. While designing the framework, we concentrated on three points. First, preprocessing should handle missing values and reduce the feature size. Second, sampling should make the data more balanced and contribute to the performance. Third, classifier should be learned fast and accurately on the large data set.

In this framework, we use the numeric and date data, which go through different preprocessing procedures. We apply missing data processing to the numeric data, but not the date data, since its feature extraction method is able to handle missing values. After extracting features from the numeric and date data with each method, the two data are merged, and sampled to make the data more balanced. Then the different classifiers are learned and tested with these sampled data. For details, see the following subsections.

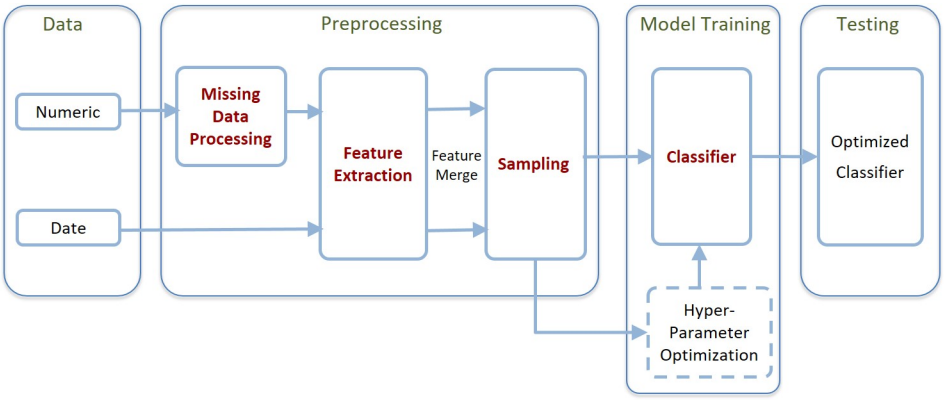

Figure 1: Data Analysis Framework

4.2 Missing Data Analysis

The missing values occupied large percentage, with missing rate in three feature sets being 81%(numeric), 82%(date), 97%(categorical), respectively. Since categorical data has extremely high missing rate, we excluded it in the following analysis. We did statistical analysis with $50,000$ random samples from the 118347 original instances in the numeric dataset. Among total 968 features in the numeric, 104 features have less than 10% missing rate, 52 features between $10-50\%$, and 812 features more than 50%. Figure 2 shows the missing value distributions by location. Since the data was collected and merged from different locations($L0, L1, L2, L3$) and their subsidiary stations, we estimated that the missing patterns are not at random, but systematically generated in the process of merging data from different sources. It implicates a possibility that most of missing values never exist and could not be estimated by any imputation method.

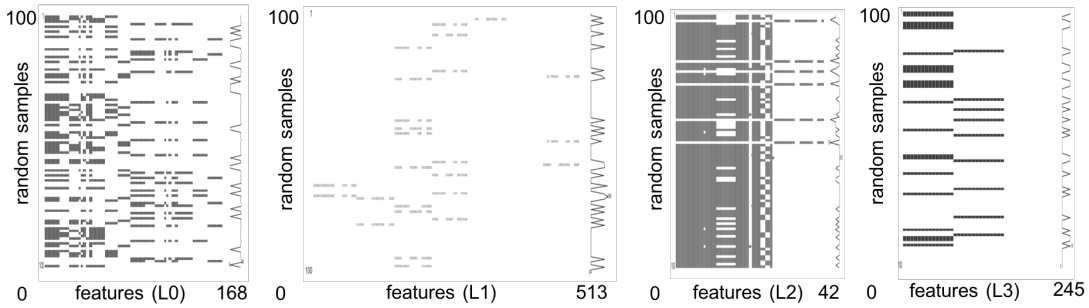

Figure 2: Missing Data Analysis

4.3 Feature Extraction

The feature dimensions are 968 from the numeric data and 1157 from the date data, so the total dimensions are 2125. Based on the missing data analysis described in section 4.2, we designed an experiment to compare two different feature selection approaches for the numeric data.

4.3.1 Extreme Imputation + PCA (EIPCA)

As a baseline of features extraction method with imputation, we considered the combination of extreme imputation and principal component analysis (PCA), which requires no missing values in input data. Extreme imputation is to fill missing values with an extreme value, typically chosen out of the range of non missing values. We set -1.1 as an extreme value for missing values in the numeric data of which value range is between -1 and 1. In the PCA step, we selected 126 and 198 principal components from the range of elbow.

It should be noted that missing by sampling, which is our case, must be distinguished from 'non response missing at random' where people intentionally miss some responses. We assumed that indicating missing values with an extreme value would give less bias than imputing values based on the other features, and designed the next feature selection method.

4.3.2 Feature Selection by Missing Rate (FSMR)

As a feature selection method without imputation, we developed an idea to select features based on missing rate by feature, which is differentiated from the existing guideline to handle missing values by instance. In the first step, we discarded features with missing rate more than 50% because they could cause noises rather than give information. Next, we divided two groups of features with missing rate less than 10% and less than 50%, which are 104 feature and 156 features, respectively.

4.3.3 Date Feature Extraction

This feature extraction idea is from Kaggle forum, not from ours. It is generated by two steps: a) get the starting date of each observation entering the product pipeline, and b) generate the difference of IDs between the current and the 5 next rows after sorting on increasing starting date and ID.

Table 2: Description for Feature Extraction

Data Types	Methods	Reduced Dimensions	Original Dim.
Numeric	FSMR(missing rate by feature) EIPCA(from the range of the elbow)	$104(< 0.1)$, $156(< 0.5)$ $126, 198$	986
Date and ID	Difference of ID sorting by starting date and ID	5	1157

4.4 Sampling

We used sampling for two different purposes. First, we reduced the data size by random sampling in order to accelerate hyper-parameter optimizations. Second, we made the data more balanced using up and downsampling, and examined which method and ratio is optimal for our dataset.

Despite the general guideline that downsampling is preferable to a big and imbalanced dataset, there is no obvious rule about which sampling method and ratio are optimal. It has been even reported that imbalanced data gave better performance than making it balanced in some datasets [5]. Thus, we compared both up and downsampling under different ratios, including the original ratio $1 : 171$. Regarding choosing the specific method for up and downsampling, we compared random downsampling and upsampling with replacement.

4.5 Classifiers

We chose Gradient Boosting Trees, as our main decision model with the following reasons: a) handling missing values and noise inherently, b) fast to learn a large dataset due to parallelization, c) handling complex features including different types of features.

Also, to speed up the training procedure, we considered the binning strategy. Binning strategy is to sort the features according to their scales and divide them into different bins with either equal width or equal quantiles. We adopted XGBoost for equal qunatiles and SAS Viya for equal width, and compared them.

5 Experiment

5.1 Evaluation Methodology

For imbalanced data, merely accuracy is not a good metric to evaluate the classification performance, if it classifies the whole dataset into the majority class, it could still achieve a high accuracy. So we utilized ROC curve as an evaluation measurement. The ROC curve is created by plotting the true positive rate against false positive rate at various threshold settings.

5.2 Classifiers

In order to determine a suitable classification model for our framework, we compared performances of four classifiers including XGBoost, SAS Viya, Random Forest and SVM. Their parameters are determined by grid search. For XGBoost, we fix the learning rate as 0.1, and then search out the optimal tree depth and number of trees which are two important parameters for XGBoost. Here, we utilized concatenated features of all the numeric data and the transformed date data.

The ROC curves of four classifiers are shown in Figure 3 and 4. First, XGBoost achieves the best performance comparing to other methods, and Random Forest and SAS Viya GBT are the followings. The ROC of SVM is nearly coincident with the diagonal, which is nothing better than random guess. Second, all of the four methods show similar results between validation and testing dataset, indicating these methods have good generalization performances. Third, the binning strategy can highly affect the perfomance of GBT, because SAS Viya is even worse than Random Forest.

The binning is highly related to feature distributions. From the result, we can interpret that quantile binning (XGBoost) outperforms equal binning(SAS Viya) if there are many skewed features. So we analyzed our dataset and found that our dataset has different distributions among features that some

of these features are normally distributed, while others are highly skewed, as shown in Figure 5 and Figure 6, respectively.

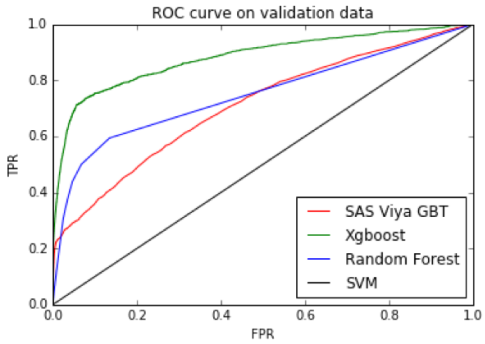

Figure 3: Four Classifiers on Validation Dataset

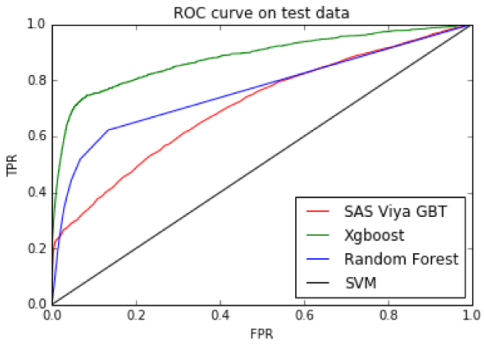

Figure 4: Four Classifiers on Test Dataset

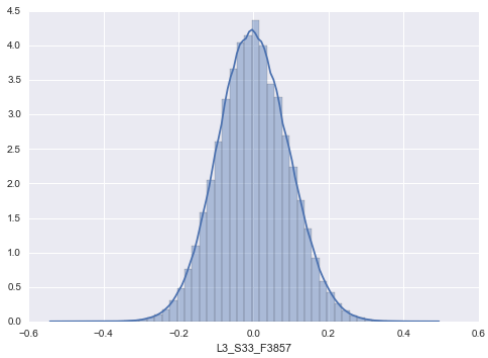

Figure 5: normal distributed feature

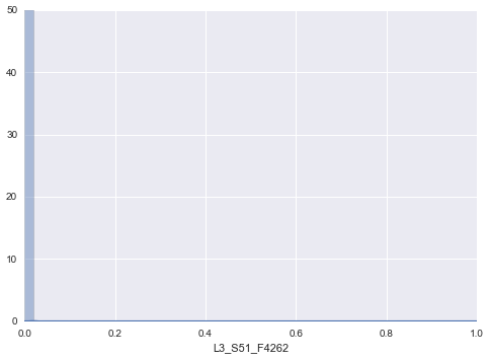

Figure 6: skewed feature

5.3 Sampling

We used XGBoost as the optimal classification model in this experiment, and the numeric data is preprocessed by FSMR with missing rate less than 0.1, and then concatenated with the transformed date data. Then we compared two sampling strategies, i.e. random downsampling and upsampling with replacement.

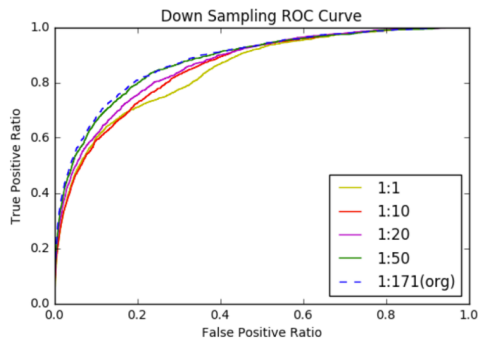

Figure 7: Random Downsampling

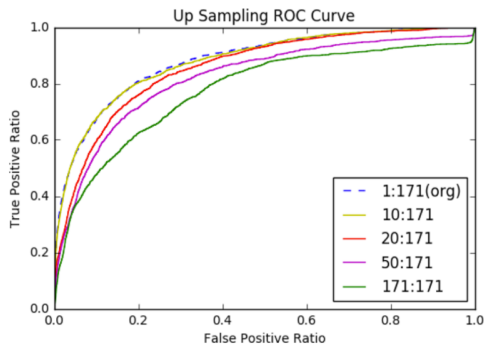

Figure 8: Upsampling with Replacement

The ROC curves of two sampling strategies are shown in the Figure 7 and Figure 8. First, $1:50$ ratio random downsampling can achieve comparable performance to the original ratio $1:171$. Second, considering it requires more complexity, upsampling with replacement is not competitive, even though its $10:171$ ratio achieves similar results to the original ratio.

5.4 Feature Extraction

We implemented the experiments to compare two feature extraction strategies, EIPCA (Extreme Imputation + PCA) with 126 and 198 principal components, and FSMR (Feature Selection by Missing Rate) with 104 (missing rate < 0.1) and 156 (missing rate < 0.5) selected features. These experiments were conducted on XGBoost with random downsampling ($1:50$ ratio).

Figure 9 and Figure 10 show the ROC curves comparing these methods, where Figure 9 contains the results only utilized the numeric data, while Figure 10 contains the results combined the numeric with the date data. From the figures, we can see that FSMR achieves better performance than EIPCA, indicating either the extreme imputation is not beneficial or PCA is not very effective when the missing rate is high. Another notable point is that $FSMR\_0.1$ (with missing rate less than 10%) also shows a similar performance to $FSMR\_0.5$ (with missing rate less than 50%), which is the best ROC curve. Regarding the number of features in $FSMR\_0.1$ is only two third of $FSMR\_0.5$, it can be interpreted that the considerable amount of features are less informative in the numeric data.

In addition, it is confirmed that the feature extractions contribute to the better performance, when comparing the results shown in Figure 4 and Figure 10. It implicates the feature extractions discarded some noise and chose the representative features for classification.

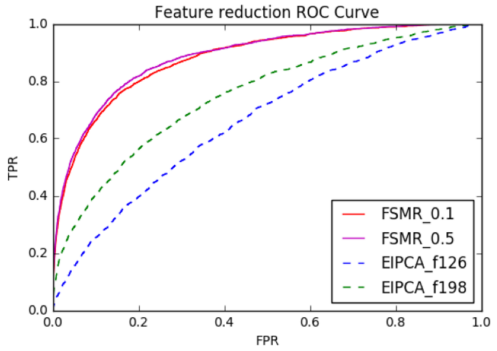

Figure 9: ROC of feature extraction (numeric)

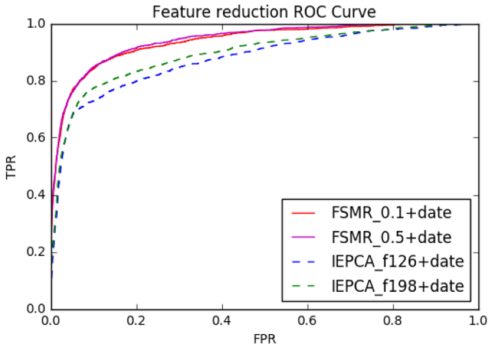

Figure 10: ROC of feature extraction (numeric + date)

6 Discussion

In summary, our framework could be utilized for binary classification of big, imbalanced data with large, complex features and high missing value rate. The best strategies determined through experiments are XGBoost as a classifier, random downsampling for reducing and balancing data, and FSMR for reducing missing value rate and feature selection. Since the characteristics revealed from our dataset commonly exist in many other modern manufacturing industries, this framework might also been applied to other problems in manufacturing.

However, in order to verify the effectiveness of our framework, we need to explore better imputation and upsampling methods. Also, to improve its performance, we should incorporate a method enable to analyze the categorical data in our framework, even though it has extremely high missing rate.

Despite some limitations, we learned several lessons through our data analysis process. First, when a dataset has high missing rate, it might be a good start to select features based on missing rate by feature. Second, XGBoost can be a good candidate to solve a classification problem with big and complex dataset. Lastly, for a big and imbalanced data, random downsampling could achieve comparable performance to the original dataset with less time and space complexity, while upsampling with replacement can cause overfitting problem.

7 Reference

[1] Choudhary, A.K., Harding, J.A. & Tiwari, M.K. (2009) Data mining in manufacturing: a review based on the kind of knowledge. *Journal of Intelligent Manufacturing*, **20**:501-521.

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

[3] Cismondi, F, Fialho, A.S., Vieria, S.M., Reti, S.R., Sousa, J.M.C., & Finkelstein, S.N. (2013) Missing data in medical databases: Impute, delete or classify?, *Artificial Intelligence in Medicine*,**5**(8):63-72.

[4] Dong, Y., & Peng, C.Y.J. (2013) Principled missing data methods for researchers, *SpringerPlus 2* 1–17. doi:10.1186/2193-1801-2-222.

[5] He, H., & Garcia, E. (2009) Learning from Imbalanced Data. *IEEE Transactions of Knowledge and Data Engineering*, **21**(9):1263-1284

[6] Chawla, N.V., Japkowicz, N. & Kotcz, A. (2002) SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, **6**(1):321-357.

[7] Han, H., Wang, W.Y., & Mao, B.H. (2005) Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. In D.S. Huang, X.-P. Zhang, G.-B. Huang (Eds.), *International Conference on Intelligent Computing*, Part I, LNCS 3644, pp. 878-887. Springer-Verlag.

[8] Tianqi Chen, & Carlos Guestrin. (2016) XGBoost: A Scalable Tree Boosting System. *KDD.*

[9] Keck T. (2016) FastBDT: A speed-optimized and cache-friendly implementation of stochastic gradient-boosted decision trees for multivariate classification *arXiv preprint arXiv*: 1609.06119.

[10] http://www.sas.com/en_us/software/viya.html

[11] Breiman, L. (2001) Random forests. *Machine learning*, **45**(1):5-32.

[12] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

8 Appendix

8.1 Teammate and Work Division

Our code link is http://github.com/DataScienceNCSU/ImbalancedData

The work division is as follows:

Xi & Yeojin: data exploration, feature reduction, & sampling

Liang: data exploration, classifiers, & hyperparameter optimization