Monitoring Production Line Performance to Reduce Manufacturing Failures

Desh Raj, Abhilasha Sancheti, Mrinal Tak, and Kunal Jain
Department of Computer Science and Engineering
Indian Institute of Technology Guwahati
Guwahati, India - 781039

***Abstract*—Predicting failures in manufacturing processes is important for improving cost and time efficiency in any industry. Modern engineering relies upon fine-grained data captured at every step along manufacturing lines for fault analysis. However, such an analysis poses two challenges: (1) faults occur extremely rarely, which makes it difficult for a machine learning system to model the characteristics of a faulty sample, and (2) since measurements are obtained at each station, the data is very high-dimensional. In this work, we pose the task of fault analysis as a binary classification problem, and explore a combination of various techniques to solve this problem for a high-dimensional skewed distribution containing numerical, categorical, and timestamp features. These include various sampling methods, feature selection, sparse matrix approaches, and a meta-optimization on the evaluation metric. We analyse these on the Bosch Production Line Performance data set, and evaluate each technique incrementally to demonstrate how pre-processing, classifier selection, and post-processing may improve classification performance.**

I. Introduction

With an increasing emphasis to reduce production failures in manufacturing processes, data science methods are being proposed as the next big revolution. Especially with the ushering of an *Internet of Things (IoT)* era, industries are producing large sets of high-dimensional data every day to analyze and detect failure patterns. This focus towards collection, aggregation, organization, and analysis of data is driven by factors such as reduced price of sensors, rapid decline in the cost of data storage and computation, and increasing accuracy of machine learning methods in modeling diverse processes.

However, this abundance in data generation also leads to a few challenges. First, due to the presence of multiple sensors at each station along the production line, a large number of features are generated for every sample, and most of these features may be heavily correlated or entirely unimportant. Further, if the training set is small in size, any learning algorithm may grossly overfit on this high-dimensional data, such that the model performs poorly in practice.

Second, in addition to a high dimensionality, modern manufacturing processes are highly efficient such that failures are few and far between. For this reason, any distribution of data obtained from industry is highly skewed towards positive samples (success) which makes it difficult for machine learning systems to learn the characteristics of negative samples (failures). The skewed class distribution problem is common in many settings, and various sampling techniques have been devised to tackle these [1], [2], [3].

Fault analysis is often tackled using either of two approaches:

1) *Anomaly detection*: In data mining, anomaly detection is the identification of items, events, or observations which do not conform to an expected pattern or other items in a dataset. Unsupervised anomaly detection techniques detect anomalies in an unlabeled test data set under the assumption that the majority of the instances in the data set are normal by looking for instances that seem to fit least to the remainder of the data set.
2) *Binary classification*: It is a supervised learning task in which we pose the task of predicting internal manufacturing failures as that of classifying datapoints into two classes, namely success or failure. Given examples of datapoints each marked with one of the classes, a classifier can learn to predict the correct class for a datapoint. Classifiers typically assume that the number of datapoints in the two classes as balanced. This assumption is violated when detecting internal manufacturing failures, since these occur very rarely in typical production workflows.

In this work, we pose the problem of fault detection as a binary classification task. We propose and evaluate a combination of various techniques to simultaneously tackle the problems of high-dimensionality and skewed class distribution. We begin by sampling the dataset to obtain a smaller representative distribution. The numerical, categorical, and timestamp features are then independently reduced using several methods to obtain a smaller dimensionality.

This sampled data set is then evaluated using weighted and non-weighted classifiers to determine the best base classifier. In a final post-classification step, the minority class is oversampled and a Bayesian approach is used to optimize the evaluation metric by varying the weight of each class as parameter in the weighted classification.

II. Dataset Description

We use the Bosch Production Line Performance data set released by Bosch as part of a Kaggle machine learning contest [1].

Bosch has supplied a huge dataset (14.3 Gb) containing three types of feature data: numerical, categorical, date stamps

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

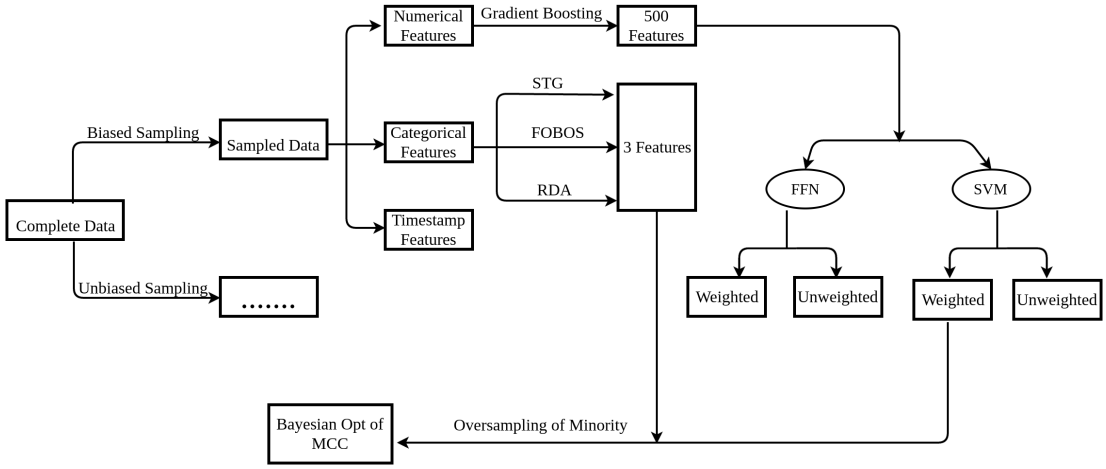


Fig. 1: Architecture of the proposed method.

and the labels indicating the sample as good or bad. There are 968 numerical features, 2140 categorical features and 1156 date features.

The training data has 1184687 samples and the learned model was used to predict on a test dataset containing 1183748 samples. However, since the labels of the test dataset were not provided, we discarded the test set and used only the training samples for our evaluation.

III. Proposed Method

An overview of the proposed method is represented through the flow diagram shown in Fig 1. As shown in the figure, the proposed method comprises 4 major stages, i.e., (i) initial sampling, (ii) feature selection, (iii) choice of base classifier, and (iv) Bayesian optimization of evaluation metric. In this section, we discuss each of these stages in some detail.

A. Initial sampling

Due to a large sample size, it becomes difficult (or even impossible) to train a classifier on the entire dataset. Further, it may be redundant to perform such a training since the characteristics of each class may be well represented by a smaller sample size. For these reasons, our first step is undersampling the entire dataset, using an unbiased and a biased approach as explained below.

1) Unbiased: In this approach, we select a subset of the original samples without taking into account the corresponding labels.

2) Biased: In a biased sampling technique [3], all the positive instances are retained while performing sampling. The rationale behind this approach is that if the number of positive samples are low in the original dataset, further subsampling may lead to information loss.

In practice, we found that a 0.001 fraction of the training set was sufficiently able to represent the characteristics of all the samples. Once samples are obtained, we split them in an 80:20 ratio for training and testing, respectively. It is to be noted that a stratified sampling technique was used for performing this split. The statistics of the biased and unbiased samples thus obtained are shown in Table III. During further experimentation, it was found that the biased sample naturally performs better than the unbiased sample, and hence we take this sample under consideration.

Statistic	Biased	Unbiased
Total samples	73693	59188
Train size	58954	47350
Test size	14739	11838
Positive instances (train)	5456	278
Positive instances (test)	1423	

TABLE I: Dataset Summary Statistics.

B. Feature selection

After undersampling records, we perform feature selection using several methods to select the most important features among numeric, categorical, and timestamp features.

1) Numerical features: There have been numerous investigations for determining efficient methods of dimensionality reduction for numerical features. While traditional approaches such as principal component analysis (PCA), factor analysis, and classical scaling employed linear methods, various nonlinear approaches, such as kernel PCA, Isomaps, maximum variance unfolding, etc., have recently been proposed for the same purpose. Despite the large variability in these approaches, they usually fail to outperform traditional linear techniques [4] due to several reasons such as (a) presence of trivial optimal solutions, (b) reliance on convex objective functions, and (c) dependence on neighbourhood graphs to model the local structure of the data.

We use a Gradient Boosting Machine [5], [6] technique (GBFS) for numerical feature selection in our architecture. This choice of algorithm is due to the following properties of GBFS:

1) As it learns an ensemble of regression trees, it can naturally discover nonlinear interactions between features.

2) In contrast to, e.g., random forests, it unifies feature selection and classification into a single optimization.
3) In contrast to existing nonlinear algorithms, its time and memory complexity scales as $O(dn)$, where d denotes the number of features and n the number of data points, and is very fast in practice.
4) GBFS can naturally incorporate pre-specified feature cost structures or side-information, e.g., select bags of features or focus on regions of interest, similar to generalized lasso in linear feature selection.

We do not describe the details of the algorithm here for sake of brevity. We select 500 features from the 968 given features and use this in further experimentation. During validation, it was found that selecting fewer features resulted in performance decrease, whereas more features did not necessarily improve the classifier performance.

2) Categorical features: Traditionally, categorical features are represented in one-hot notation and concatenated with numerical features. However, due to the presence of 2140 categorical features in our given dataset, conventional feature selection algorithms fail miserably in selecting important categorical features.

Instead, we employ three popular sparse online classification techniques to learn the one-hot encoded categorical features. These are described below in some detail

1) *Stochastic Truncated Gradient (STG)* [7]: Consider a loss function $L(w, z_i)$ which is convex in w, where $w_i = (x_i, y_i)$ is an input-output pair. The soft-regularization formulation of $L1$-regularization can be defined as

$$\widehat{w} = \arg\min_{w} \sum_{i=1}^{N} L(w, z_i) + g||w||_1 \tag{1}$$

This can be understood as an online version of an efficient $L1$ loss optimization approach. At a high level, it works with the soft-regularization formulation 1 and decays the weight to a default value after every online stochastic gradient step. This simple approach enjoys minimal time complexity (which is linear in k and independent of d) as well as strong performance guarantee. For mathematical details, the reader is advised to refer the paper by Langford et al. [7].

2) *Forward-Backward Splitting (FOBOS)*: [8] In an online approach, every iteration of the FOBOS algorithm consists of two steps:

$$w_{t+\frac{1}{2}} = w_t - \alpha_t g_t \tag{2}$$

$$w_{t+1} = \arg\min_{w} \{\frac{1}{2}||w - w_{t+\frac{1}{2}}|| + \lambda||w||_1\} \tag{3}$$

This algorithm removes the problems of non-differentiability in cases such as l1-regularization by taking minimization steps mixed with sub-gradient steps. For mathematical details, the reader is advised to refer the paper by Duchi et al. [8].

3) *Enhanced Regularized Dual Averaging (ERDA)*: [9]: At each iteration of the ERDA algorithm, the sum of three terms is minimized, the dual average- a linear function which is obtained by averaging all previous sub-gradients $g'_t = \frac{t-1}{t} g'_{t-1} + \frac{1}{t} g_t$, the regularization function $\Psi(w) = \lambda||w||_1$, and an additional strongly convex regularization term $\frac{\gamma}{\sqrt{t}}(\frac{1}{2}||w||_2^2 + p||w||_1)$. The weights are updated as

$$w_{t+1} = \arg\min_{w} \{< g'_t, w > +\lambda_t^{RDA}||w||_1 + \frac{\gamma}{2\sqrt{t}}||w||_2^2\} \tag{4}$$

For mathematical details, the reader is advised to refer the paper by Xiao et al. [9].

Each of these sparse online learning algorithms are trained on 50% of the training set and then used to predict scores for the training and test sets. These scores may then be considered as representative of all the categorical features. In essence, we thus obtain 3 numerical features which can effectively replace the 2140 categorical features (at the cost of some acceptable loss of information).

3) Timestamp features: We extracted several manually engineered features from the date and time features, such as time interval between stations, maximum/minimum/average time, time of day at particular stations, etc. However, such a feature engineering was found to fail in improving classifier performance. For this reason, we ignore these features and continue with only the selected numerical categorical features.

C. Selection of base classifier

After obtaining a subsampled dataset with selected numerical features, we evaluate several classification algorithms to select a base classifier for further optimization. In our experiments, we have evaluated two methods, namely Support Vector Machine (SVM), and a Feedforward Neural Network (FFN) method. However, any classifier may be exploited for this purpose, depending upon the user's preferences.

1) A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples.
2) Feedforward networks consist of multiple layers of computational units, usually interconnected in a feed-forward way. Each neuron in one layer has directed connections to the neurons of the subsequent layer. For our classification purpose, we used a single hidden layer perceptron wherein the units of these networks apply a sigmoid function as an activation function.

We further evaluate two versions of each method: the general technique and a weighted variation [10]. The results of each of these are summarized in Table II.

Since the weighted SVM was found to perform best, we selected this method as the base classifier for the Bayesian optimization phase. We further included the 3 features obtained from the categorical features with the numerical features for further training, and

Model	Non-weighted			Weighted		
	Precision	Recall	F1-score	Precision	Recall	F1-score
FFN	**92.53**	4.35	8.32	58.11	10.82	18.24
SVM	84.61	0.77	1.53	13.25	**67.39**	**22.15**

TABLE II: Results of base classifier selection process (in %) for weighted and non-weighted FFN and SVM, using 500 numerical features selected earlier.

Statistic	Precision	Recall	F1-score
Initial	13.25	**67.39**	22.15
+ SMOTE	13.35	66.83	22.25
+ categorical	**19.41**	52.21	**28.30**

TABLE III: Results (in %) obtained using SMOTE and categorical features.

D. Bayesian Optimization of evaluation metric

This stage comprises two components, namely SMOTE oversampling [2], and Bayesian optimization on the evaluation metric.

1) Synthetic Minority Oversampling Technique (SMOTE): : The minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors. Depending upon the amount of over-sampling required, neighbors from the k nearest neighbors are randomly chosen. For instance, if the amount of over-sampling needed is 200%, only two neighbors from the five nearest neighbors are chosen and one sample is generated in the direction of each. Synthetic samples are generated in the following way: Take the difference between the feature vector (sample) under consideration and its nearest neighbor. Multiply this difference by a random number between 0 and 1, and add it to the feature vector under consideration. This causes the selection of a random point along the line segment between two specific features. This approach effectively forces the decision region of the minority class to become more general.

2) Bayesian optimization: [11]: Most machine learning classifiers optimize the linearly decomposable metric of accuracy by minimizing a loss function. We optimize a modified parametrized objective function $g(w)$ with parameter w such that parameter balances the trade-off between losses incurred on the samples belonging to the positive and negative classes. The model obtained by solving this optimization problem is denoted by $M(w)$. By varying the parameter w, we can correct the imbalance between losses incurred on the samples belonging to the positive and negative classes.

$$M(w) = \arg\min g(w) = \arg\min\{w \cdot FP + (1-w) \cdot FN\} \quad (5)$$

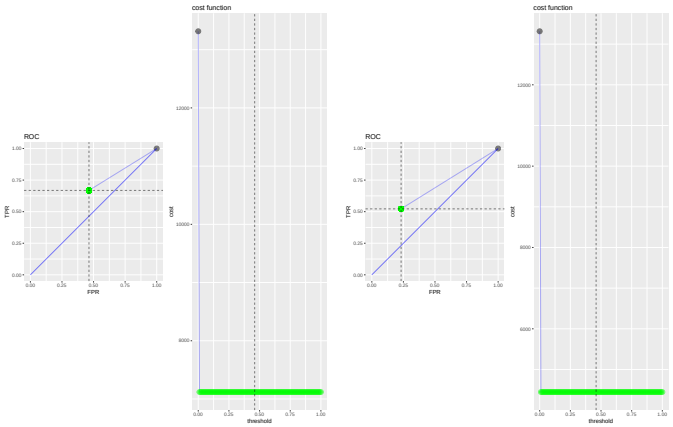
where w is a parameter between (0,1).

Here, $M(w)$ is the SVM classifier obtained by minimizing the weighted loss function $g(w) = \{w \cdot FP + (1w) \cdot FN\}$. Thus, we have a technique for providing a weight w, optimizing the weighted loss function $g(w)$ on the training dataset to obtain a classification model $M(w)$, and receiving the evaluation metric such as the Matthew's Correlation Coefficient MCC(w) on the validation dataset. By varying w, we search over a large space of machine learning models for a model M^* that directly optimizes MCC(w) as

$$M^* = \arg\max MCC(w). \quad (6)$$

Using a meta-optimization technique like Bayesian optimization, we can find the parameter w such that parametrized algorithm returns the highest MCC. Bayesian optimization [11] based on Gaussian processes is good for exploration versus exploitation trade-off in black-box optimization, and focuses on areas of parameter space w that have higher chances of attaining maximum objective value MCC(w). As a result, we chose Bayesian optimization as our meta-optimization algorithm.

At the time of submission of this report, this meta-optimization is still training, though initial exploration suggests an improvement of 4-5% on the MCC using w approximately between 0.3 and 0.4.



(a) **(b)**

Fig. 2: Effect of feature types on ROC in weighted SVM: (a) numerical features, and (b) numerical + categorical features.

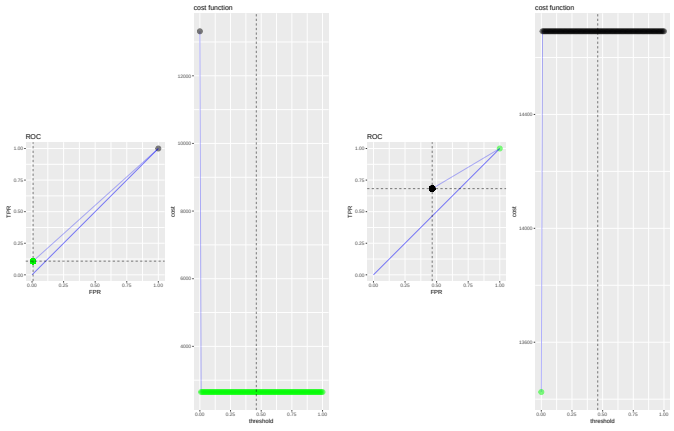
IV. Results and Discussion

A. Effect of feature types

The data set consists of numerical, categorical and timestamp features, of which we use the 2 former feature types for classification purpose. During experimentation, it was found that most of the sensitivity of the base classifier was obtained due to the numerical features, and the 3 categorical features used as described earlier only contributed a little in improving performance. This is pictorially represented in Fig. 2, where it is evident that adding categorical features barely changes the ROC curve. The area under curves (AUC) for the two ROCs are found to be 0.6023 and 0.6452, respectively.

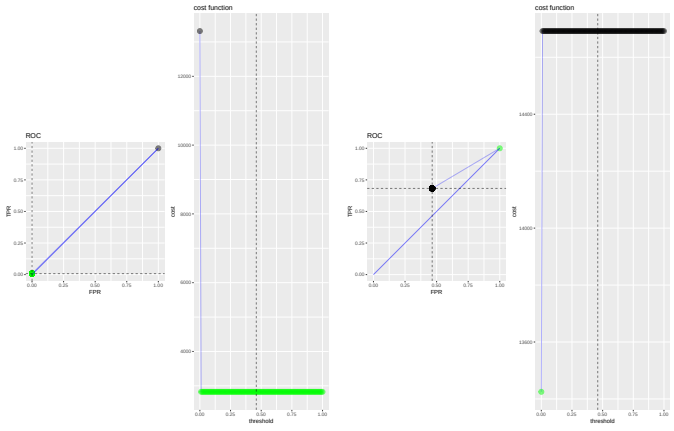
B. Base classifier performances

The variation of classification performances for FFN and SVM classifiers is more glaringly obvious from the ROC curves shown in Fig. 3. The AUCs for the same were found to be 0.5499 and 0.6014, respectively.



(a) **(b)**

Fig. 3: ROC variation for (a) feedforward network (FFN) and (b) support vector machine (SVM) classifiers.



(a) **(b)**

Fig. 4: Effect of class weighting on SVM classifier performance: (a) unweighted and (b) weighted SVM.

C. Effect of class weights

The effect of class weighting on the SVM classifier performance is shown using the ROC curves in Fig. 4. The AUCs for the same were found to be 0.5038 and 0.6014, respectively.

V. Conclusion

In this work, we explored the simple task of binary classification in an industrial setting, such that the large sample sizes and high dimensionality of features made the task extremely challenging. For this exploratory analysis, we evaluated several feature selection techniques, base classifiers, and post-classification optimization, and organized these diverse methods in one architecture that greatly improved classification performance. While a simple SVM trained on the full dataset was found to train for several days and performed poorly (AUC$\sim$0.5), our final model trains quickly and performs relatively well, with an AUC of $\sim$0.65. While this performance may not be considered "good" in an absolute sense, the relative improvement along with the time and memory efficiency speaks for the advantages of our architecture. We argue that a better base classifier such as Extreme Gradient Boosting, and better sparse online learning algorithms such as the Follow the Regularized Leader (FTRL) method may be able to improve classification performance further. We also believe that some pattern may be obtained by carefully analyzing the timestamp features, although our initial exploration suggested otherwise.

References

[1] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 2, pp. 539–550, 2009.

[2] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[3] G. Kollios, D. Gunopulos, N. Koudas, and S. Berchtold, "Efficient biased sampling for approximate clustering and outlier detection in large data sets," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 5, pp. 1170–1187, 2003.

[4] L. Van Der Maaten, E. Postma, and J. Van den Herik, "Dimensionality reduction: a comparative," *J Mach Learn Res*, vol. 10, pp. 66–71, 2009.

[5] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[6] Z. Xu, G. Huang, K. Q. Weinberger, and A. X. Zheng, "Gradient boosted feature selection," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 522–531, ACM, 2014.

[7] J. Langford, L. Li, and T. Zhang, "Sparse online learning via truncated gradient," *Journal of Machine Learning Research*, vol. 10, no. Mar, pp. 777–801, 2009.

[8] J. Duchi and Y. Singer, "Efficient online and batch learning using forward backward splitting," *Journal of Machine Learning Research*, vol. 10, no. Dec, pp. 2899–2934, 2009.

[9] L. Xiao, "Dual averaging methods for regularized stochastic learning and online optimization," *Journal of Machine Learning Research*, vol. 11, no. Oct, pp. 2543–2596, 2010.

[10] J. A. Suykens, J. De Brabanter, L. Lukas, and J. Vandewalle, "Weighted least squares support vector machines: robustness and sparse approximation," *Neurocomputing*, vol. 48, no. 1, pp. 85–105, 2002.

[11] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, "Boa: The bayesian optimization algorithm," in *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1*, pp. 525–532, Morgan Kaufmann Publishers Inc., 1999.