2017 13th IEEE International Conference on Control & Automation (ICCA)
July 3-6, 2017. Ohrid, Macedonia

Machine Learning for Large Scale Manufacturing Data with Limited Information*

S. Nedelkoski[1] and G. Stojanovski[1]

Abstract— **Improving the efficiency of the production plants has always been focus of manufacturing industry. Recently the utilization of data analytics tool is dramatically increased since these methods bring new insights into already existing data. Nevertheless, manufacturing industry in general is still reluctant to make the data available to researcher due to privacy issues. One such example is the challenge sponsored by Bosch and run by kaggle.com, where anonymized data was made available to data scientist with very limited description. In this work, we present our solution to the Bosch assembly line performance challenge, specifically in respect to dealing with raw big data without detailed explanation. The data science methods applied were used to successfully predict internal failures along the assembly lines, although no details of the structure and line description was available.**

I. Introduction

Data science and machine learning frequently used by large manufacturing companied to exploit and extract more information from their databases. Nevertheless, these companies do not share the data with the data science community, mainly due to privacy and intellectual property issues. Also making the data available to the public will give the competition advantage and insight into the organization and production plans of the company.

Small improvement in the direction of making data available to the scientist was made through the Bosch challenge, hosted and run by kaggle.com [1], where a large (11 GB on hard disk, 80GB loaded as float in RAM) dataset was made available for public competition. Unlike classical data science competitions, here the additional challenge was the fact that the data was delivered with very limited description. Many data scientist approached the problem from different aspects. Some of the remarkable works could be found in [2], [3].

In this work, we present an approach to model the product failures based on a measurements taken on an assembly line, with limited knowledge of the real plant. The outline of the rest of the article is divided in as follows: Section II introduces the detailed problem description. Section III is focused on exploring the data and initial data analysis to extract information regarding the correlation of the features to the assembly line failures. Section IV presents the model proposed. Finally, Section V presents some results. Section VI concludes the paper with remarks and hints about future work.

*The authors would like to express their gratitude to Mr. Aditya Sinha, Mr. Prakash Subedi, Mr. Abilash Awasthi and Mr. Mohd Shadab Alam for the support and collaboration during the Kaggle challenge

[1] S. Nedelkoski and G. Stojanovski are with the "Ss. Cyril and Methodius University", Skopje, Macedonia, Faculty of Electrical Engineering and Information Technologies goranst@feit.ukim.edu.mk

II. Problem Description

In the manufacturing industry there are assembly lines and each of them has stations along the lines. In modern industry, the companies that manufacture these parts collect data at each station of these lines. Not all of the produced parts are of desired quality, there are parts that do not pass the required quality check performed after they have been produced. This affects the company in many ways such as:

- selling parts which do not correspond to the required quality and safety standards
- direct impact on company's profits

As mentioned above, data is recorded at each station, which automatically gives us a mechanism to improve the manufacturing process by using an advanced analysis. Predicting internal failures by using thousands of measurements and tests performed for each component along the assembly line would enable a company to provide products with better quality at lower costs to the end user.

A. Data Description

The data used here represents measurements of components as they move along the Bosch's assembly lines. Each component has a unique ID and multiple measurements are recorded on a fixed position stations. The goal is to predict which components will fail quality control (represented by a Response = 1)

The dataset contains an extremely large number of anonymized features. The features are named according to a convention that contains the production line, the station on that line, and a feature number (i.e. measurement). E.g. L3_S36_F3939 stands for line 3, station 36, and feature number 3939. Because of the size, the dataset is distributed in three separate files that contain the numerical, categorical and date features respectively. The date features represent the timestamps of each measurement. 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 (see Table I). In total there are 969 numerical, 2140 categorical and 1156 date features.

The number of failure events (Response column in Table I) is highly unbalanced (less than 1% failed components), which makes the predicting task significantly more complex.

The original dataset is divided by the organizers into three groups. The first one is train dataset, which is publicly available for developers to train their models; the second one

TABLE I

DATASET STRUCTURE OF NUMERICAL AND DATE FEATURES

ID	L0_S0_F0	L0_S0_D0	...	Response
4	0.030	82.24	...	0
...	...	...	...	...
7	0.088	1618.70	...	1

is public test dataset (random 30% sample of the complete test set); and the third is the private dataset which was used by the organizers to calculate the final score in order to check if the algorithm makes proper generalization. The developers used cross-validation techniques and public dataset results to evaluate their performance.

B. Evaluation Function

The results of the machine learning models are evaluated using the Matthews's correlation coefficient (MCC) [4] between the predicted and observed response (1).

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \quad (1)$$

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

III. EXPLORATORY DATA ANALYSIS

Statistical analysis is the initial approach for all anonymized datasets. At the beginning we create plots on the numeric features for visual inspection, correlation matrices, and manually search the data to check is there are patterns. One can easily notice that the data contains duplicate rows. Whenever a duplicate row is detected, the probability that at least one of the duplicate rows represents a failed component (the "Response" feature column of at least one of the duplicate rows equals one) is significantly greater than in unique rows. This data property will be exploited together with the numerical and date features.

A. Feature Extraction

We divide the features into three categories based on what the features represent. The categories defined are:

1) numerical features;
2) categorical features; and
3) date features.

In the following subsections we will give detailed explanation about the characteristics of all feature types.

1) Numerical Features: The data provided by Bosch contains 969 numerical features of process measurements in total. The value of the feature represents the corresponding measurement at a certain station (identified only by a code in the table column). This dataset contains data from 4 assembly lines form the plant with 51 stations distributed among the lines. Each particular component, in the process of assembling, is driven through different set of stations. One can naturally conclude that in a production plant multiple different products are being assembled and they are processed through different stations. The average number of measurements for each product is 183.

TABLE II

THE FAILURE RATE PERCENTAGE PER STATION IN DESCENDING ORDER

Station ID	No. Features	Samples	Error rate(%)
S32	1	24543	4.719
S24	229	183727	0.8348
S38	3	27142	0.7872
...	...	...	...
S33	10	1114695	0.5
S44	8	29804	0.4923
S31	4	39003	0.2725

TABLE III

THE STATIONS WITH STRONGEST CORRELATION TO THE ERROR RATE

S32	S33	S34	response == 1	samples	error
0	0	0	319	62482	0.005105
0	1	0	5	1529	0.00327
0	1	1	5449	1095190	0.004975
1	0	0	837	4602	0.181877
1	0	1	177	1965	0.090076
1	1	1	92	17960	0.005122

Next, we need to implement data pre-processing such as removing perfect correlation and removing duplicates so we end up with 604 numeric features (about 5GB) and reading them in sparse matrix format has reduced the size of the numerical features to about 1.5GB. That allows to process the data and extract features even on computers with low memory (under 8GB). Figure 1 shows the number of jobs that each station executes, from which we can see the production flow across different station-line combinations and the number of jobs passing through. We used all numerical features since we believe they carry the main information from predictive point of view. We trained our first model with XGBoost [5] algorithm for classification (which is explained in details below) on only numerical features plus the number of non-NAN values. Table II presents the top 3 and bottom 3 rows of the failure rate table, presented in descending order. One can notice that the feature importance plot shows the importance of some numerical features that are obviously carrying more information than others. Hence, while exploring if certain lines or stations that have high score of importance were correlated to higher error rates we found that is worth mentioning that Station 32 in particular gives 4.7% error rate, compared to the mean error rate of 0.6%. The station has only one feature marked with column ID: L3_S32_F3850.

The next logical step is trying combination between station in order to find some with high error rate, as presented in Table III, and the most interesting pattern is S32, S33, and S34 see table. We can see that if part passes through S32 and do not pass though S33 and S32 we have 18.1% error rate. This pattern will be exploited more when we discuss about Leak/Data property features.

where 0 stands for components that didn't pass through the

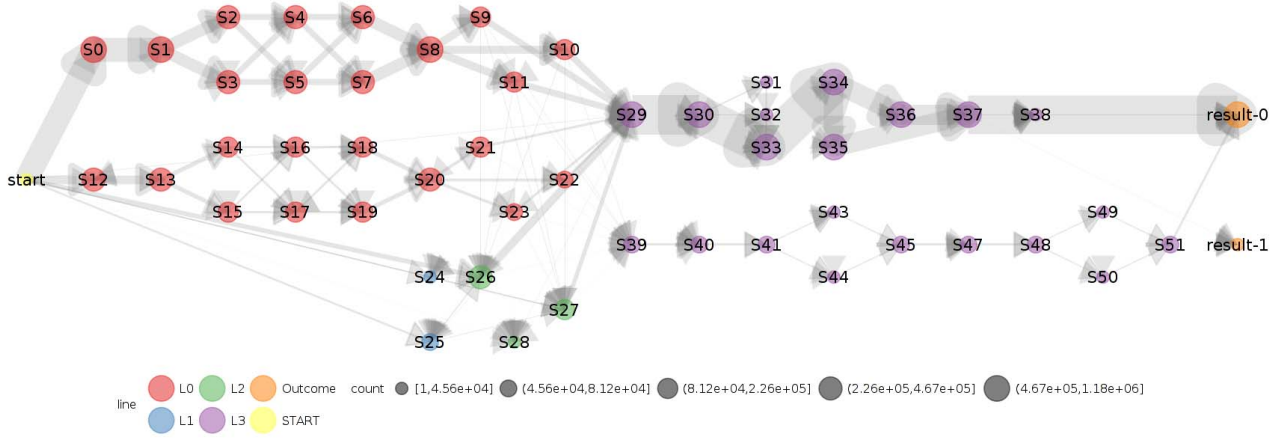

Fig. 1. The number of parts that pass through each station.

station, and 1 stands for components that passed through station.

2) Categorical Features: The file that contains categorical data has 2140 features. We can separate all categorical features into three classes: with single value, empty and with multiple values. In the final model we did not include any categorical features since initial research and statistical analysis have shown that their contribution to the failure detection is very low.

3) Date Features: The date features are timestamps at the moment of measurement at each station. This data is anonymized so the user can only get the resolution of a time interval, defined as a number without any unit attached to it. In this case this number is 0.01. The duplicate columns, present in the raw data, present large number of measurements within one time period. Therefor, one can conclude that the time interval of 0.01 represents a relatively large time unit, in particular 6 minutes [6].

Using the time interval as a basis we will present ideas for feature construction, based on the time information. Small list of such features includes:

- Start Time (the time when the product/component enters the assembly line),
- End Time (the time when the product/component leaves the assembly line),
- Duration (of assembling),
- Station Time Difference,
- Number of records within a period.

Additionally we will take advantage of the fact that there is row based correlation between parts processed at the same starting time. This property will also be exploited in the Leak/data property features in Section III-A.4.

4) Leak/Data Property Features: If we analyze the details of the data, we will notice that most of the failed components appear in the duplicates rows. The hypothesis behind this property is that the measurement system records data duplicates when it experiences power glitches or some kind of station failure. Using the information that the general characteristics of the components processed are strongly dependent on the component ID and of the time of processing, we created some features which are based on row vs. row correlation, such as difference between components ID. This is done by first sorting the data by ”Start Time” and then computing for each row the difference between components IDs for the given row and the consecutive row. The logic behind this feature is that, when the data is sorted by Start Time, subsequent component IDs denote components that were positioned closely on the assembly line. Therefor, the possibility of failure detection is correlated.

We repeat this procedure in ascending and descending direction, for data sorted once by ”ID” and once by ”Start Time”, we generate four new features. In Figures. 2-3 we can observe the basic characteristics of the features plot generated by sorting ”Start Time” and difference between component ID. One can notice that there is structural difference between failed and non failed components that we can use to improve our results.

As addition to the extracted feature error rate from the combination of stations S32, S33 and S34, we include the newly created, so called, ”magic” features. When we include the ”magic” features, that are computed using the difference between IDs and sorted by ”Start Time” column, we generate a model that predicts error rate of 69% for the failed parts when the component passes through station S32, don’t pass through S33 with different combinations of values for ”magic” features. Therefor we use this example as a confirmation that row correlation based features combined with the information about the processing stations of the component (in this case processed through S32 and S33) represents a strong feature in our model. There are about 20 features extracted from the data that give small improvement on the score based also on correlation between rows and distance features but after sorting by mean time, max time etc.

The most important features are based on row correlation on numerical features after sorting the complete dataset by ID and ”Start Time”. We compute 16 features, 8 sorted by ”Start Time”, and 8 sorted by ID. Correlation is computed

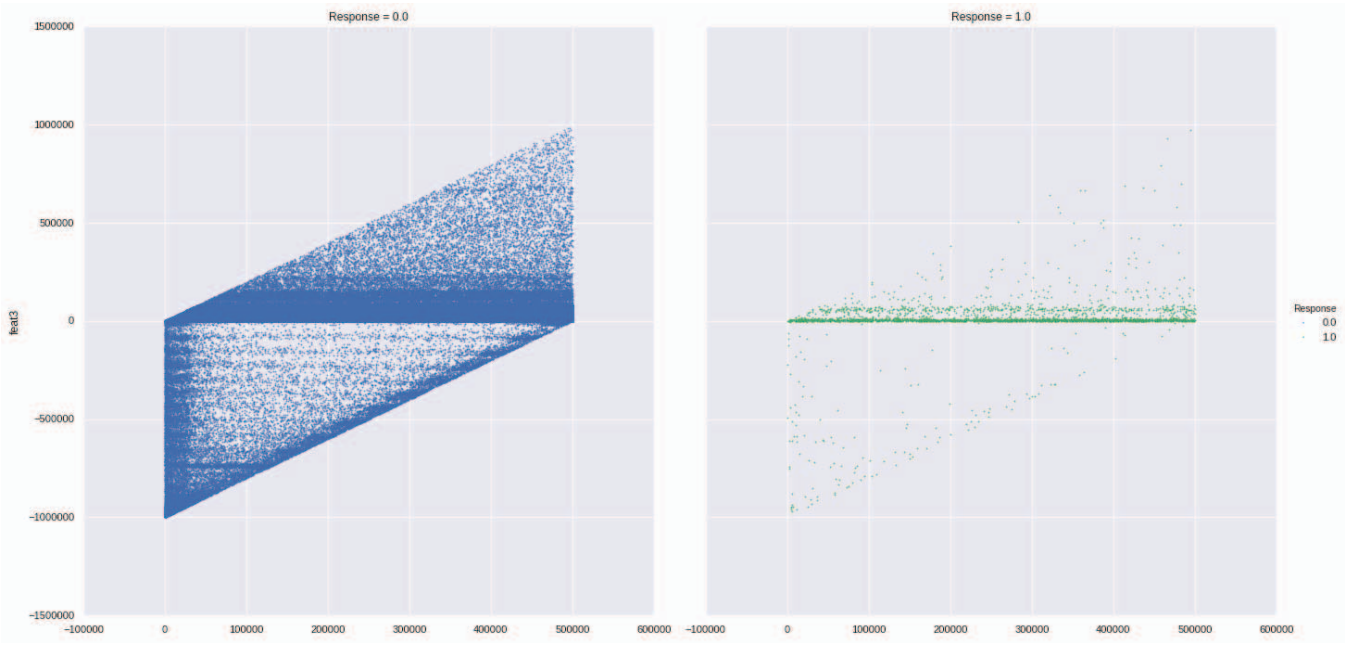

Fig. 2. Sorting time and Euclidean difference between IDs (time distance) for feature 3

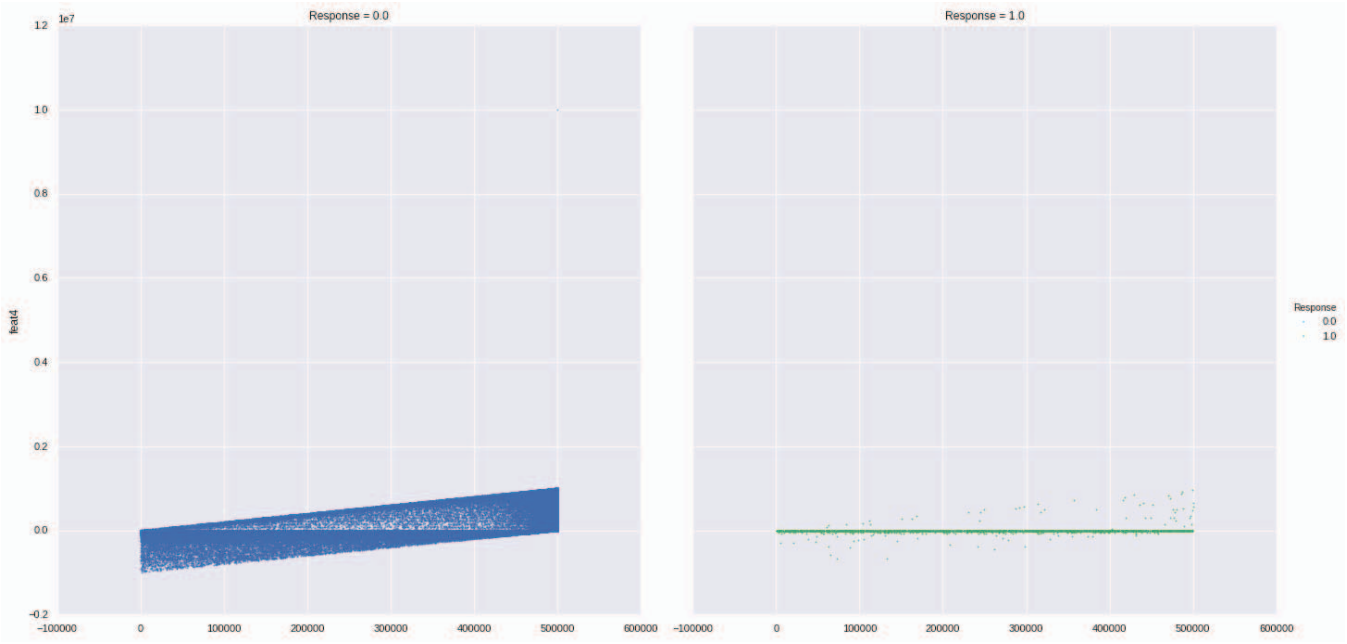

Fig. 3. Sorting time and Euclidean difference between IDs (time distance) for feature 4

between Nth row and $N-1$, $N-2$, $N-3$, $N-4$, $N+1$, $N+2$, $N+3$, and $N+4$. So for each row we get 16 new columns. These features can catch duplicates or similarity between samples (rows) effectively. For example, let's say we have 4 samples, where the first two are duplicates along all features, the third differs only in three features and the fourth is unique (all features are different compared to the three previous samples). The correlation coefficient will be 1 on full duplicates, something less, but still very close to 1 for the samples that differ only in 3 columns, and a very small value for the last sample, since it's unique.

IV. Xtreme Gradient Boosting as Machine Learning Model

A. Model

Nowadays, the Gradient Boosting Machine (GBM) [7] is a technique that shows constantly good results in diverse application field [8]. XGBoost is used for supervised learning problems, where we use the training data X to predict a target

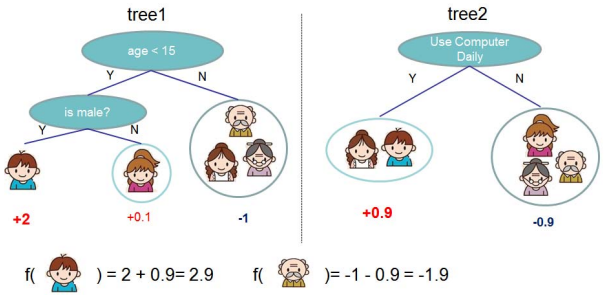

Fig. 4. Tree Ensemble Model. The final prediction for a given example is the sum of predictions from each tree

variable Y. First, we need to introduce the XGBoost model: tree ensembles. This model consists of a set of classification and regression trees. As explained in the original paper [5], if we have input which is some family member characteristics (age, gender, etc.) and we want to predict if particular person plays computer games. Classification and regression tree would assign real score to each leaf of the tree, as shown on Fig. 4. Usually, a single tree is not strong enough to be used in many simple or complex problems, so tree ensemble is used instead. This model is utilized by both Random Forest and Boosted Trees algorithms. The crucial step is the the procedure of training of the model, that is explained in detail in [5].

The XGBoost implementation has several advantages over original GBM implementation such as:

- regularization which prevents over-fitting;
- parallel processing (we know boosting is sequential process, each tree can be built only after the previous one, but in XGBoost single tree can be made by using all the cores),
- it allows high flexibility so the user can define custom optimization objective and evaluation criteria,
- the model can handle missing values,
- it has built-in cross-validation at each iteration so it is easy to get the optimum number of boosting iterations, and
- at last but not least we can save the model and use it in future for predicting.

The algorithm offers several hyper-parameters for tunning and performance optimization. The most important parameters are: maximum depth of a tree, learning rate, number of estimators (the number of boosting stages to perform), subsample (number of samples being used at random for each tree), colsample by tree (number of features being used at random for each tree) and a few more. For detail about all parameters available in XGBoost, we refer the reader to [5].

B. Feature Importances

Feature importance scores are created based on the number of times a variable is used as splitting point in tree in XGBoost [9], [5], weighted by the squared improvement to the model as a result of each split, and averaged over all trees. Or, it represent the frequency of feature occurring in the model trees. In Table IV we can see the 14 most important features. We used this technique for feature selection. Not all features appear in the feature importance table (from above 4000 features as given input to the algorithm, there are only 754 features that XGBoost model has used. So simply we remove them and just retrain our model on these 754 features. This helps in reducing run time and memory needed to perform computation

TABLE IV
TOP 14 FEATURES

importance feature	name
0.013889	corplus1
0.013056	Time_S33
0.012361	corplus2
0.011319	time difference
0.008819	Time_S1
0.00875	L3_S33_F3857
0.008611	corminus1
0.008542	MaxTime4
0.008333	MaxTime3
0.008264	Magic4
0.008194	Time_S32
0.007708	L3_S33_F3865
0.007639	L3_S30_F3759
0.007292	L3_S33_F3859

TABLE V
FOUR CATEGORIES OF DATA TO SPLIT THE TRAIN DATASET

Category	Selection condition
Category 1	$+1/-1$ ID distance and duplicate
Category 2	$+1/-1$ ID distance but not duplicate
Category 3	no sequential ID and not duplicate
Category 4	no sequential ID but is duplicate

C. Extreme Gradient Boosting Training

From the extracted features and after using feature importances function from the algorithm to determine the most important features we trained our model on 754 features. The important thing we did not mention before is that based on two features ID distance and whether the sample is duplicate or not, we split the train data into 4 categories as shown in Table V.

Since our model cannot learn and predict all four categories in the same time, we will create different models for each of the four defined categories. After this segmentation, we can notice that samples from category 3 and category 4 have 0 True Positives (see Table VI) To improve the prediction model and to reduce the noise on category 1 and 2, we define separate models from the data in these two categories and predict the test set only on categories 1 and 2 and we fix the model response on data from category 3 and 4 to 0. The values of the parameters after the tuning process are presented in Table VII.

V. RESULTS

We optimized our model using the MCC as a performance measure, as previously explained. The cross-validation procedure we perform is based only on samples from categories

TABLE VI

RESULTS PER CATEGORY FOR A MODEL WITH A TOTAL MCC SCORE OF 0.433 ON THE LEADER BOARD

Category	TP	FP	FN	TN
Category 1	724	225	2653	87186
Category 2	1062	630	925	30362
Category 3	0	36	1487	1022638
Category 4	0	2	28	35789

TABLE VII

PREDICTION MODEL PARAMETERS

Parameter	Value
max depth	10
eta	0.005
objective	'binary:logistic'
subsample	0.9
colsample by tree	0.7

1 and 2. The cross-validation method we have implemented is five fold stratified cross validation. This method splits the train data into five random chunks that have uniform distribution of fail events across the folds and then alternatively uses for of this chunks for training and the fifth for testing.

For training and cross validation process we used 64GB RAM machine with medium class processor. The time needed for training the models is 24 hours, and we need additional 48 hours for cross validation. The cross-validation procedure is configured to use 20.000 estimators and parameter early stopping rounds equal to 1000. The parameter early stopping round means that the training process will stops if there is no score improvement in number equal to early stopping rounds.

The final model we have developed model stops the training procedure at iteration 1127 with the results as:

$$\begin{aligned} test - MCC: &\quad 0.518675 + 0.00999965 \\ train - MCC: &\quad 0.741047 + 0.00564555 \end{aligned} \tag{2}$$

and threshold used = 0.27 (for selecting each class out of the probability given by the model) which is computed based on the cross validation. Then we use the threshold information and the number of estimators to train on all samples. At the end, as a final result we have 2838 detected failed parts from the test data, which calculated as performance criteria (MCC) equals 0.48446 on private test data and 0.49421 on public data. The models shows small of over-fitting, which could be easily neglected from the perspective of learning algorithms for systems without any detailed knowledge.

VI. CONCLUSION

In this work, we have presented a procedure for tackling problems with large datasets containing only limited information and description of the same. This presented dataset is large (more than million samples), with rare occurrence of failure (less than 1%). This makes the problem computationally complex for machine learning implementation. Nevertheless, using standard statistical and machine learning techniques, we managed to identify some key characteristics, necessary to detect clusters in the dataset. Then the train data is split into clusters and separate models are implement to predict data for each cluster. The results show that it is possible to successfully predict the failure of a component at an assembly line even in situation where no additional details are available.

As future work on the presented problem, one could obviously try to exploit categories 3 and 4 even further. Another possibility is to try and extract more information form the categorical features available, but this will drastically increase the computational burden. At the end, we think that the best way to improve the predictions is to generate new innovative ensemble models.

REFERENCES

[1] Kaggle. (2016) Bosch production line performance. [Online]. Available: https://www.kaggle.com/c/bosch-production-line-performance

[2] A. Mangal and N. Kumar, "Using big data to enhance the bosch production line performance: A kaggle challenge," *CoRR*, vol. abs/1701.00705, 2017. [Online]. Available: http://arxiv.org/abs/1701.00705

[3] B. Pavlyshenko, "Machine learning, linear and bayesian models for logistic regression in failure detection problems," *CoRR*, vol. abs/1612.05740, 2016. [Online]. Available: http://arxiv.org/abs/1612.05740

[4] B. Matthews, "Comparison of the predicted and observed secondary structure of {T4} phage lysozyme," *Biochimica et Biophysica Acta (BBA) - Protein Structure*, vol. 405, no. 2, pp. 442 – 451, 1975. [Online]. Available: //www.sciencedirect.com/science/article/pii/0005279575901099

[5] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," *CoRR*, vol. abs/1603.02754, 2016. [Online]. Available: http://arxiv.org/abs/1603.02754

[6] Belluga. (2016) Date exploration (6 min == 0.01). [Online]. Available: https://www.kaggle.com/ gaborfodor/bosch-production-line-performance/ notebookd19d11e4f2

[7] J. H. Friedman, "machine." *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, oct 2001. [Online]. Available: http://projecteuclid.org/Dienst/getRecord?id=euclid.aos/1013203451/

[8] P. Li, "Robust logitboost and adaptive base class (ABC) logitboost," *CoRR*, vol. abs/1203.3491, 2012. [Online]. Available: http://arxiv.org/abs/1203.3491

[9] P. Li, C. J. C. Burges, and Q. Wu, "Mcrank: Learning to rank using multiple classification and gradient boosting," in *Proceedings of the 20th International Conference on Neural Information Processing Systems*, ser. NIPS'07. USA: Curran Associates Inc., 2007, pp. 897–904. [Online]. Available: http://dl.acm.org/citation.cfm?id=2981562.2981675