Assignment: Advanced Machine Learning and Computational techniques for Industry 4.0

LUIS GASCÓ, PhD in Engineering, Spain

This report contains the result of the technical work done for the postdoctoral research position in Advanced Machine Learning and Computational techniques for Industry 4.0. In this document we explain the decisions taken to generate a prediction model for the problem presented in Kaggle's challenge entitled "Bosch Production Line Performance" whose purpose is to reduce the number of manufacturing failures in an industrial process.

CCS Concepts: • **Computing methodologies → Machine learning**.

Additional Key Words and Phrases: Industry, Data Science

ACM Reference Format:
Luis Gascó. 2020. Assignment: Advanced Machine Learning and Computational techniques for Industry 4.0. 1, 1 (September 2020), 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The fast development and rise of technologies such as artificial intelligence and the Internet of Things (IoT), together with the decrease in cost and improvement of computing capacities have fostered the beginning of a revolution in the industrial sector, popularly known as Industry 4.0. The vast majority of companies are interested in this revolution, because it allows them to improve their production systems to decrease costs and improve the quality of the final products, as well as being more competitive with respect to the rest of the companies in their respective sectors.

As it occurred in other fields of research, such as city management [1], after researching a lot on the improvement of sensors and the ecosystems to access data, researchers have started developing intelligent systems to exploit this data through machine learning techniques that can improve different aspects such as the detection of defective products [2, 3] or the optimization of production processes [4, 5].

In this document we present the results of a first approach to the analysis of data from industrial environments based on the data from the Bosch Production Line Performance Kaggle challenge [6]. The objective of the document is to demonstrate our knowledge to continue within the selection process for the postdoctoral position offered by the Universidad Politécnica de Madrid.

The rest of the manuscript is organized as follows. In Problem, we describe the problem faced in the challenge. In Data, we do a preliminary analysis of the dataset and show some of the results of those analyses. In Methods, the processes to reduce the feature space and the models trained are shown. In the last section, we review the results and we give some possible improvements.

Author's address: Luis Gascó, luisgascosanchez@gmail.com, PhD in Engineering, Madrid, Spain.

XXXX-XXXX/2020/9-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 THE PROBLEM

The challenge we face is to design a system that allows us to predict the production failures of components in an assembly line from measurements made to each of them. For this purpose, the manufacturer Bosch provides a wide set of data (14.3GiB) in which each component is associated with a set of numerical, categorical and temporal features, as well as a binary value that indicates whether the component had a manufacturing failure or not. Since we have a set of features for each component associated with an output value (or label), we are dealing with a supervised learning problem. In particular, the challenge can be seen as a classification problem, in which from a feature vector input and an associated label, we must train a model able to detect failure in new elements not seen before by the model.

3 DATA

As mentioned above, we are dealing with a large dataset. The training dataset has almost 1.2 million elements, which have to be used to train a model capable of classifying the test dataset, which also has almost 1.2 million elements as well.

Each element is associated with a set of 4265 numerical,categorical and time-related features . The sparsity of the dataset is high: the numeric dataframe has 80.9% missing values, the categorical matrix has XX% , and the one with date information has a 82.2%. Furthermore we find a pronounced imbalance of classes: there are 1176868 elements that were manufactured correctly and just only 6879 products in which any manufacturing defect was detected. The marked class imbalance problem will make us face a challenge during the model training step. Furthermore, the large volume of data will be difficult to allocate in memory, and the large number of features could produce bias problems in the final model.

In terms of the preliminary analysis of the numerical data, we find 968 features for each element. The name of each feature provides information about where in the production chain the value of the feature was recorded. The feature names follow this convention:

$$Lx\_Sx\_Fxxxx \tag{1}$$

where L and S are the line and station number where the feature was acquired, and F represents the feature number.

After analyzing this dataset we find 4 production lines and 51 stations. The graph presented in the Figure 1-Top represents the number of features measured at each of the stations. It can be seen that the number in each of the stations is not uniform, being particularly high in stations 24,25, 29 and 31. In relation to the number of elements processed in each station, Figure 1-Middle shows an inequality in each of the stations, being those of Line 3 the ones that process the largest number of elements. Finally, the percentage of failed elements processed by each station is less than 1% in all cases, except in the case of station 32 which exhibits 4.5%. Despite that, this anomaly is unlikely to have an effect on a future model given the low number of elements processed by this station. This can be easily verified when we analyze the same parameters aggregated by manufacturing line, as shown in Figure 2 where it is visually verified that even though the error rate is higher on line 3 it will not have a big effect because fewer elements are processed.

Regarding the categorical dataset, there are 2140 features for each element. One approach to work with categorical data in ML is the use of the one-hot-encoding technique [7], which transforms a categorical value with k distinct elements into k binary columns that can be processed by a ML model. If we use this approach, we would have a really large number of feature columns making it difficult to train the model with the memory limitations of a conventional PC (16/32GB), so we decide to follow another approach to work with this variables while reducing the dimensionality of the feature space.

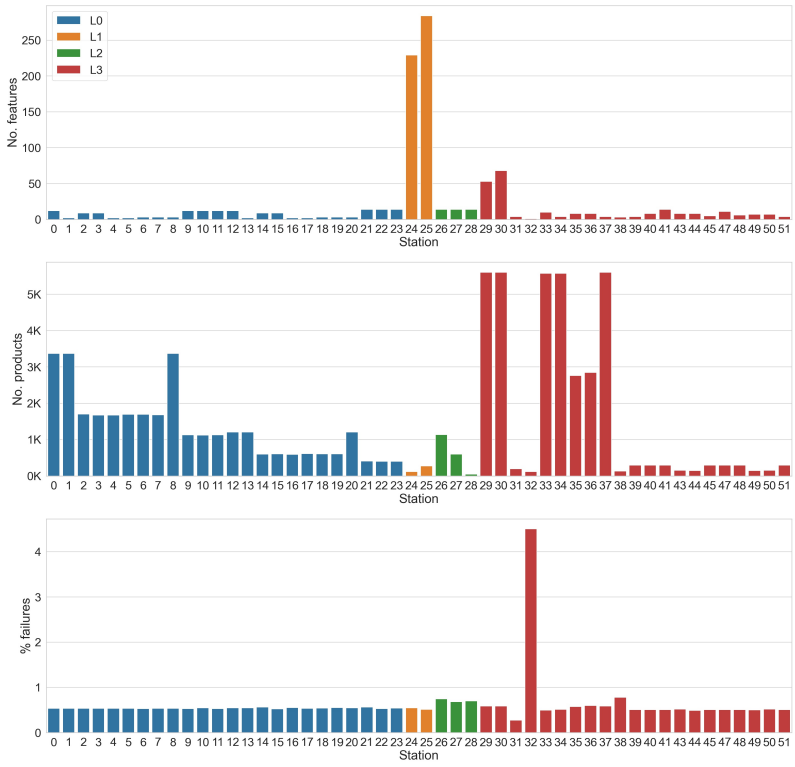

Fig. 1. Top - Number of features measured by station. Middle - Number of products by station. Bottom - Percentage of failures detected by station

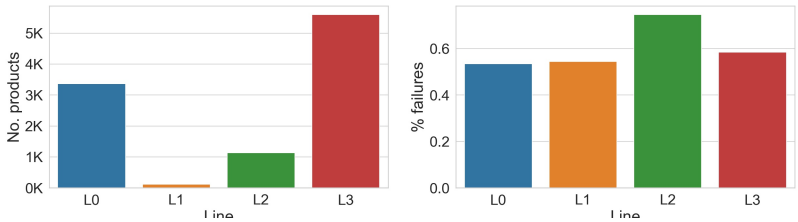

Fig. 2. Top - Number of products by station. Bottom - Percentage of failures detected by station

Finally, the time-related dataset has 1157 different features. These data have a periodic pattern as demonstrated by the winners of the Bosch Kaggle competition in the paper they published afterwards [8] (and shown in the Figure 3). But we decided not to make any feature engineering with this data due to the nature of this technical task, where other challenges related to memory usage and a large number of features are present and should be resolved.

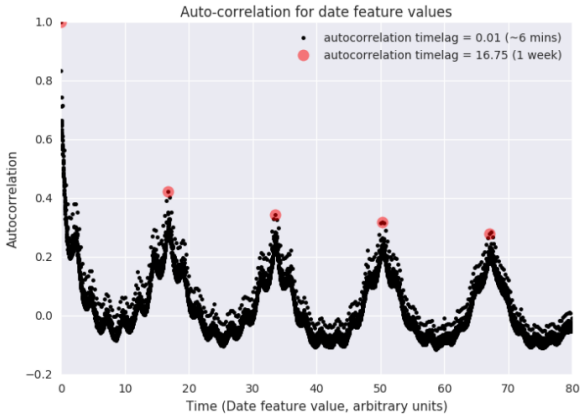

Fig. 3. Autocorrelation in the date dataset [8]

4 METHODS

In the previous section, we analyzed the nature of the dataset. We had a classification problem with a dataset of heterogeneous nature, in which no information is provided about the nature of the numerical characteristics (i.e., it is not specified whether they are continuous or discrete) and with categorical information. The main limitations are that the different dataframes present a great sparsity, about $\approx 81\%$ of the dataset are missing values; all features are anonymized, making feature engineering even more complicated; and, we have to work with a large amount of data that cannot be allocated in memory without problems in the training step. Considering these limitations, we will try to solve this classification problem with models based on decision trees, which show good behavior with data of this type since there is no need to normalize the data, they can work with both categorical and numerical features, and they are less memory (and time) demanding than other algorithms such as SVMs [9].

We followed the workflow shown in the scheme in Figure 4. First, we concentrate our efforts on reducing the feature space to optimize the use of PC memory through feature selection processes. Once the most important features were selected, we trained a final model to be used to classify the test data set.

4.1 Feature space reduction

4.1.1 Categorical features. As shown in the previous section, there are a large number of categorical features. If we were to apply one-hot-encoding to each of the features, we would have several thousand more columns in the dataset, something that would make the training of the models difficult. For that reason, the following method has been used to reduce the number of categorical features in the final classifier:

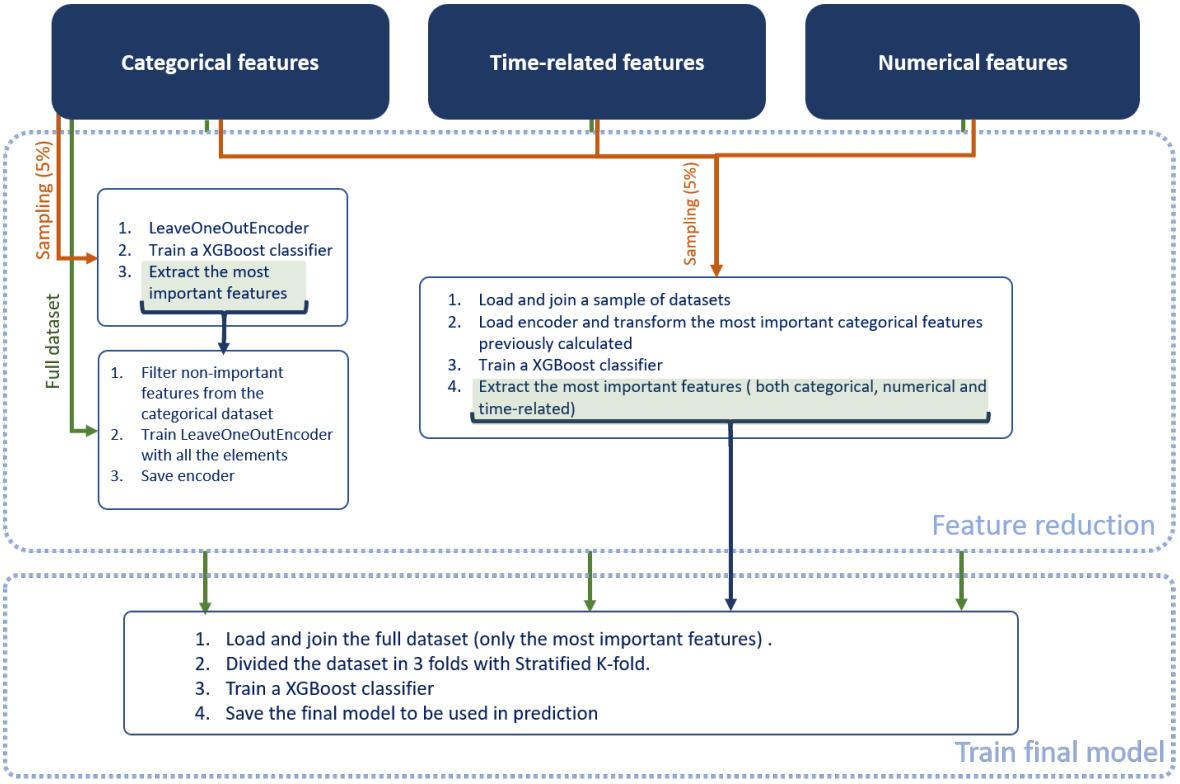

Fig. 4. Workflow of the system designed

(1) We randomly selected the 5% of samples of the categorical dataset, but maintaining the label distribution[1].
(2) We applied the method *Leave One Out Encoder*. This method is similar to the *Target Encoder* proposed by Micci-Barreca [10], where each category from each feature is replaced by a calculation of the posterior probability of the target given the category and the prior probability of the target on the training data. We choose the Leave One Out Encoder because it works better with outliers [2].
(3) We trained a XGBoost model with all the transformed categorical variables and we selected the k most important ones (the variables that more time were been selected for splitting the trees of XGBoost)[3].

Then, after obtaining a reduced set of the 75 most important categorical variables, we had to retrain the enconder to work properly with any dataset value. To do this, the categorical dataset was fully loaded, but filtering by the most important columns previously extracted to save memory. Afterwards, using the complete set of elements, the encoder was retrained and saved for use in future steps.

4.1.2 Rest of the features. In the previous step we were able to reduce the number of categorical features from 2140 to 75. Despite this, we still have a total of 2200 features when we include the temporal (1157) and numerical

[1]The idea was to do it with the 20%, but we had no time enough to train the encoder
[2]Link to the resource: http://contrib.scikit-learn.org/category_encoders/leaveoneout.html
[3]We could also have a feature selection by measuring the correlation between the transformed features and taking only the ones with a correlation lower than 0.95

(968) variables. If we were to train the model with that large feature set, we would risk overfitting it during training due to the high dimensionality. Therefore we trained an intermediate XGBoost model with 5% of samples[4]. Once again, before the training step we transformed the categorical variables to their probabilistic version through the previously generated *Leave One Out Enconder*, and after the training we extracted a set of the 150 features that were most important in explaining the output.

4.2 Classifier

Once those 150 features were determined, we reduced the feature space enough to fit the entire data set in memory and have more memory space to train the final model using it. Unfortunately due to temporal constraints, we had to limit the size of the training dataset to a 40% of randomly selected samples. We would also like to mention that we wanted to test and tune several decision tree based classifiers such as Random Forest, AdaBoost, Gradient Tree Boosting, and Extreme Gradient Boosting, but due to the temporal constraint we only had time to tune the hyperparameters for the last one, an Extreme Gradient Boosting classifier [11].

For tuning the hyperparameters and evaluating the performance of the model, we have divided the dataset into 3 splits using a stratified 3-fold cross validation algorithm, which divides our dataset in 3 parts and uses 2 folds for training and the last fold for testing. Our dataset was highly imbalanced, as the number of failures, labelled as class 1, was lower than the other ones identified as class 0. We decided to use stratification to maintain the original percentage of samples of each class in each split.

Since the dataset is very imbalanced, we optimized the model by means of the *Area Under Curve*, and after the hyperparameter tuning, we got the best results with the following key-pair parameters:

- learning_rate = 0.1
- max_depth = 5
- min_child_weight = 3
- n_estimators = 100

With that parameters we obtain the scores of AUC of: 0.704, 0.688, and 0.686. Obtaining an **overall AUC of 0.693±0.008**. It is also required to define the best value of the Matthew Correlation Coefficient. We have used the model previously obtained with the 3-fold cross validation and we have chosen the optimal threshold of this coefficient, that is 0.246, as shown in the Figure 5

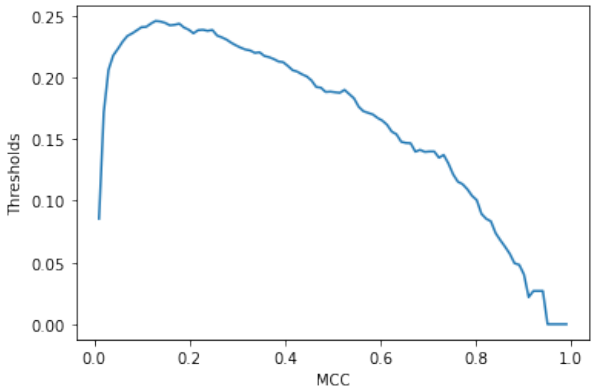

Fig. 5. Matthew Correlation Coefficient graph

[4]Initially we were going to train it with the half of the dataset, but we did not have time enough to train the model

Finally, Figure 6 shows the 10 most important variables in the model. It is noteworthy that the only categorical feature that was selected for the final model is the one that contributes more to the model. The rest of the variables, except for the eighth most important one, are numerical. That is to say that the previous analyses of categorical variable selection have had a valuable final result to improve the model's quality.

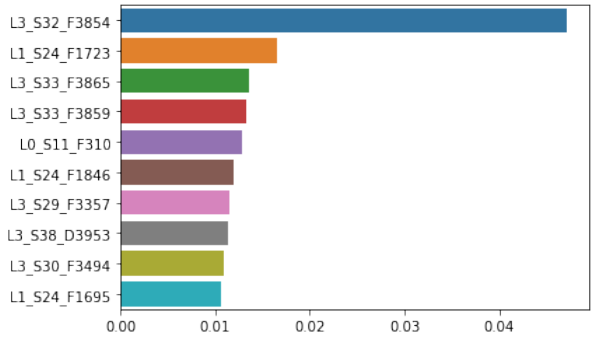

Fig. 6. Feature importance of the final XGBoost model.

5 DISCUSSION

In this report we have shown one of the multiple models that could be trained from the data provided. Despite the fact that the dataset was complicated to analyze, due to the large amount of empty data and the existence of limitations in my PC's memory, we have gotten a final model with an acceptable quality in terms of AUC, that is comparable the models of other Kaggle users, although it could easily be improved if more time were available for features engineering.

It is important to emphasize that after analyzing some of the solutions proposed by other Kaggle data scientist, it has been seen that they have made a great work of feature engineering, especially for the time-related dataset where for example they identify cluster of products and made different models for each cluster. We decided not to implement these features in order to not increase the complexity of the analyses in the reduced available time to do them.

The code that has been used to carry out this work can be found in the following github repository https://github.com/luisgasco/postdoc_technical_task/[5].

REFERENCES

[1] Luis Gasco, Cesar Asensio, and Guillermo De Arcas. Towards the assessment of community response to noise through social media. In *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, volume 255, pages 2209–2217. Institute of Noise Control Engineering, 2017.

[2] JA Carvajal Soto, F Tavakolizadeh, and Dávid Gyulai. An online machine learning framework for early detection of product failures in an industry 4.0 context. *International Journal of Computer Integrated Manufacturing*, 32(4-5):452–465, 2019.

[3] Mustafa Jahangoshai Rezaee, Akram Salimi, and Samuel Yousefi. Identifying and managing failures in stone processing industry using cost-based fmea. *The International Journal of Advanced Manufacturing Technology*, 88(9-12):3329–3342, 2017.

[4] José A Alfaro and Luis A Rábade. Traceability as a strategic tool to improve inventory management: a case study in the food industry. *International Journal of Production Economics*, 118(1):104–110, 2009.

[5]I have only upload notebooks, since I had no time to make a proper set of classes to produce the final model. I would like to produce a docker container with all the data, but the deadline is over. The paths in the files are absolute, so it is necessary to change them before executing in other PC than mine.

[5] Chih-Hung Hsu. Data mining to improve industrial standards and enhance production and marketing: An empirical study in apparel industry. *Expert Systems with Applications*, 36(3):4185–4191, 2009.
[6] Bosch production line performance.
[7] Patricio Cerda and Gaël Varoquaux. Encoding high-cardinality string categorical variables. *IEEE Transactions on Knowledge and Data Engineering*, 2020.
[8] Ankita Mangal and Nishant Kumar. Using big data to enhance the bosch production line performance: A kaggle challenge. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 2029–2035. IEEE, 2016.
[9] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.
[10] Daniele Micci-Barreca. A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. *ACM SIGKDD Explorations Newsletter*, 3(1):27–32, 2001.
[11] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, and Yuan Tang. Xgboost: extreme gradient boosting. *R package version 0.4-2*, pages 1–4, 2015.