A Hierarchical XGBoost Early Detection Method for Quality and Productivity Improvement of Electronics Manufacturing Systems

Alexandre Gaffet[1,2], Nathalie Barbosa Roa[1], Pauline Ribot [2,3], Elodie Chanthery[2,4] and Christophe Merle[1]

[1] *Vitesco Technologies France SAS 44, Avenue du Général de Croutte, F-31100 Toulouse, France*
alexandre.gaffet@vitesco.com
nathalie.barbosa.roa@vitesco.com
christophe.merle@vitesco.com

[2] *CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France*
elodie.chanthery@laas.fr
pauline.ribot@laas.fr

[3] *Univ. de Toulouse, UPS, LAAS, F-31400 Toulouse, France*

[4] *Univ. de Toulouse, INSA, LAAS, F-31400 Toulouse, France*

Abstract

This paper presents XGBoost classifier-based methods to solve three tasks proposed by the European Prognostics and Health Management Society (PHME) 2022 conference. These tasks are based on real data from a Surface Mount Technologies line. Each of these tasks aims to improve the efficiency of the Printed Circuit Board (PCB) manufacturing process, facilitate the operator's work and minimize the cases of manual intervention. Due to the structured nature of the problems proposed for each task, an XGBoost method based on encoding and feature engineering is proposed. The proposed methods utilise the fusion of test values and system characteristics extracted from two different testing equipment of the Surface Mount Technologies lines. This work also explores the problems of generalising prediction at the system level using information from the subsystem data. For this particular industrial case: the challenges with the changes in the number of subsystems. For Industry 4.0, the need for interpretability is very important. This is why the results of the models are analysed using Shapley values. With the proposed method, our team took the first place, capable of successfully detecting at an early stage the defective components for tasks 2 and 3.

1. Introduction

The 2022 PHME Data challenge encourages participants to solve multiple classification problems for a real production line from Bitron Spa. The dataset includes data from Solder Paste Inspection (SPI) and Automatic Optical Inspection (AOI) equipment of a real industrial production line equipped with automated, integrated and fully connected machines (Industry 4.0). A detailed description of the dataset is given in 3. The challenge is to design an algorithm to predict test labels for the components. Specifically, the goal is to develop a hierarchical classification predicting: 1. whether the AOI classifies the component as defective; 2. in the case of a defect, the label applied by the operator; 3. in the case of confirmation of the defect by the operator, the repair label.

To tackle this challenge, we pursue the following steps: data exploration and domain knowledge extraction, data cleaning, data preparation (normalization and encoding), data modelling (model training and validation) and results in analysis. The four last steps were made recursively while trying different approaches, as shown in Section 4. Data exploration allowed us to identify three main issues within the given dataset: missing information, highly imbalanced classes (for all tasks) and high cardinality on the categorical features. The latter is not necessarily an issue but implies that a special treatment needs to be done to these features *a priori*. We will elaborate on the issues in Section 4.1 and on the categorical encoding in Section 4.2.

To solve each task, we take different information units formed by feature tuples, corresponding to different levels on the data hierarchy. At the same time, different features are kept as relevant for each task and followed by specific normalization or encoding. The specific tools used for each task are described in detail in Section 4.3. Finally, after presenting the experimental setup used to tune the model hyperparameters (Section 4.4), the achieved results are discussed in Section 5.

2. Related work

Several scientific articles already present machine learning applications for Surface Mount Technology production lines. (Richter, Streitferdt, & Rozova, 2017) proposes a convolution neural networks deep learning application working on the AOI system to automatically detect defects. In (Tavakolizadeh, Soto, Gyulai, & Beecks, 2017), some binary classifiers are tested to detect defects inside products using simulated production data. These data are simulated from several SMT lines and give a good classification score. In (Parviziomran, Cao, Yang, Park, & Won, 2019), a component shift prediction method is proposed to predict the shift of the pad during the reflow process. In (Park, Yoo, Kim, Lee, & Kim, 2020) SPI data are used to predict at an early stage the potential defects of the prediction. This work is based on a dual-level defect detection method. In (Jabbar et al., 2018) some tree-based machine learning methods are used to predict the defects found in AOI using SPI data. (Gaffet, Ribot, Chanthery, Roa, & Merle, 2021) proposes an unsupervised univariate method for monitoring the In-Circuit Testing machine (located at the end of the Surface Mount Technology lines) and components. Another large topic of interest for this study is prognosis and health management at different levels: system-level or sub-system level. In our case, we have to use information from a pin level to retrieve the health at a system level which is the product component. This topic is linked to the decentralized diagnosis approach. (Zhang, 2010) proposes a decentralized model-based approach with a simulation example of automated highway systems. (Ferdowsi, Raja, & Jagannathan, 2012) proposes a decentralized fault diagnosis and prognosis methodology for large-scale systems adapted for aircraft, trains, automobiles, power plants and chemical plants applications. (Tamssaouet, Nguyen, Medjaher, & Orchard, 2021) proposes a component interaction-based method to provide the prognosis of multi sub-system model.

3. Data description

The PHME provides the dataset used in this article as part of the 2022 conference data challenge (PHM Society, 2022). The dataset includes measurement information from two different steps of the PCB production (see Figure 1). The first step is the SPI, in which each solder pad is checked to verify its compliance and, accordingly, a sanction is generated depending on the quality of the solder (evaluated on several physical aspects). The second step is the AOI. In addition to checking the finished solder pads, their position, shape, etc., this process also inspects the component itself, looking for defects like missing or misaligned components. The AOI inspection has two types of sanctions: the automatic sanction provided by the machine itself and the one given by an operator that verifies the first one in case of spotted defects.

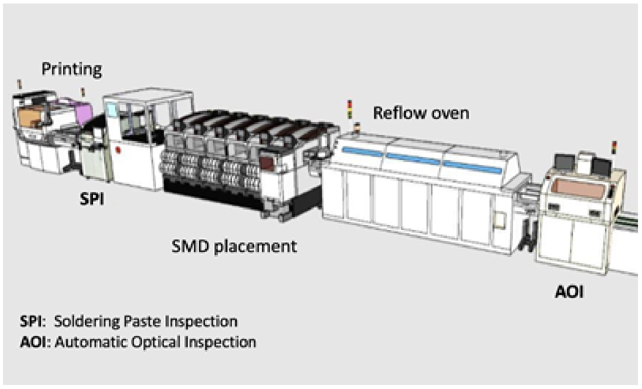
Printing
Reflow oven
SPI
SMD placement
AOI
SPI: Soldering Paste Inspection
AOI: Automatic Optical Inspection
Figure 1. Surface Mount Technologies Production Line. (PHM Society, 2022)

Table 1. Summary characteristics of the datasets

Feature	SPI	AOI
Number of lines	5 985 382	31 617
Number of panels	1924	1 924
Number of components	129	102
Figures/panel	{1,8}	{1,...,8}
Components/panel	{128,129}	{2,...,27}
Lines/panel	{3112,...}	{2,...,203}

Simple data exploration allows to discover anomalous entries and clean the datasets. A summary of the information found in each dataset is shown in Table 1. The cleaned SPI dataset is composed of 1921 panels, each with 3112 entries. A panel is an ensemble of 8 PCBs, also called figures, grouped. Each figure is composed of 129 components. The component reference is found on the feature `ComponentID`. These components have several pins that can be identified using the `PinNumber` feature. The SPI test results provide the volume, area, height, size, shape and offset of each `PadID` as well as a final result flag. A `PadID` corresponds to a unique combination of {`FigureID, ComponentID, PinNumber`}. Each panel provided by the competition has at least one component detected as a defect by AOI automatic sanction. Each line of the datasets describes a `PadID` of one electronic board.

For the AOI, each line corresponds to a unique entry of the set {`PanelID, FigureID, MachineID, ComponentID, PinNumber`}, where the `PinNumber` can be fill as `NaN`. On such cases, we believe the AOI does not inspect the solder paste but the component itself.

As introduced before, the challenge is divided into three tasks.

Task 1 is to predict whether or not a component will be classified as defective by the AOI using only the inputs provided by the SPI, i.e. the pad measurements. Task 2 is about predicting the operator's label using the SPI test results and the automatic defect classification provided by the AOI (`AOIlabel`). Finally, task 3 concerns the reparation operation. Again, the objective is to predict whether the component detected as a defect by the operator can be repaired or not. For this task, the information used for the prediction is the SPI test result, the `AOIlabel` and also the `OperatorLabel`.

4. Methodologies

4.1. Challenges

The exploratory analysis of the training data has revealed several issues that need to be tackled to solve correctly the different tasks:

1. Missing values: for three different `PanelID`, the proposed data missed some information in the SPI dataset. We choose to exclude the lines with no information.
2. Class imbalance: the number of components detected as defects by the AOI is much lower than the number of components from the SPI dataset. Similarly, the number of components really classified as a fault by the operator is much lower than the number of components classified as correct.
3. High cardinality of the categorical features: the categorical features `PadID` have more than 1000 modes. Without any sort of variable encoding, classifiers are very difficult to use for such variables. `PadID` is already encoded in the sort because the values are ordered by `FigureID`. In a way, the variable is encoded by component area.
4. High bias in continuous features: the continuous features such as volume, area, height... are highly correlated to the categorical feature `PadID`.
5. Level of prediction: the prediction has to be done at a component level, whereas the available data are given at a `PadID`. This leads to a lot of issues in creating the target training and prediction. Indeed, for instance, it is unclear if the target training has to be created by component or by pad.
6. Different numbers of pins: the number of pins depends on the component. It is difficult to use all the pin results as input of a classifier for each component, as the number of pins and therefore features varies. The generalisation of the training depending on the component is very difficult.

4.2. XGBoost algorithm and categorical features encoding

For tabular data applications, Gradient Boosting Decision Trees (GBDT) are widely used, being XGBoost (Chen et al., 2015), CatBoost (Prokhorenkova, Gusev, Vorobev, Dorogush, & Gulin, 2018) and LightGBM (Ke et al., 2017) the algorithms with the best results. Among these algorithms, we decide to use XGBoost, which is a scalable, paralleled and distributed implementation of the original gradient boosting tree algorithm. GBDT is an ensemble model algorithm, i.e. it combines several decision trees to perform a better prediction than a single model. XGBoost uses, in particular, the idea of boosting: it uses a collection of weak models to generate a strong model. In practice, for XGBoost, the idea is to use a gradient descent algorithm over a cost function to iteratively generate and improve weak models. On each iteration, a new weak decision tree is generated based on the error residual of the previous weak model. The final prediction is a weighted sum of all the iterated weak trees. Among ensemble methods, boosting can minimize the model's bias. We propose one model for each task. In this section, these models as well as the used features are presented.
XGBoost is a very performing algorithm, although some caution has to be taken when categorical features are used. This is true for all tree-based or boosted tree methods. In particular, one hot encoding can lead to very poor results when the categorical features have many levels. Indeed, a large number of levels leads to sparsity as for each level, a new variable is created. These new variables have only a small fraction of data points that shall have the value 1 and the other the value 0, which is a problem for tree-based methods because tree split searching for the purest nodes. Indeed, a hot encoded variable is not very likely to lead to the purest nodes if it is very sparse. That is why, the tree split will not be done using this one hot encoded variable. Even if the original categorical feature has a lot of importance for the prediction. In our cases, other encoding techniques should be used.

First, a common technique is the Hash encoding. It is already present in our dataset with a numerical value for each `PadID` level. The `PadID` level values depend on the `FigureID` of the pad. Here the Hash encoding is done with only one feature but in general, it could be encoded into more features. One of the most used hashing methods is described in (Yong-Xia & Ge, 2010).

The next approach is the frequency-based encoding method: it is based on the frequency of the levels as the label value. If the frequency is linked to the target, it will help the prediction of the variable. For instance, in tasks 2 and 3, the frequency of one component is probably linked to the issues that can exist for each component. This encoding can be useful in that case.

Finally, the last type of encoding is label-based. The idea of label encoding is to replace each categorical value with the conditional probability of the class to be predicted by knowing the categorical features. This can be done by several methods such as Leave One Out Encoding, and CatBoost encoding (Prokhorenkova et al., 2018). The main issue with this method is to learn the conditional probability without overfitting. It can be realized by not taking the observation into account in the learning of the probability for each observation, as in the Leave One Out Encoding. This can also be done more efficiently using CatBoost encoding. For this case, we found that the CatBoost encoding performs best.

The frequency-based encoding and Hash encoding have less success.

4.3. Solving the challenges

Task 1

The first task is the most challenging of all. The main difficulty arises from the fact that some defects are related to the pin, and others to the component itself. In our opinion, the most important question for each task is “Should we predict (model) by component or by pin ?”. For this first task, we decide to go for a per-pin prediction. This is mainly guided by the difficulty to generate coherent labels and features at the component level for this task. Indeed, only one pin can have an issue. It does not seem right to affect the same label to a component with only one pin detected as a defect by AOI equipment and another component with multiple pins with defects. Moreover, the number of pins varies too much depending on the studied component. As a result, any aggregation of continuous variables will probably hide important information if only one pin has a defect. For instance, the aggregation with the mean of the `Volume` will not contain a lot of information if there are many pins, and only one defective pin for the considered component.

For each tuple, we want to predict if the tuple is detected as a defect by the AOI equipment or not. Accordingly, the training target column is 1 if the tuple appears in the AOI dataset and 0 otherwise. It is worth noting that we are not considering as defective tuples for which only `PanelID, FigureID, ComponentID` appear with `PinNumber = NaN`. Both, categorical and continuous features, are used as input. We use the following continuous variables: `Volume(%),Area(%),OffsetX(%), OffsetY(%),Shape(um),PosX(mm), PosY(mm),SizeX, SizeY` that we simply call “numerical features” in the following of the article. As a categorical value, we only keep `ComponentID` that we encode using a CatBoost encoding method (Prokhorenkova et al., 2018).

Task 2

For this second task, we first split the AOI dataset into two parts according to whether or not `PinNumber = NaN`. In the case where `PinNumber` is specified, we can join the AOI and the SPI easily using the columns `PanelID, FigureID, ComponentID, PinNumber` in each dataset. From the joint dataset, we use the “numerical features” from the SPI test results as defined in task 1. For the categorical features, we keep three features: `AOILabel, ComponentID` and `FigureID_ComponentID` where the later is the string concatenation of the variables with the same name. For these features, we use a CatBoost encoding based on the categorical encoders python library. We believe a better result can be achieved with deeper work on the optimization of the encoder hyper-parameters. Finally, we also create two new meta-features (not encoded) `Count_Pin_Component` and `Count_Pin_Figure`. These two variables are counting the number of pins detected as a defect by the AOI respectively for the component and figure of the tuple `PanelID, FigureID, ComponentID, PinNumber`. The XGBoost classifier algorithm classes each tuple into the class “Bad” or “Good” of the `OperatorLabel` target column.

For the AOI defect without any `PinNumber` associated, we propose to also use an ensemble of categories and continuous variables. We use the same categorical and meta features, while for the numerical features we use the mean values per component of the following variables: `Volume(%), Area(%), OffsetX(%), OffsetY(%)` to keep the information only on the `PanelID, FigureID, ComponentID` tuple level.

Finally, we also use an XGBoost classifier algorithm to class each tuple `PanelID, FigureID, ComponentID, PinNumber`, with `PinNumber` referenced as NaN as described before.

For the final sanction (that must be given at the `PanelID, FigureID, ComponentID` three-tuple level), we use the following rule: if one of the four-tuple `PanelID, FigureID ComponentID, PinNumber` entry is predicted as “Bad”, then the associated three-tuple will also be considered as “Bad”. If not, the label “Good” is assigned to the three-tuple.

Task 3

Task 3 is about the prediction of one categorical value presented in the AOI dataset `RepairLabel`. This label can take two values, `FalseScrap` or `NotPossibleToRepair`. For this task, we tried the same approach as in task 2 predicting each four-tuple `PanelID, FigureID, ComponentID, PinNumber` and merging the results per component, but the approach was not successful. To improve the result, we choose to do the prediction for the `PanelID, FigureID, ComponentID` three-tuple directly. As input, we use the same method as for task 2, grouping the SPI values per three-tuple using the mean as an aggregation method for the “numerical features”. The categorical variables used are `ComponentID, FigureID_ComponentID`. As before, these variables were encoded using the CatBoost encoding method.

The meta-features generated are `Count_Pin_Component`, `Count_Pin_Figure` and `Count_Pin_Panel`, respectively, the number of pins detected as defects by the AOI per component, figure and panel. We also created one-hot encoded features from the `AOILabel`. For each labelled type of error, the associated feature has the value of 1 if the three-tuple is detected as having this error by the AOI machine (in at least one pin) and 0 otherwise. To predict the class of each tuple, we use the XGBoost classifier.

4.4. Hyper-parameter tuning

The XGBoost model has been tuned using the Optuna python library (Akiba, Sano, Yanase, Ohta, & Koyama, 2019). This package is an optimization framework that searches for the best hyper-parameters of the space defined by the user. It uses some distributed computation and early stopping to improve the speed of the solution. It is implemented with various optimization algorithms. In our case, we use the sampling algorithm Tree Parzen Estimator Sampler (TPES) (Bergstra, Bardenet, Bengio, & Kégl, 2011). It is a sequential model-based optimization. As a bayesian optimization algorithm, it computes a probability model of the optimization function and selects the best hyper-parameter according to this probability model and the real cost result. For each step, the tuning is done using the mean of the F1-score of a 4 cross-validation method. The dataset is split into four parts, each of these parts is recursively the testing dataset while the other is used for training the model. For the sake of reproducibility, hyper-optimization techniques and training algorithms are available in (Gaffet., 2022). Our prediction can probably be easily improved with more iterations of the tuning phase, as we did not spend much time on it.

5. Results and Discussion

The results obtained for the three tasks are detailed in this section.

5.1. Score

Table 2. F1-score for the three tasks with training and testing data set

Dataset	**Task 1**	**Task 2**	**Task 3**	**Score**
Training	0.43	0.66	0.90	0.66
Testing	0.41	0.67	0.77	0.62

Table 2 shows the F1-score for each task of the challenge. The training and testing set results are close, showing good generalization capability. It seems that our models avoid overfitting issues that are difficult to handle with this dataset. Indeed, the imbalance issue and the fact that some variables such as `ComponentID` have a lot of importance can lead to large bias.

5.2. Feature importance

Figure 2 presents the feature importance of the classification model for task 1. The most important features are the component's position on the panel and the encoded `ComponentID` variable. Because almost 30 per cent of the defects found in the AOI dataset are coming from one component ("BC1"),

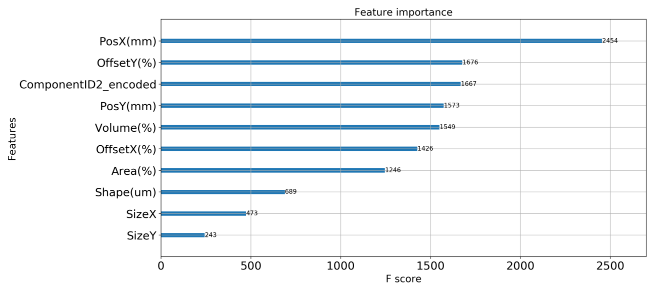

Figure 2. Task 1: feature importance of the XGBoost classifier

.

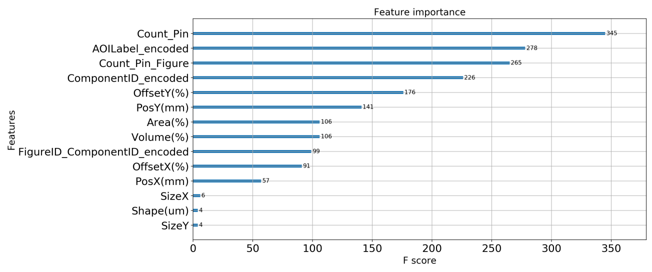

Figure 3. Task 2: feature importance of the XGBoost classifier for the four-tuple (AOI defect is linked to a pin)

this creates a bias in the model results that depends on `ComponentID` features.

Figure 3 and Figure 4 present the feature importance for task 2 on the four-tuple and three-tuple classifiers respectively. The position and the `ComponentID` are also important for this task, but the continuous variables `Volume(%)`, `OffsetX(%)`, `OffsetY(%)` and `Shape(um)` seem to have also a great impact on the prediction. Even more, the generated meta-features `Count_Pin_Component`, representing the number of pins with a defect per component, and `Count_Pin_Figure`, representing the number of defective pins per figure, have also a large impact. Actually, these two features allow improving the F1-score for this task by almost 0.2.

Figure 5 shows the feature importance of the XGBoost clas-

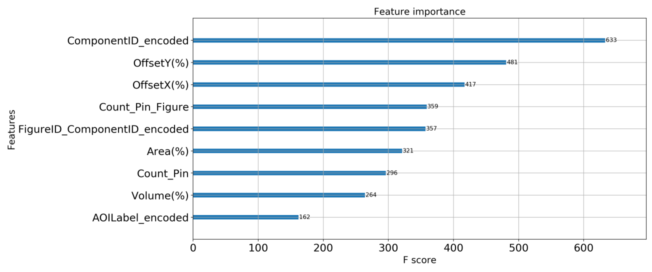

Figure 4. Task 2: feature importance of the XGBoost classifier for the three-tuple (AOI defect is not linked to a pin)

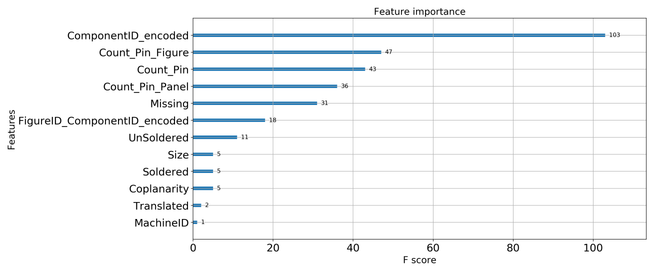

Figure 5. Task 3: feature importance of the XGBoost classifier

sifier for task 3. Without surprise, given the previous results, `ComponentID` is shown as the feature with the biggest importance value. `ComponentID` could be thought of as a domain-specific key factor, if, for example, depending on the type of component, the reparation is possible / allowed or not. For instance, if there is an issue with a microchip, this is far more difficult and costly to solve than an issue with a simple resistance. The following most important features are the meta-features being in the order of importance more critical the number of defective pins per figure, then component and final panel. This also seems correct as the number of issues increases the potential damage to the product. The “Missing” `AOILabel` (one-hot encoded) also has a considerable impact. Maybe it is not possible to replace a missing component due to oven operation.

5.3. SHAP values

The interpretation of the machine learning model described as a black block model is a really important topic in the industry. Indeed, for the acceptance of a model, it is mandatory to explain the model decision to the process experts. The interpretation allows validating the model by comparing the model and experts’ explanation of a phenomenon. It also allows recommending some repair actions to the experts. SHAP (SHapley Additive exPlanation) interpretation (Lundberg & Lee, 2017) is based on the game theory (Štrumbelj & Kononenko, 2014). The idea is to compute an interpretation value called SHAP value and denoted ϕ_j for all variables and each sample of the dataset. The output of the model is described by the sum of the SHAP values ϕ_j.

$$\phi_j = \sum_{S \subseteq J \setminus \{j\}} \frac{|S|!(M - |S| - 1)!(f(S \cup \{j\}) - f(S))}{M!} \tag{1}$$

where M is the number of variables, J is the ensemble of variables, f the model output and j the variable index. SHAP is an additive method. The output of a classifier can be described as the cumulative sum of the impact of all variables.

Figure 6, Figure 7, Figure 8, Figure 9 present the computed impact of the features on the model output. For task 1, a

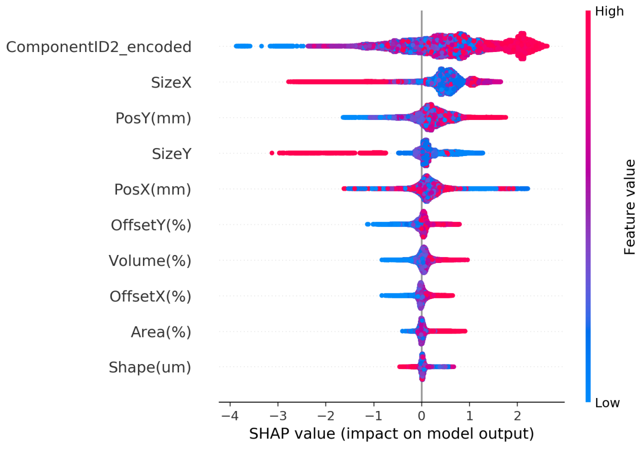

Figure 6. Task 1: SHAP value of the XGBoost classifier

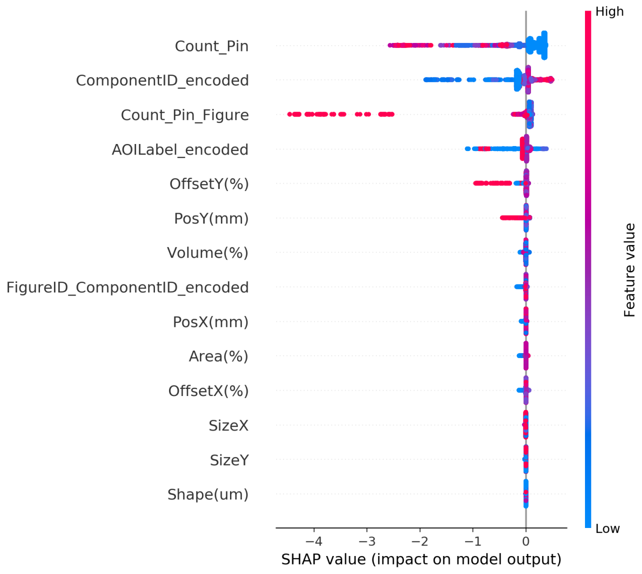

Figure 7. Task 2 for the four-tuple (AOI defect is linked to a pin): SHAP value of the XGBoost classifier

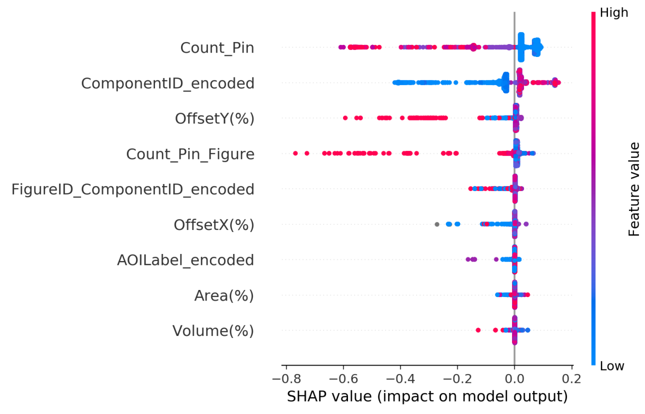

Figure 8. Task 2 for the three-tuple (AOI defect is not linked to a pin): SHAP value of the XGBoost classifier

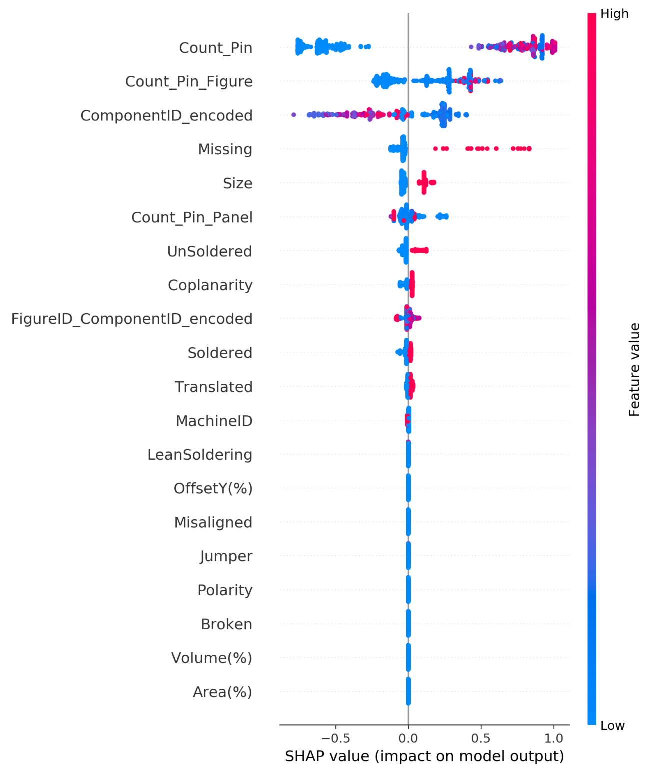

Figure 9. Task 3: SHAP value of the XGBoost classifier

		Predicted Label	
		Defects	Not Defects
True labels	Defects	8263	23345
	Not Defects	4204	1937814

Table 3. Confusion matrix for all training sets

		Predicted Label	
		Defects	Not Defects
True labels	Defects	8256	1064
	Not Defects	4203	1922

Table 4. Confusion matrix for the training set observation of the component “BC1”

larger value of the CatBoost encoded variable `ComponentID` logically results in larger model output (the encoding is done by replacing the level value with its conditional probability of target). The variable `SizeX` does not seem to be connected to the model output. A more interesting larger value of the continuous variable of `Volume` and `Offset` seems to be representative of a defect for task 1. For task 2, a low number of defective pins per component and per figure is more likely to be a “Good” `OperatorLabel`. This result was expected as the more the AOI equipment found defective pins, the more likely a real defect in the component is. The impact of continuous variables is very low for this task. For Task 3, the output value 1 is associated with the “Not Possible to Repair” label and the output value 0 with “FalseScrap”. The SHAP values indicate that the more the number of defective pins per component per figure, the more likely the product is impossible to repair. If a component or a pin is missing, then the product seems also more likely to be not possible to repair. The same result is found for “Unsoldered” `AOILabel` level. For the other level of `AOILabel`, the results are unclear.

5.4. Issues with Task 1

In this subsection, the issues encountered in the prediction of task 1 are detailed. As previously reported, the results of the prediction of task 1 are highly biased by one component “BC1”. Table 3 and Table 4 present the confusion matrix results of the training of task 1. It can be seen that most of the correctly detected defects by the classifier model of task 1 is coming from the results of the component “BC1”. Only five other tuples are correctly classified for the defects. Obviously, this is a concern for our method. Another observation is that the F1-score obtained with the results coming from Table 4 is 0.76 whereas the F1-score obtained considering all the tuples as defects is 0.75. Both scores are very close and depending on the cross-validation random selection, the simple rule considering all the tuples of the the“BC1” component as defects can be better than a more advanced classifier. To solve this issue, more advanced analyses are required. From our experience, it seems easier to predict the defects associated with the “Unsoldered” `AOILabel` than the other.

AOILabel / Score	Score Task 1
Coplanarity	0.00
Translated	0.00
Soldered	0.00
Unsoldered	0.55
Size	0.00
LeanSoldering	0.27
Misaligned	0.15
Missing	0.00
Jumper	0.00

Table 5. Task 1 score results for different AOILabel levels

6. Conclusion and Future work

To solve the different challenges in the Printed Circuit Board production, we proposed three different methods. These methods use the XGBoost classifier and are based on variable encoding and feature engineering. We have shown that generating meta-features representing the circuit state at different levels (component, figure and panel) improved the generalization of the classifiers. The use of different encoding techniques for the categorical features has shown CatBoost as the most promising one, due to the high cardinality issues. The first task results are probably too biased to be used in production. However, the results of task 2 and task 3 are promising for a production application.

In future work, it could be interesting to add some information to improve the prediction. The time between operations for instance is missing (probably to not share cycle time information). A product with a larger cycle can lead to an issue with dust deposit, humidity change... This information could improve the performance of all tasks. Some extra information like the temperature curves of the oven can also help to better predict future issues. Finally, for task 1, unsupervised monitoring methods may be a better solution due to the class imbalance issue.

References

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th acm sigkdd international conference on knowledge discovery & data mining* (pp. 2623–2631).

Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, *24*.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., … others (2015). Xgboost: extreme gradient boosting. *R package version 0.4-2*, *1*(4), 1–4.

Ferdowsi, H., Raja, D. L., & Jagannathan, S. (2012). A decentralized fault detection and prediction scheme for nonlinear interconnected continuous-time systems. In *The 2012 international joint conference on neural networks (ijcnn)* (pp. 1–7).

Gaffet., A. (2022). *Phme data contest 2022.* https://github.com/alexandregft/PHME-data-contest. GitHub.

Gaffet, A., Ribot, P., Chanthery, E., Roa, N. B., & Merle, C. (2021). Data-driven capability-based health monitoring method for automative manufacturing. In *Phm society european conference* (Vol. 6, pp. 12–12).

Jabbar, E., Besse, P., Loubes, J.-M., Roa, N. B., Merle, C., & Dettai, R. (2018). Supervised learning approach for surface-mount device production. In *International conference on machine learning, optimization, and data science* (pp. 254–263).

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., … Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, *30*.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in neural information processing systems*, *30*.

Park, J.-M., Yoo, Y.-H., Kim, U.-H., Lee, D., & Kim, J.-H. (2020). D 3 pointnet: Dual-level defect detection pointnet for solder paste printer in surface mount technology. *IEEE Access*, *8*, 140310–140322.

Parviziomran, I., Cao, S., Yang, H., Park, S., & Won, D. (2019). Data-driven prediction model of components shift during reflow process in surface mount technology. *Procedia Manufacturing*, *38*, 100–107.

PHM Society. (2022). *Data challenge.* (Sponsored by Bitron Spa. Published electronically at https://phm-europe.org/data-challenge)

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, *31*.

Richter, J., Streitferdt, D., & Rozova, E. (2017). On the development of intelligent optical inspections. In *2017 ieee 7th annual computing and communication workshop and conference (ccwc)* (pp. 1–6).

Štrumbelj, E., & Kononenko, I. (2014). Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, *41*(3), 647–665.

Tamssaouet, F., Nguyen, K. T., Medjaher, K., & Orchard, M. E. (2021). Online joint estimation and prediction for system-level prognostics under component interactions and mission profile effects. *ISA transactions*, *113*, 52–63.

Tavakolizadeh, F., Soto, J., Gyulai, D., & Beecks, C. (2017). Industry 4.0: mining physical defects in production of surface-mount devices.

Yong-Xia, Z., & Ge, Z. (2010). Md5 research. In *2010 second international conference on multimedia and in-*

formation technology (Vol. 2, pp. 271–273).

Zhang, X. (2010). Decentralized fault detection for a class of large-scale nonlinear uncertain systems. In *Proceedings of the 2010 american control conference* (pp. 5650–5655).