Prediction of Production Line Status for Printed Circuit Boards

Haichuan Tang[1], Yin Tian[2], Junyan Dai[3], Yuan Wang[4], Jianli Cong[5], Qi Liu[6], Xuejun Zhao[7], Yunxiao Fu[8]

[1,2,6,7,8] *CRRC Academy, Beijing, 100161, China*
thc@crrc.tech
ty@crrc.tech
lq@crrc.tech
zxj@crrc.tech
fyx@crrc.tech

[3]*Rutgers University, New Brunswick, NJ, 08854, USA*
jd1394@scarletmail.rutgers.edu

[4]*Southern University of Science and Technology, Shenzhen, Guangdong, 518000, China*
wang.skoud@gmail.com

[5]*Southwest Jiaotong University, Chengdu, Sichuan, 610031, China*
jlcong2019@my.swjtu.edu.cn

ABSTRACT

This paper focuses on the problem of predicting production line status for Printed Circuit Boards (PCBs). The problem contains three prediction tasks regarding PCB producing process. Firstly, data exploration is carried out and it reveals several data challenges, including data imbalance, data noise, small sample size, and component difference. To predict production line status for components of PCBs using records of inspection on pins, we proposed two possible feature extraction methods to compress the pin-level data into component-level. A statistical feature extraction method, which retrieves descriptive statistics such as mean, standard deviation, maximum, and minimum of pins on the same component, is applied to Task 1, while a PinNumber-based feature extraction method, which keep original values for 2-pin components, is applied to Task3. In addition, a neural-net model with feeding imbalance control is established for Task 1. and a random forests model is applied for both Task 2 and Task 3. Moreover, a threshold moving technique is proposed to optimize the threshold selection. Finally, the result shows that our models achieved f1-scores of 0.44, 0.54, and 0.71 on the test set for the three tasks, respectively.

1. INTRODUCTION

The PCB is a platform installed with semiconductor chips, capacitors, and other components, providing electrical interconnection between components. It is used in virtually all electronic products. Accurate prediction of PCB production line status can effectively reduce the production cost. During the production line, the PCB goes through the printing machine, solder paste inspection (SPI), surface mount device placement, reflow oven, and automatic optical inspection (AOI) in sequence. The production line contains two inspections: the SPI and the AOI. The SPI measures the shape of the solder paste and detects possible problems with the solder paste after it is placed by the printing machine. The AOI checks defects that might occur after the PCB goes through surface mount device placement and reflow oven. The AOI produces three labels as follows:

1) AOI label: indicating the type of defects detected by the AOI machine.
2) Operator label: assigned by the human operator, indicating whether the AOI machine raised a false defect or not.
3) Repair label: assigned by the repairment operator, indicating the repair action.

There are many benefits for predicting the above-mentioned production status. For example, by accurately predicting operator label and repair label, operators can improve their efficiency by arranging their work orders based on the prediction results. The problem of predicting the above three labels using SPI and AOI data is posed in PHME 2022 Data Challenge (PHME Data Challenge, 2022).

Predicting production line status is a challenging problem due to its extremely imbalanced dataset, where the population of the normal status data is far more than that of the faulty data. As imbalanced data is common in real world problems, many methods have been proposed by prior studies to deal with it. Data resampling is often considered as the first choice to solve the data imbalance problems (Batista, et al., 2004). The data resampling methods include randomly oversampling the minority class, randomly undersampling the majority class, and other advanced approaches such as the synthetic minority oversampling technique (Chawla, et al., 2002) and adaptive synthetic sampling (He, et al., 2008). However, resampling the data may lead to a worse model for some reasons: 1) undersampling may discard useful information and 2) oversampling may increase the chance of overfitting and the learning time. Another approach to deal with imbalanced data is moving decision thresholds of a learned model. This approach has been implemented and tested as a better alternative for resampling (Provost, 2000; Maloof, 2003).

The problem of PHME 2022 Data Challenge is even more challenging in that the proposed models need to consider not only the high imbalance ratio, and noise in the dataset, but also the different numbers of pins for different components, which means specific feature extraction is needed to unify the feature structure.

In our solution, we propose a machine learning-based method to predict production line status using data from previous production steps. A statistical feature extraction and a PinNumber-based feature extraction method are proposed to reconstruct pin-level data into component-level data to perform predictions on PCB components. The statistical feature extraction method retrieves descriptive statistics such as mean, standard deviation, maximum, and minimum values of pins on the same component. The PinNumber-based feature extraction method treats PCB components differently based on the number of pins they contain. This is inspired by the data exploration that most of the PCB components have 2 pins. To deal with the imbalanced dataset, we introduce a neural network model with feeding imbalance control and a random forests method with a threshold moving technique. As a result, our proposed method achieved f1-scores of 0.44, 0.54, and 0.71 on the test for the three tasks posed by this data challenge.

The rest of the paper is organized as follows: the problem definition, datasets, and scoring functions are described in Section 2. Data exploration and preprocessing methods are provided in Section 3. In Section 4, classification methods are introduced. Results are discussed in Section 5 and conclusions are drawn in Section 6.

2. Data Challenge Description

2.1. Problem Definition

The data challenge focuses on predicting labels produced by the AOI. As the AOI produces three labels, the problem is divided into three tasks as follows:

1) Task 1: Predict whether any defects in PCB components will be detected by the AOI machine.
2) Task 2: Predict whether the AOI machine will raise a false defect according to the human operator.
3) Task 3: Predict the repair label assigned by the repairment operator.

Task 1 and Task 2 are binary classification problems, whereas Task 3 is the only multi-class classification problem in this data challenge.

2.2. Datasets

The datasets of this challenge were collected from the SPI and AOI. After removing the data with a null value, the SPI data contains 5,985,381 records on 1,969,523 components, while AOI data contains 31,617 records on 27,514 components. This is because a component may have several pins used for soldering. For a detailed description of the datasets, please refer to https://phm-europe.org/data-challenge.

2.3. Scoring

In this challenge, the F1-score is used to evaluate the model performance. The F1-score is calculated based on the ground truth and the predicted labels. The relating functions are listed as follows:

$$F1_l = 2 \cdot \frac{precision_l \cdot recall_l}{precision_l + recall_l} \tag{1}$$

$$precision_l = \frac{TP_l}{TP_l + FP_l} \tag{2}$$

$$recall_l = \frac{TP_l}{TP_l + FN_l} \tag{3}$$

Where TP represents the true positive, FP represents the false positive, FN represents the false negative, and the sub-notation *l* denotes the positive class. The specific scoring function for each task is listed in Table 1. Task 1 considers the components in the AOI dataset as the positive class. Task 2 considers the "Bad" operator label as the positive class. In Task 3, which is a multi-class classification problem, the average of the F1-scores using "NotPossibleToRepair" and "FalseScrap" as positive classes respectively is calculated. The final score is the average of the F1-scores computed from the three tasks.

Table 1. Scoring function for each task

Task No.	Scoring Function
Task 1	$F1_{inAOI}$
Task 2	$F1_{Bad}$
Task 3	$(F1_{FalseScrap}+F1_{NotPossibleToRepair})/2$

3. Data Exploration and Preprocessing

In this section, we explore the original dataset to find a potential design basis for the data preprocessing methods as well as the prediction models. According to our exploration, we propose two feature extraction methods to reduce the data redundancy and construct a proper data frame to be used as model input.

3.1. Data Exploration

To find the correlation between each two data columns, we plot two-dimensional dot charts using one column for the x-axis and the other for the y-axis. An example of the dot charts is shown in Figure 1. The x-axis represents the value for "Height(um)" in SPI data, while the y-axis denotes the "Volume(um3)" value. In Figure 1, the dots constitute multiple linear patterns, which we assume are caused by different component characteristics. Thus, we group the components into 14 types based on the letters in the ComponentID. For example, ComponentIDs {"BC1", "BC2", "BC3", "BC4"} belong to the component type "BC". The different colors in Figure 1 indicate the component types, which validates our assumption.

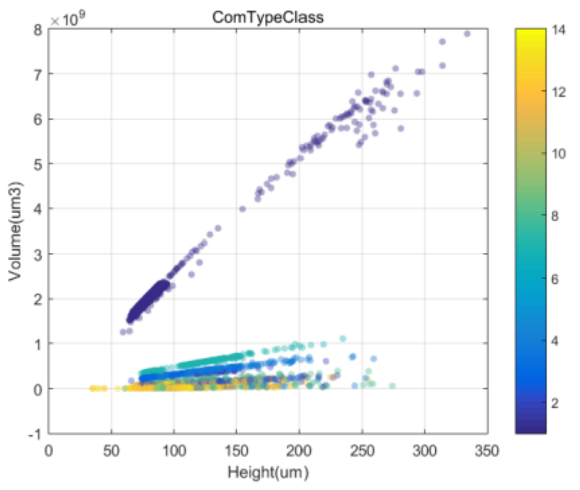

Figure 1. Scatter plots of Volume(um3) and Height(um) with component types in different colors.

Since this challenge focuses on component-level prediction when the original datasets stores pin-level records, we need to convert the original records to component-level features. Figure 2 below shows the frequency distribution of the number of pins in each component in the SPI dataset, where 2-pin components are the majority.

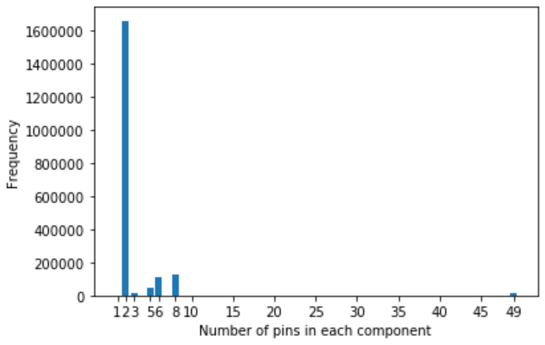

Figure 2. Frequency distribution of the number of pins in each component.

For task 2, the distribution of 'OperatorLabel' concerning PosX/Y is presented in Figure 3.

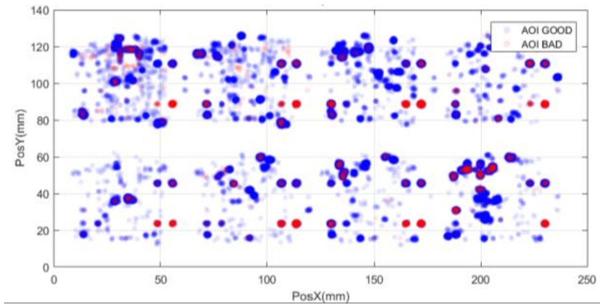

Figure 3: The 'OperatorLabel' distribution with respect to PosX/Y.

Some essential features of the dataset are summarized as follows:

- **Highly imbalanced**. The ratio between the majority and minority classes is quite high, 250:1 for Task #1 for example.
- **Quite noisy**. There is a lot of noise inside the dataset.
- **Small sample size**. For Task #2 and #3, the data sample sizes are quite small.
- **Component difference**. There are 14 different types of components and the number of pins of the components varies from 2 to 49. Also, it can be found that different components may have very different feature distributions, and are related to a different defect probability.

3.2. Statistical Feature Extraction

SPI dataset contains measurements regarding welding paste characteristics (i.e., volume, height, area, offset, size) for every pin. To retrieve features for components, we group all measurements of pins in the same component and compute their statistical features (i.e., mean, standard deviation, minimum, maximum). Based on the data exploration above, we also consider the component type and the number of pins as two important features. The pseudo-code for merging pin-level records into components and the statistical feature extraction algorithm is shown in Algorithm 1.

Algorithm 1: get_stat_features(*spi*)

```
1  com_ids = get_unique_com_ids(spi)
2  rows = empty list
3  for id in com_ids:
3      panel_id = id[0]
4      figure_id = id[1]
5      com_id = id[2]
6      spi_temp = spi[(spi.PanelID==panel_id) &
   (spi.FigureID==figure_id) &
   (spi.ComponentID==com_id)]
7      means = get_mean(spi_temp)
8      stds = get_std(spi_temp)
9      max_min = get_max(spi_temp) – get_min(spi_temp)
10     com_type = get_comtype(com_id)
11     com_pin_num = len(spi_temp)
12     rows.append([panel_id, figure_id, com_id,
   com_type, com_pin_num, means, stds, max_min])
13 return rows
```

3.3. PinNumber-based Feature Extraction

As shown in Figure 2, most components in the SPI dataset contain 2 pins. Thus, we keep all original measurements for 2-pin components to ensure that no information regarding the majority is discarded. For the components with only 1 pin, we duplicate the measurements to form a unified structure. For the rest components with more than 2 pins, we adopt the maximum and minimum values for each measurement as follows:

$$\begin{cases} \hat{X}_{c,f,1} = \min\limits_{p\in[1,P_c]}\left(X_{c,p,f}\right) \\ \hat{X}_{c,f,2} = \max\limits_{p\in[1,P_c]}\left(X_{c,p,f}\right) \end{cases} \quad \forall c \in \{C\}, \forall f \in \{F\} \qquad (3)$$

Where c represents a specific component, {C} is the complete set for all components during PCB manufacture, p is the pin number, P_c is the max pin number for a given component c, f is one of the numerical features, {F} is the complete set of numerical features, $X_{c,p,f}$ is the original numerical feature with given component and pin number, $\hat{X}_{c,f,1}$ and are two new numerical features for a given component to replace original ones.

The pseudo-code for merging pin-level records into components and pin number-based feature extraction algorithm is shown in Algorithm 2.

Algorithm 2: get_pinn_features(*spi*)

```
1  com_ids = get_unique_com_ids(spi)
2  rows = empty list
3  for id in com_ids:
3      panel_id = id[0]
4      figure_id = id[1]
5      com_id = id[2]
6      spi_temp = spi[(spi.PanelID==panel_id) &
   (spi.FigureID==figure_id) &
   (spi.ComponentID==com_id)]
7      com_pin_num = len(spi_temp)
8      if com_pin_num == 2:
9          rows.append(spi_temp[0]+spi_temp[1])
10     else if com_pin_num > 2:
11         rows.append(get_max(spi_temp) +
   get_min(spi_tem[1]))
12     else:
13         rows.append(spi_temp[0]+spi_temp[0])
14 return rows
```

4. Classification Methods

4.1. Neural Network with Feeding Imbalance Control

The neural-net structure is given in Figure 4. It is a typical forward architecture with multiple dense nets and there is a SoftMax layer before the final output. The key points are given as follows:

- **Categorization**. All continuous features are converted into categorical features according to percentiles division. The percentiles are set as [0, 1, 5, 10:10:90, 95, 99, 100]. This operation will somehow compress the information of some continuous features, such as height(mm) and area(mm). This can be taken as a denoise approach.
- **Binarization**. All categorical features are converted into binary vectors according to one-hot coding and then merged into one binary vector. The label is also encoded by one-hot coding.
- **Training by Feeding Imbalance Control**. According to Section 3.1 Data exploration, we know that the dataset is extremely imbalanced. When training a neural net to fit an imbalanced dataset, we need to control the imbalance ratio of each mini-batch during the training process. As a result, we proposed the concept of Feeding Imbalance Ratio (FIR), which is an implementation of an under-sampling approach.

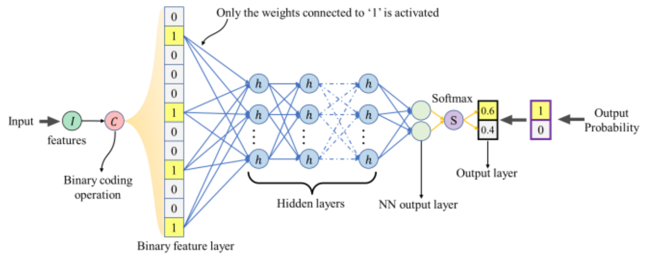

Figure 4. Forward Architecture of Neural-net Model for Probability Prediction

Note that the AOI defect is a rare event, the overall occurrence probability of an AOI defect is about 0.4%. The ratio between non-event and event is around 250:1. To enhance the performance of the model, we design the training process in a particular way. Firstly, we use the stochastic gradient descent (SGD) method to train the model. Secondly, during the training process, instead of feeding the data randomly, we feed the data with a constraint on the ratio between the majority and minority. We define Feeding Imbalance Ratio as follows:

Definition: Feeding Imbalance Ratio (FIR). The ratio between the majority and minority classes within the resampled mini-batches fed into the NN model during the training process.

Notably, FIR is an important parameter for training the model. If FIR is too large, the dataset fed into the model is imbalanced, and it is hard to learn the feature combination related to the AOI defect. FIR is set to 1 in our case. The training algorithm is given in Table 2.

Table 2. Training Algorithm for probability prediction by feeding imbalance control.

```
Input:
✧   $FIR = 1, batch\_size, n\_epoch, learning\_rate$
✧   Training dataset: $(\boldsymbol{F}, \boldsymbol{L})$;
✧   The number of layers and neurons of the neural net;

Initialize:
✧   Initialize a neural-net $\boldsymbol{p}(* \mid \boldsymbol{\theta})$;
✧   Split the $(\boldsymbol{F}, \boldsymbol{L})$ into $(\boldsymbol{F}, \boldsymbol{L})^+$ and $(\boldsymbol{F}, \boldsymbol{L})^-$ according to the label,
        '+' and '-' represent the majority and minority classes,
    respectively.

Main:
For _ in range ($n\_epoch$), do
    $(\boldsymbol{F}, \boldsymbol{L})^+ = (\boldsymbol{F}, \boldsymbol{L})^+.\text{shuffle}()$
    $(\boldsymbol{F}, \boldsymbol{L})^- = (\boldsymbol{F}, \boldsymbol{L})^-.\text{shuffle}()$

    For _ in range (round($size((\boldsymbol{F}, \boldsymbol{L})^-)/batch\_size$)), do
        $(\boldsymbol{F}, \boldsymbol{L})_i^+ = (\boldsymbol{F}, \boldsymbol{L})^+.\text{next\_batch}(batch\_size)$
        $(\boldsymbol{F}, \boldsymbol{L})_i^- = (\boldsymbol{F}, \boldsymbol{L})^-.\text{next\_batch}(FIR * batch\_size)$
        $\boldsymbol{F}_i^+ = onehot(\boldsymbol{F}_i^+)$
        $\boldsymbol{L}_i^- = onehot(\boldsymbol{L}_i^-)$
        $(\boldsymbol{F}, \boldsymbol{L})_i = \text{shuffle}\big(\text{concat}(\boldsymbol{F}_i^+, \boldsymbol{L}_i^+), \text{concat}(\boldsymbol{F}_i^-, \boldsymbol{L}_i^-)\big)$
        Update the parameter $\boldsymbol{\theta}$ of $\boldsymbol{p}(* \mid \boldsymbol{\theta})$ given mini-batch $(\boldsymbol{F}, \boldsymbol{L})_i$
    End For
End For

Output: The trained neural-net $\boldsymbol{p}(* \mid \boldsymbol{\theta})$.
```

4.2. Random Forests

A random forests classifier is an ensemble of tree-structured classifiers. It is an upgraded Bagging algorithm (Breiman, 2001). In Bagging, a bootstrap sample is used to train each weak classifier and the majority vote of the weak classifiers is considered the final prediction. Random Forests further introduces feature randomness to Bagging. Instead of using all features, Random Forests splits each node using a randomly selected subset of features. The Random Forests classifiers are used to solve task 2 and task 3 and are implemented by the scikit-learn python library (Pedregosa et al., 2011).

4.3. Threshold Principles

Since the performance evaluation is based on the F1-score, we have to set a proper threshold for each task once we obtain the continuous output from our model to present our final predicted result set. There are two possible principles for the selection of thresholds:

- **Equal probability**. A threshold is selected assuming that the minority class in the test set has the same occurrence probability as the training set.
- **Best in Training**. A threshold of the test set is selected that performs best in the training set.

Note that the threshold principles work only for Task 1 and Task 2.

5. Analysis of Results

In this section, we apply our methodologies to PHME 2022 Data Challenge. The scores are displayed in Table 3 below. The train scores are averages of the scores computed through a 5-fold cross-validation and the test scores are the final scores of our team shown on the leaderboard. In addition, a detailed analysis for each task with only the best result of our experiments is demonstrated in this section.

Table 3. F-Scores of the proposed methods

	Task 1	Task 2	Task 3	Final
Training data	0.43	0.68	0.83	0.65
Test data	0.44	0.54	0.71	0.56

5.1. Task 1

Task 1 is focused on predicting whether the AOI machine will raise a defect record based on the SPI information of a component. For the feature extraction part, the main challenge is to unify the pin features of different component types.

In our solution, for each component, regardless of the number of pins, we compress the pin-level features ("*Volume(%)*", "*Height(um)*", "*Area(%)*", "*OffsetX(%)*", "*OffsetY(%)*", "*Volume(um3)*", "*Area(um2)*", "*Shape(um)*", "*PosX(mm)*", "*PosY(mm)*") to component-level by introducing three statistical operators: (1) average, (2) standard deviation and (3) maximal-minimal difference. As a result, the whole SPI dataset is converted into an SPI-Com-level dataset, and each unique component has one data sample with 33 features(the other 3 features are "*ComponentID*", "*com_type*", and "*com_pin_num*").

The Neural-net model is initialized according to the parameters given in Table 4. During the training process, the FIR is fixed to 1.0. The convergence curve is presented in Figure 5 (x-log scale). Note that the initial value of the loss function is 0.5, indicating the initial untrained neural-net outputs randomly. Soon after about 10 training iterations, a quick converging trend can be found, and the converging trend slows down as the learning rate decays.

Table 4. Parameters of Neural-net for Task 1.

	Parameter	**Value**
1	Number of neurons	256
2	Batch size	32
3	Learning rate	1e-2 to 1e-4
4	F.I.R.	1.0
5	Training epochs	50
6	Train-test split	70% training and 30% test

The F1-scores of the training and test dataset in terms of different thresholds are given in Figure 6. It can be found that the best F1-score is only slightly larger than the test result. More importantly, the related threshold for the best F1-score is almost the same, which is about 0.976. As a result, conclusions can be drawn as follows:

- The best F1-score of the training and test set is about 0.43, and our model is less likely over-fitted. Note that it may be possible to increase the model scale and do more iterations to achieve better performance.

The best threshold to be selected is about 0.976. More generally, the best threshold can be determined according to the one who performs best in the training dataset.

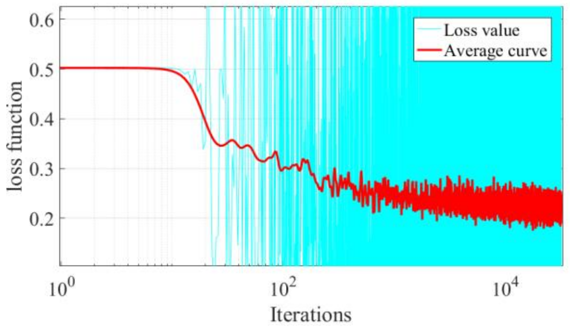

Figure 5: The loss function with respect to training iterations.

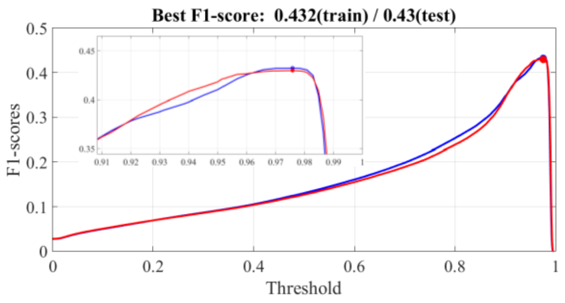

Figure 6: The F1-scores of the training and test dataset.

5.2. Task 2

Task 2 focuses on predicting if the AOI machine will raise a false defect. Thus, we need to remove components that do not exist in the AOI dataset and 27,514 components are left in the training set. Among the 27,514 components, there are 412 components having a "Bad" operator label, and 27,093 components having a "Good" operator label, which is considered highly imbalanced. To tackle the data imbalance issue, we apply a threshold moving technique, best in training, to select the optimized threshold and use it for final testing. Figure 7 validates our threshold moving technique by comparing the predicted test F1-score and the best test F1-score among all possible thresholds. Based on our experiments, a random forests model is built using the selected features: "*com_pin_num*", "*Result*", "*com_type*", "*AOILabel*", and "*MachineID*". 200 n_estimators, 21 max_depth, and 4 min_samples_split are considered the best hyperparameters for our model using grid-search optimization. Note that one-hot encoding is applied for categorical variables.

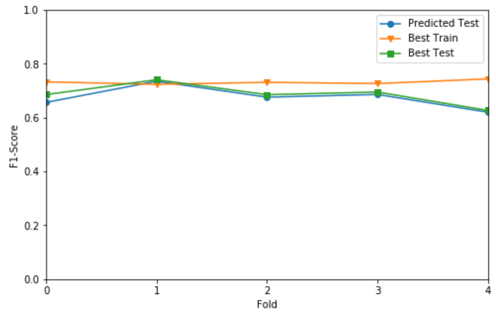

Figure 7. F1-scores for the training set, test set using selected threshold, and the global best f1-score for the test set

5.3. Task 3

Task 3 aims to predict the repair label for only faulty components. Thus, we need to remove components if OperatorLabel is "Good". There are 412 components left in the training set for task 3. Based on our experiments, a combination of the PinNumber-based feature extraction and random forest algorithm produces the best result. The selected features are "*com_pin_num*", "*com_type*", "*Volume(%)*", "*Height(um)*", "*Area(%)*", "*OffsetX(%)*", "*OffsetY(%)*", "*Volume(um3)*", "*Area(um2)*", "*Shape(um)*", "*PosX(mm)*", "*PosY(mm)*", "*Result*", "*AOILabel*", and "*MachineID*". In addition, 300 n_estimators and 7 max_depth are considered the best hyperparameters for our model using grid-search optimization.

Using the optimal model parameters, the model performance on the training set of this multi-classification problem has been presented in the form of a confusion matrix, which is shown in Figure 8.

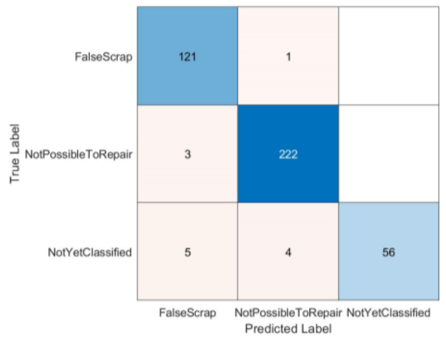

Figure 8. Task 3 confusion matrix using the training set

6. Conclusion

This paper focuses on three prediction tasks regarding the PCB manufacturing process. Firstly, data exploration is carried out and it reveals several data challenges: (1) highly imbalanced data, (2) noisy data, (3) small sample size, and (4) component difference. Secondly, to address these challenges, statistical feature extraction is proposed to compress the pin-level dataset into component-level. Thirdly, a neural-net model with feeding imbalance control is established for Task 1. Fourthly, the random forests model is applied for both Task 2 and Task 3. Moreover, a threshold moving technique is proposed to optimize the threshold selection. Finally, the results show that our models achieved F1-scores of 0.44, 0.54 and 0.71 (average of 0.56) using the test dataset for the three tasks, respectively.

References

Giordano, D., & Trevisan, M. (2022). "PHME Data Challenge". *European conference of the prognostics and health management society*.

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

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *the Journal of Machine Learning Research*, *12*, 2825-2830.

Batista, G. E., Prati, R. C., & Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter*, *6*(1), 20-29.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, *16*, 321-357.

He, H., Bai, Y., Garcia, E. A., & Li, S. (2008, June). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)* (pp. 1322-1328). IEEE.

Provost, F. (2000, July). Machine learning from imbalanced data sets 101. In *Proceedings of the AAAI'2000 workshop on imbalanced data sets* (Vol. 68, No. 2000, pp. 1-3). AAAI Press.

Maloof, M. A. (2003, August). Learning when data sets are imbalanced and when costs are unequal and unknown. In *ICML-2003 workshop on learning from imbalanced data sets II* (Vol. 2, pp. 2-1).

Biographies

Haichuan Tang received a B.E. (in 2009), and Ph.D. (in 2015) both in Electrical Engineering from Southwest Jiaotong University, China. He also received Engineer Diploma in 2012 from Ecole Centrale de Nantes, France. He currently leads the AI lab of CRRC Academy. His main research interests are data-based fault diagnosis and prognosis.

Yin Tian received a B.E. degree in Electronic and Information Engineering, in 2009, and a Ph.D. degree in Traffic Safety Engineering, in 2015, both from Beijing Jiaotong University. He currently serves as deputy director of the Technology Research Department of CRRC Academy. He is conducting research regarding AI, big data, and their applications in manufacturing, logistics, and maintenance.

Junyan Dai is a Ph.D. student at Rutgers University. His research focuses on big data analysis for intelligent transportation problems. He received a B.S. degree in Computer Science from Rutgers University.

Yuan Wang obtained his Ph.D. (in 2019) and bachelor's degree (in 2014) at Southwest Jiaotong University, China. He is a cool guy who loves programming and algorithm design.

Jianli Cong is a Ph.D. student in Civil Engineering at Southwest Jiaotong University. His research focuses on railway inspection and sensing technologies.

Qi Liu received a B.E. degree in 2016, and M.Eng. Degree in 2018, both in Software Engineering, from Beijing Jiaotong University. She currently works at the AI Lab of CRRC Academy. Her research interests include Prognostic and Health Management in the field of rail transit and wind power.

Xuejun Zhao received his Ph.D. degree in Safety Science and Engineering from Beijing Jiaotong University. He has joined CRRC Academy as an algorithm engineer since 2020. His research interests mainly focus on signal processing algorithm development and algorithm acceleration based on multi-core computing platforms.

Yunxiao Fu received his Ph.D. in Transportation tool application engineering from Beijing Jiaotong University (Beijing, China), in 2017. His current research projects in AI Lab of CRRC Academy include developing intelligent control strategy of train in transit operation.