Technical-report: A287570 dataset

Chia-Ming Lee
Institute of Data Science,
National Cheng Kung University
re6124043@gs.ncku.edu.tw

April 7, 2023

1 Abstract

Traditionally, defect detection tasks have been regarded as classification problems, which can be analyzed using supervised learning methods to estimate whether test data is defective based on the distribution of the data. After observation, we believe that the image data we receive contains multi-modal information, including different types of images, maps of particle defects, maps of yellow substrate defects, and numerical statistical data. We used three different methods to analyze this data. The first method is based on optical character recognition (OCR), where we used the CnOCR module to convert the embedded numerical statistical data on the original image into a CSV file, and then analyzed it using machine learning methods. The second method involved converting the original image data into pure image data and analyzing it using convolutional neural networks. The third method integrated both datasets and used a multimodal model to predict whether the original data is defective.

2 Exploratory Data Analysis

This A287570 dataset consists of 336*2 different images, including two types of images: one is the distribution map of pink spot defects, and the other is the distribution map of yellow base defects. They both share the same statistical data, which includes a total of 38 numerical variables obtained through OCR scan.

In order to analyze the feature distribution of images in a high-dimensional space, we utilized t-SNE to project the images onto a two-dimensional plane for data visualization. This allowed us to visualize the distribution of different image categories in the data and assess the degree of separability between them. The use of t-SNE facilitated the identification of potential challenges in distinguishing between certain categories based on their feature representations if our use image-based approach to analysis this dataset.

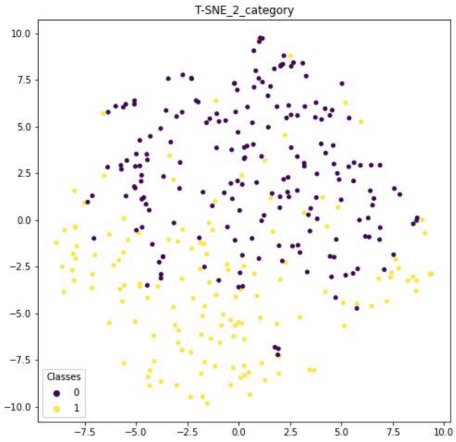

Figure 1: The t-sne map of pink dot defect images cropped from original data, which reflect the distribution of image between two different classes.

The statistical data part of A287570 dataset have 38 numeric variables and 1 binary variable, including "Radius-5", "Radius-10", "Radius-15", "Radius-20", "Radius-25", "Radius-30", "Radius-35", "Radius-40", "Radius-45", "Radius-50", "TotalPoint", "X-axis-Avg", "X-axis-std", "X-axis-norm-uplimt", "X-axis-norm-downlimit", "X-axis-Ca", "X-axis-Cp", "X-axis-Cpk", "X-axis-3Sigma", "X-axis-Avg+3Sigma", "Y-axis-Avg", "Y-axis-std", "Y-axis-norm-uplimt", "Y-axis-norm-downlimit", "Y-axis-Ca", "Y-axis-Cp", "Y-axis-Cpk", "Y-axis-3Sigma", "Y-axis-Avg+3Sigma", "offset-Avg", "offset-std", "offset-norm-uplimt", "offset-norm-downlimit", "Process-StandardlityCa", "Process-PrecisionCp", "Process-AbilityCpK", "Process-3Sigma", "Process-Avg+3Sigma", and "label". The "label" variable is the target variable we aim to predict, with 194 samples labeled as normal(0) and 142 samples labeled as defective(1).

3 Analysis Process

Our analysis is divided into several sub-parts: (1) a machine learning model that primarily uses statistical data, (2) a convolutional neural network model that primarily uses image data, and (3) a multimodal model that combines the previous two models. We will introduce each of them in turn.

3-1-1. Variable selection Generally, when analyzing structured tabular data, data analysts use various mathematical methods to perform dimensionality reduction in order to avoid the curse of dimensionality, reduce the risk of overfitting, and make the model lightweight for easier deployment.

In this dataset, all the statistical data obtained from the original images through OCR packages are numerical data, which can be easily subjected to

dimensionality reduction using PCA or LASSO. Our dimensionality reduction process is divided into two stages: we use LASSO in the first stage, and the box-plot method in the second stage to eliminate those features that have small differences between positive and negative samples in terms of statistical values.

3-1-2. Preliminary Modeling We believe that after dimension reduction using LASSO and boxplot methods, the original data is more suitable for analysis using tree-based models. Tree-based models are built by minimizing entropy to split the feature space. If there is a large difference in variable distribution between categories, the tree-based model will have a better ability to analyze it. We first use the XGBoost model to test the effect without adjusting any hyperparameters. We want to know the model's performance without tuning any hyperparameters and use the best metric (f1-score) as the baseline.

3-1-3. Threshold Selection In supervised learning, the model outputs a confidence score or probability that tells us how sure it is in its classification of test data. For binary classification problems, the threshold is usually set at 0.5, with values above and below indicating predicted class 1 and class 0, respectively. Choosing the best threshold is critical for optimal model performance and is based on metrics such as ROC curves, precision-recall curves, or cost functions.

3-1-4. K-fold Validation K-fold cross-validation involves dividing a dataset into K subsets or "folds", and then performing K rounds of model training and testing, using K-1 folds for training and leaving one fold out for testing each time. By using K-fold cross-validation, we can make full use of the dataset and reduce the model's dependence on a single data split. Overall, K-fold cross-validation can provide average performance metrics and stability of the model, helping us better understand if the model can generalize to unseen data.

3-1-5. Model's Metric In a classification problem, the following three metrics are commonly used for evaluating the model performance, providing insights into the model's detection ability, identification ability, and overall performance respectively:

Sensitivity (Se), also known as recall or true positive rate, measures the proportion of true positive samples that are correctly predicted by the model. It is calculated as TP / (TP + FN).

$$Sensitivity = \frac{TP}{TP + FN} \tag{1}$$

Specificity (Sp), also known as precision or true negative rate, measures the proportion of true negative samples that are correctly predicted by the model. It is calculated as TN / (TN + FP).

$$Specificity = \frac{TP}{TP + FP} \tag{2}$$

F1-score is a commonly used metric for evaluating classification model performance. It balances the model's SE and SP. In this report, we use f1-score as our model performance's metric.

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{3}$$

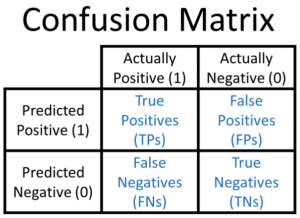

Figure 2: Confusion Matrix[1]

3-2-1. Observation on some sample We observed that the pink dot or yellow background patterns in the images are located at fixed positions in the original dataset. Therefore, we cropped the main analysis area from the original images according to their fixed coordinates, resulting in a 350x350 resolution image data. After observing some samples, we found some characteristics that could be referenced:
(1) The pink dot pattern of normal images generally converges within a circle of radius 35.
(2) The pink dot pattern of abnormal images may have different types, which can be roughly divided into (i) the pattern is too dispersed and obviously spreads outwards, (ii) the center of the pink dot is shifted away from the center of the coordinate axis, (iii) the pink dot pattern is distributed along the X-axis, and (iv) the pink dot pattern is distributed along the Y-axis.

3-2-2. Image Denoising There are some noises in the original images, such as irregular gray pixels appearing in the white area, black circular axis markings, and watermarks. Generally speaking, removing the noise from the images can help the model extract features more effectively and further improve its performance. In the processing, we iteratively traverse each pixel in the image, and if the values of the RGB channels are all below or above a certain threshold, we directly convert the pixel values in the RGB channels to [255, 255, 255], which is white. After denoising, only the distribution of pink dots is retained in the image, and we hope that this can improve the model's performance.

3-2-3. Convolutional Neural Network Convolutional Neural Network (CNN) is a type of deep learning neural network that is widely used in image

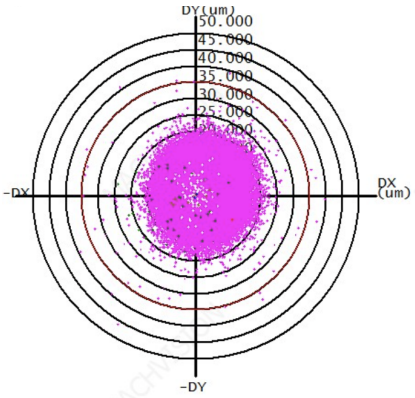

Figure 3: Normal images - distribution converge in center

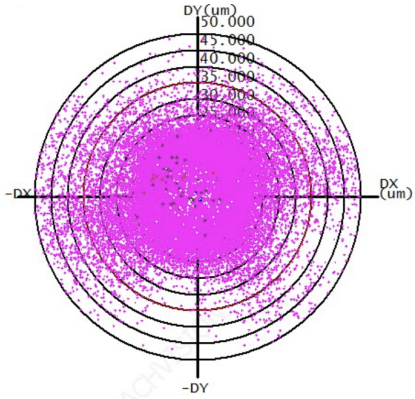

Figure 4: Abnormal images - distribution center offset

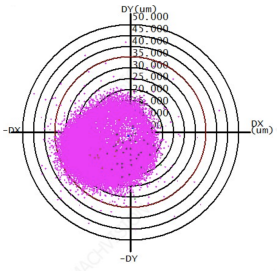

Figure 5: Abnormal images - distribution is too scattered outward

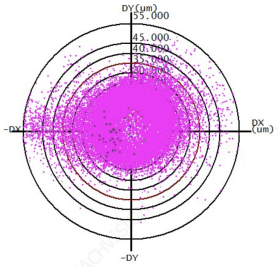

Figure 6: Abnormal images - distribution is scattered along the X-axis

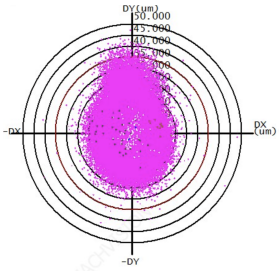

Figure 7: Abnormal images - distribution is scattered along the Y-axis

and video analysis. It is designed to process inputs image, by using convolutional layers to automatically learn hierarchical representations of the input data. The main advantage of CNNs is their ability to capture spatial features of the input data through the use of convolutional filters. This makes them particularly effective for tasks such as object detection, image classification, and image segmentation.

4 Experiment Result

Our experiment is mainly divided into three parts: the first part is based on statistical table data, the second part is based on image data, and the third part is the experimental result obtained by modeling based on both statistical table data and image data. The random seed for all experiments in this paper is set to 42.

4-1-1. Threshold Selection We divided our data into K subsets using a similar approach to K-fold cross-validation(here we call K-fold splitting), and cyclically split the original dataset into training and testing sets with a ratio of 0.8:0.2. We then built K different models based on each subset of training data, which were used to predict the testing data, and confidence scores were obtained for each prediction. Afterwards, we converted the confidence scores to predicted classes based on a threshold (defaulted to 0.5) and calculated the f1-score.

We then tested 100 threshold values ranging from 0.4 to 0.6 with a step size of 0.02, and evaluated each threshold value using K=3 and K=5. We determined the optimal threshold value, which was then applied to the original model, resulting in improved classification performance.

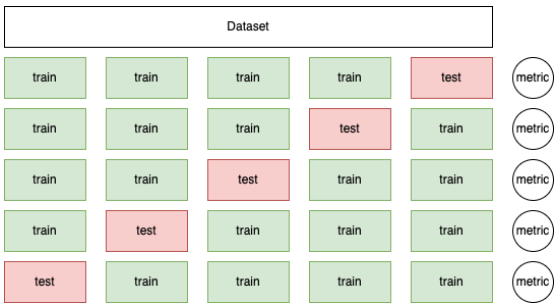

Figure 8: 5-folds splitting data for threshold selection

4-1-2. Baseline We used K-fold cross-validation to build the model, with XGBoost as the baseline model. XGBoost is an ensemble tree model with good accuracy and robustness, and has sparse pruning function to prevent overfitting. It is also easy to visualize the model's operating process. We also analyzed the

results of the threshold selection and variable selection. However, the modeling results showed that there was no any difference in classification performance between them.

Note: Based on the experimental results, we set the f1-score of 0.9536 as the baseline and expect the performance of subsequent models to be higher than this score.

These are the hyperparameters used in this experiment part: $\text{num}_boost_round = 1000, early_stopping_rounds = 50$. The other hyperparameters not mentioned here are set to default values.

4-1-3. Various classification models We used the autogluon toolkit developed by Amazon, which can build various classification models from raw data and search the hyperparameter space for each model to find the best set of hyperparameters to build the model. In the table, we obtained the performance of various models on the testing and validation sets (5-fold validation was also used during model training), and also included the inference time of the models on the testing data, allowing us to make the most appropriate choice based on both model performance and inference speed.

We also included whether variable selection was used in the results. We found that variable selection did not have a significant impact on performance, but it could reduce inference time and make the model more lightweight.

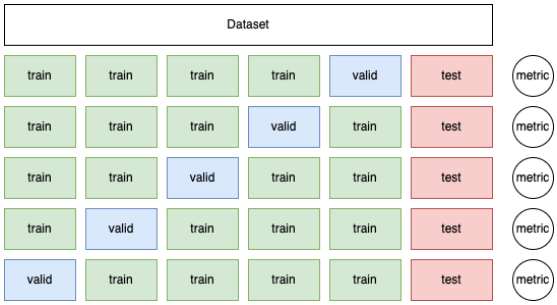

Figure 9: 5-folds validation

Table 1: Threshold selection results.

Model	"k"-folds splitting	f1-score(w/o)	f1-score(w/)	threshold
XGBoost	3	0.9453	0.9513	**0.468**
XGBoost	5	0.9389	0.9390	**0.598**

Table 2: A preliminary analysis based on XGBoost.

Model	f1-score	SE	SP	threshold	v.s.
XGBoost(w/o k-folds validation)	0.9536	0.9524	0.9615	0.5	no
XGBoost(w/o k-folds validation)	0.9536	0.9524	0.9615	0.5	yes
XGBoost(w/ 3-folds validation)	0.9536	0.9524	0.9615	0.5	no
XGBoost(w/ 3-folds validation)	0.9536	0.9524	0.9615	0.5	yes
XGBoost(w/ 5-folds validation)	0.9536	0.9524	0.9615	0.5	no
XGBoost(w/ 5-folds validation)	0.9536	0.9524	0.9615	0.5	yes
XGBoost(w/ 3-folds validation)	0.9536	0.9524	0.9615	0.468	yes
XGBoost(w/ 5-folds validation)	0.9536	0.9524	0.9615	0.468	yes
XGBoost(w/ 3-folds validation)	0.9536	0.9524	0.9615	0.598	yes
XGBoost(w/ 5-folds validation)	0.9536	0.9524	0.9615	0.598	yes

Table 3: Modeling results based on AutoGluon(with 5-fold validation and variable selection).

Model	f1-score-test	f1-score-valid	pred-time for testing
KNeighborsDist	0.880952	0.915254	0.067300
KNeighborsUnif	0.880952	0.877193	0.057298
NeuralNetFastAI	0.919540	0.967742	0.106381
LightGBMXT	0.952381	0.983607	0.006735
ExtraTreesGini	0.963855	0.983607	0.154337
ExtraTreesEntr	0.963855	0.966667	0.131274
RandomForestEntr	0.963855	0.966667	0.126605
RandomForestGini	0.963855	0.966667	0.120581
XGBoost	0.963855	0.966667	0.040725
NeuralNetTorch	0.963855	0.966667	0.018339
CatBoost	0.963855	0.966667	0.005778
WeightedEnsemble-L2	0.963855	0.984127	0.004352
LightGBM	0.963855	0.984127	**0.001407**
LightGBMLarge	**0.964706**	0.984127	0.009607

Table 4: Modeling results based on AutoGluon(with 5-fold validation and without variable selection).

Model	f1-score-test	f1-score-valid	pred-time for testing
KNeighborsUnif	0.857143	0.892857	0.062110
KNeighborsDist	0.860465	0.877193	0.069639
WeightedEnsemble-L2	0.898876	1.000000	0.025436
LightGBMXT	0.898876	0.984127	0.015811
NeuralNetTorch	0.938272	0.931034	0.018663
ExtraTreesGini	0.963855	0.983607	0.147148
RandomForestGini	0.963855	0.949153	0.122010
RandomForestEntr	0.963855	0.949153	0.118920
ExtraTreesEntr	0.963855	0.966667	0.109206
XGBoost	0.963855	0.949153	0.030311
CatBoost	0.963855	0.966667	0.007040
LightGBM	0.963855	0.984127	**0.001561**
NeuralNetFastAI	**0.964706**	0.983607	0.098168
LightGBMLarge	**0.964706**	0.984127	0.011801

5 Conclusion

References

[1] Measuring Performance: The Confusion Matrix https://glassboxmedicine.com/2019/02/17/measuring-performance-the-confusion-matrix/

[2] 2

[3] 3