Detect PCB Stack-up Error with Machine Learning Methods

Ta Chang Chen, Gold Circuit Electronics Ltd., Taoyuan, Taiwan
Fei Fei Kao PhD, Ming Chuan University, Taoyuan, Taiwan
Huang Yu Chen PhD, Taoyuan, Taiwan

Abstract
In manufacturing multilayer printed circuit boards (PCBs), the PCB layer stack-up usually includes multiple plies of various types of prepreg along with the inner layer core material. Most of the layer stack-up process is manual operation relying on an operator to place the correct type and quantity of prepreg at designated dielectric openings. Missing or adding additional one or more plies of prepreg can result in a overall board thickness which will possibly still fall within its thickness specification limits but, its electrical property will be greatly affected (e.g. impedance, loss etc.). In this case, this abnormal board could possibly escape to customer unless detected by more expensive electrical property testing. Impedance measurement may catch the missing / extra prepreg for the dielectric layers related to impedance. Other layers without impedance control could still possibly escape. Considering impedance TDR measurement is time consuming and costly measurement process, a new lower cost approach was developed to detect extra / missing prepreg and validate the correctness of the stack-up. Our first approach is looking for statistical outliers for each given lot board thickness measurements after lamination. This approach is able to detect extra / missing prepreg, but its false alarm is too high due to missing one ply prepreg can be less than 0.002" thick which is generally less than one sigma of total board thickness. Thus, second approach using machine learning technique was developed with board thickness data (known bad and known good board thickness) and other features (e.g. prepreg type, lamination parameters etc.). From the model we developed and validated that our escape rate is still 0% and its false alarm rate was reduced by 85%.

Introduction
Due to the development of high frequency and high-speed transmission in information industry, PCB customers demand stricter signal transmission integrity of PCB. Therefore, the control of signal insertion loss becomes a new challenge to PCB manufacturers. Energy is lost from a signal as heat in the conductor resistance and as heat in the dielectric caused by the inherent material properties of the base material itself. Total loss is therefore the sum of conductor loss and dielectric where such loss is undesired or at least needs to be understood and managed [1]. Insertion loss formulas are:

$$\text{Insertion Loss} = \text{Conductor Loss} + \text{Dielectric Loss}$$

$$\text{Conductor Loss} \approx I^2\sqrt{\rho \cdot \pi \cdot \mu \cdot F}\left(\frac{1}{W}+\frac{1}{6H}\right)$$

$$\text{Dielectric Loss} \approx \frac{k \cdot F \cdot \sqrt{Dk} \cdot Df}{C}$$

Where H represents dielectric thickness, Dk represents dielectric constant and Df represents dielectric loss tangent. From the formulas above, dielectric plays an important role of signal integrity. Nowadays, stack up process still lacks automation solution in most PCB plants. Operators are required to follow each part number's stack up requirement to prepare and place correct quantity and type of prepreg at each dielectric opening. For higher layer counts design, the risk of stacking wrong quantity of prepregs will increase as more prepreg required to manually placing. After lamination process, board thickness will be 100% automatically measured, once an stack-up error board with extra or missing prepreg was produced, the board thickness of this stack-up error board was possibly within thickness specification limits, since commonly used thin prepreg thickness is around 1mil to 3mil, that's quite small compared to total board thickness. The stack-up error board has a chance to be detected at impedance measurement process after outer layer or solder mask process showing in Figure 1. Measuring impedance and loss using TDR is a high cost and time-consuming method, if impedance and loss are sampling measurement, a stack-up error board has high risk shipping to customer. Accordingly, our purpose is to establish an effective detection method which can find out stack-up error boards with extra or missing prepreg right after lamination.

This paper and presentation "Detect PCB Stack-up Error with Machine Learning Methods" was first presented at the 2020 IPC Apex Expo Technical Conference and published in the 2020 Technical Conference Proceedings.

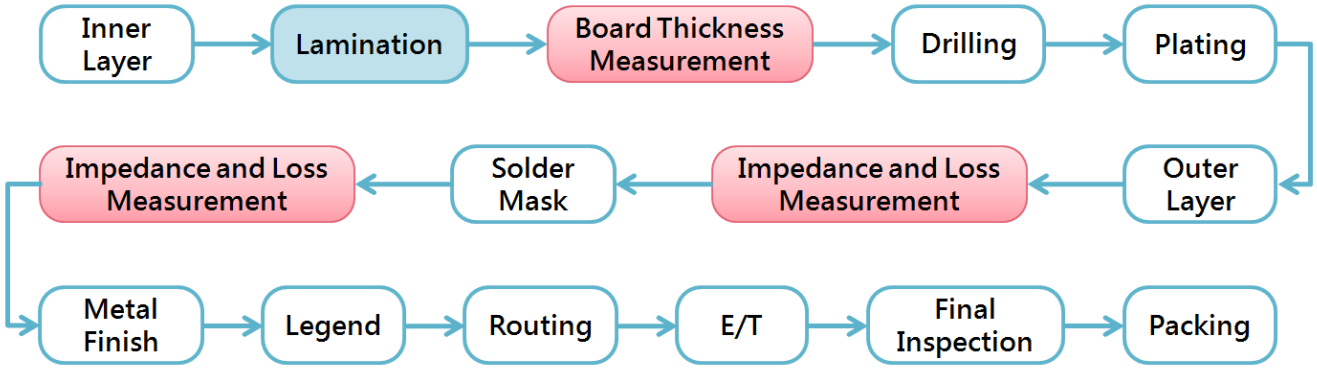

Figure 1 – Process where stack-up error board has a chance to be detected

Detect PCB Stack-up Error with Traditional Statistical Methods

After lamination, every board will pass through automatic board thickness measurement equipment which contains 3 laser displacement sensors in a row, and it will measure 3 rows (left, middle and right) of each board, that's totally 9 points in a board shown in Figure 2. In order to catch stack-up error boards, we drew a histogram first like left part of Figure 3. From the histogram, all thickness measurement data were within specification limits and capability index Cpk of this lot equaled 2.3, we couldn't find any problem here. As mentioned previously, a stack-up error board with extra or missing prepreg will not cause total board thickness out of specification limits in most situations, we can't determine a board is a stack-up error board by whether its board thickness is out of specification limits or not. And then we drew a box and whisker plot of board thickness by board as right part of Figure 3. From this plot, we observed that one board had different board thickness performance compared to other boards in the same lot. Can we detect stack-up error boards by using outliers of board average thickness? This is another idea.

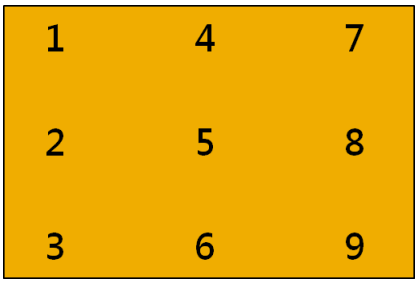

Figure 2 – Board thickness measurement positions of each board

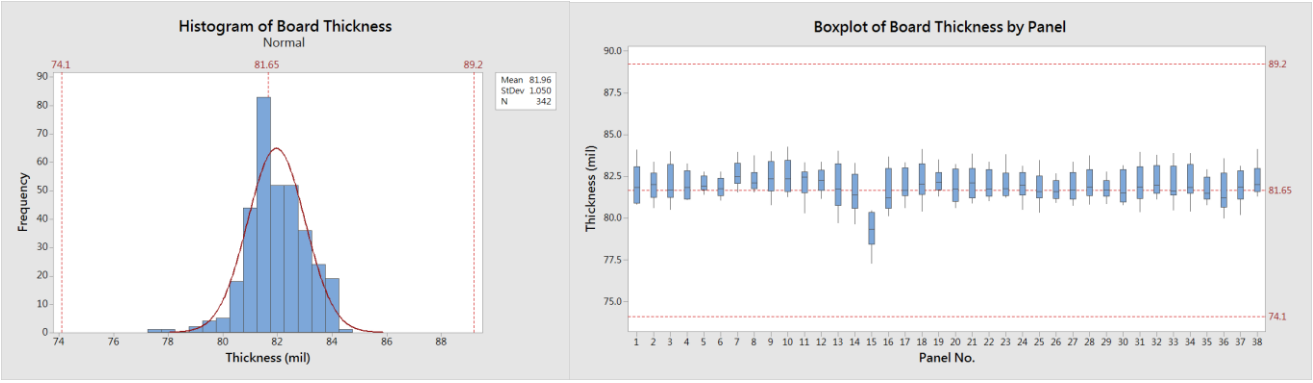

Figure 3 – Box and whisker plot of board thickness of a lot grouping by board with a stack-up error board

The most commonly used statistical outlier detection methods are (1) box and whisker plot and (2) MAD (median absolute deviation) test. In a box and whisker plot like Figure 4, the bottom of the box is 1st quartile (Q1) of the data and the top of the box is 3rd quartile (Q3) of the data, and interquartile range (IQR) equals Q3 – Q1. If any data point is greater than Q3 + 1.5*IQR or less than Q1 – 1.5*IQR, this data point is an outlier.

Another method, MAD test, is an effective, non-graphical method of detecting outliers in a set of data. To determine if there are outliers in a data set $x_1, x_2, x_3, \ldots, x_n$:

(1) Calculate $\tilde{x}$, the median of the x_i

(2) For each x_i, calculate its absolute deviation from the median: $|x_i - \tilde{x}|$

(3) Calculate MAD, the median of the absolute deviations from the median:
$MAD = \text{median of}(|x_1 - \tilde{x}|, |x_2 - \tilde{x}|, |x_3 - \tilde{x}|, \dots, |x_n - \tilde{x}|)$

(4) For each x_i, calculate $M_i = \frac{x_i - \tilde{x}}{MAD}$

(5) The data point x_i is an outlier if $|M_i| > 5.2$, meaning that the point is more than 3.5*σ away from the median.

We applied these 2 methods on 15 known bad boards to check if these 2 methods were able to detect all known bad boards. The result is as Table 1. From the result, there were 6.7% and 13.3% escape rate (number of stack-up error board was not detected/total number of stack-up error board), this was not good enough since zero escape rate was our goal.

Table 1 – Stack-up error board detection results using statistical outlier

Outlier detection method	Number of total stack-up error board	Number of escape board	Escape rate
Box and whisker plot	15	1	6.7%
MAD Test	15	2	13.3%

Thus, a 3rd approach was developed to decrease escape rate. This method compared board thickness of each board to whole lot board thickness performance. The analysis process is:

(1) Calculate median (denoted by M) of whole lot board thickness data represents the center of board thickness distribution of this lot. Since average will be affected by outliers, we decided to use median to represent center of the lot board thickness.

(2) Because of the property of lamination process, resin flow at the edge of the board is greater than center of the board. This will result in board thickness in the center thicker than perimeter as shown as a contour plot like Figure 5, and thickness distribution of a board will be affected by design pattern as well. Therefore, after a trial and error process, we calculate mean of the ith board (denoted by $\bar{X}_i$) represents the center of one board thickness distribution, and calculate range of the board (denoted by R_i) to represent the uniformity of board thickness.

(3) Calculate $Diff_i = |M - \bar{X}_i|$, if $Diff_i$ is greater than the thinnest prepreg thickness of the designed stack up times 0.9 and R_i is less than board thickness specification times 0.15, then we will determine this board as a stack-up error board.

After we implemented this detection method online and total 540,000 boards were produced in this period of time, 510 boards were detected by this method as stack-up error boards. After performing 100% cross-section, 31 of 510 boards were true defects with extra prepreg or missing prepreg, and 479 boards were false alarm. This new outlier detection method detected stack-up error boards indeed (escape rate = 0%) but along with 94% (=479/510) false alarm rate. Obviously, this method was unacceptable and still had room for improvement. Note that if we applied box and whisker plot and MAD test on those data, the escape rates were 9.7% (=3/31) and 12.9% (=4/31).

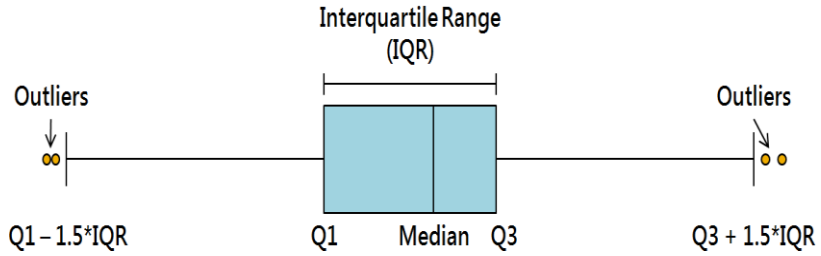

Figure 4 – Definition of outliers in box and whisker plot

Detect PCB Stack-up Error with Machine Learning

The previous statistical methods used board thickness data only to detect stack-up error boards. To establish the stack-up error board detection method, we chose 12 features which were in relation to board thickness including material, machine and manufacturing method. Since we had 12 features and the data structure was not balanced (i.e., a balanced data structure has an equal number of observations for all possible combinations of feature levels), it was not easy to build a prediction model with statistical method. In the meantime, we found that if a problem contains following three essences, there will be an opportunity to build a prediction model with machine learning.

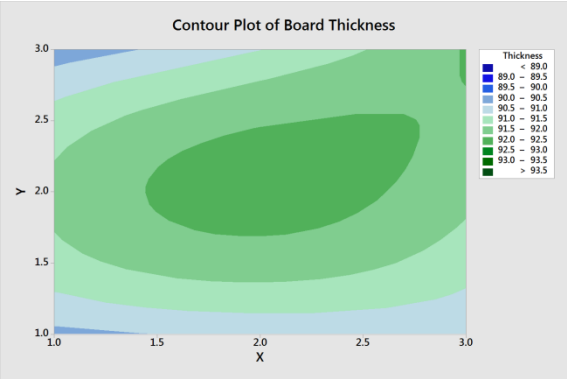

Figure 5 – Contour plot of board thickness of a board

(1) Exists some 'underlying pattern' to be learned
(2) No easy programmable definition
(3) Data is available for this pattern

That's the beginning we implement machine learning.

What is Machine Learning?

Machine learning is the scientific study of algorithms and statistical methods that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead. Traditional programming looks like the top half of Figure 6. Using data and rules to derive answer through developed program. These rules are expressed in a programming language and are the bulk of any code that you write. Ultimately, these rules will act on the data and give you an answer. If we don't know the rules or the rules are too complex to code, and then it fails to let computer output answers when we input data we collected. But what if you reverse this diagram, you give machine answers with the data and let the machine figure out what the rules are, like the bottom half of Figure 6 [3]. That's machine learning algorithm. In this article, we will introduce how we use ensemble learning, a method of machine learning, to establish an expert determination system to detect stack-up error board including four commonly used classifiers such as logistic regression, support vector machine, random forests and extreme gradient boost. This expert system will combine the results of four classifiers and make a final prediction that a board is a stack-up error board or not.

Feature Engineering

As noted above, 12 features which are in relation to board thickness after lamination list in Table 2. 8 of 12 features are numerical data and the other 4 are categorical data. None of which contains missing value.

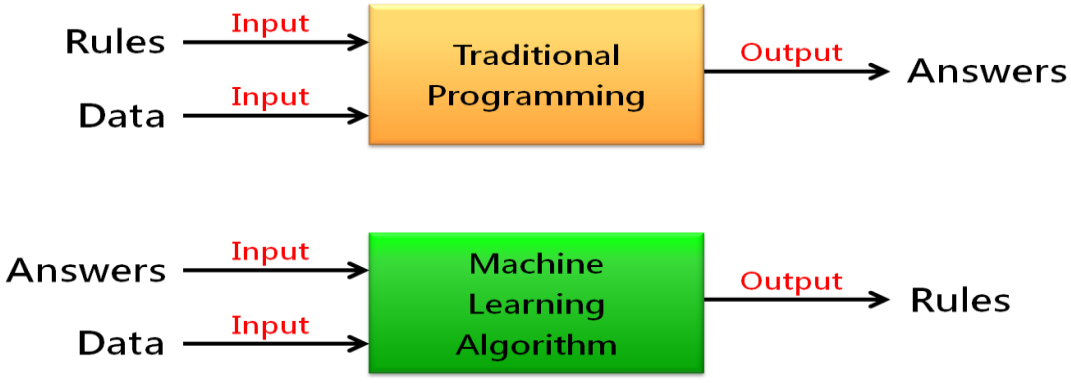

Figure 6 – Traditional programming and machine learning algorithm

Table 2 – Features List

Feature	Data Type	Feature Engineering Methods	Contain Missing Value?
Average of Board Thickness (denoted by $\bar{X}_i$)	Numeric	Standardize	No
Median of Lot Board Thickness (denoted by M)	Numeric	Standardize	No
$\text{Diff}_i = \lvert M - \bar{X}_i \rvert$	Numeric	Standardize	No
Board Thickness SPEC	Numeric	Standardize	No
Thickness of Thinnest PP	Numeric	Standardize	No
Resin Content of Thinnest PP	Numeric	Standardize	No
Material Type	Category	One Hot Encoding	No
Glass Type of Thinnest PP	Category	One Hot Encoding	No
Board Length	Numeric	Standardize	No
Board Width	Numeric	Standardize	No
Lamination Program	Category	One Hot Encoding	No
Lamination Machine	Category	One Hot Encoding	No

For categorical data, most machine learning algorithms cannot deal with it, so we have to encode this type of data into numbers before we train models. There are 2 commonly used encoders. The first one is label encoder, which assign a number to a different category. For example, in the left part of Figure 7, 1 is the label for Red, 2 is the label for Black, 3 is the label for Blue, and 4 is the label for Green. In this way, we can convert categorical data to numerical data easily. But a new problem arose. Since there are different numbers in the same column, machine learning algorithms will misunderstand the data to be in some kind of order like 1 < 2 < 3 < 4. But this isn't the case at all. To overcome this problem, we use one-hot encoder. One-hot encoder splits one column of categorical data into multiple columns, and each column includes only numbers 1 and 0 that depends on which column has what value. For example, in the right part of Figure 7, we will get 4 new columns, one for each color – Red, Black, Blue, and Green. For rows which have the original column value as Red, the Red column will have value 1 and the other 3 columns will have value 0. Similarly, for rows which have the original column value as Black, the Black column will have value 1 and the other 3 columns will have value 0. In this way, the Euclidean distance between any 2 categories are the same.

For numerical data, some machine learning algorithms, e.g. support vector machine, are sensitive to the feature scales. If the scale of one numerical feature is much greater than the other numerical feature, then it cannot train an effective model. Therefore, each numerical feature will be standardized to eliminate difference of scale. Following equation shows how to standardize data:

$$z_i = \frac{x_i - \bar{x}}{s_x}$$

where x_i is the original data, $\bar{x}$ is the mean of the feature and s_x is the standard deviation of the feature.

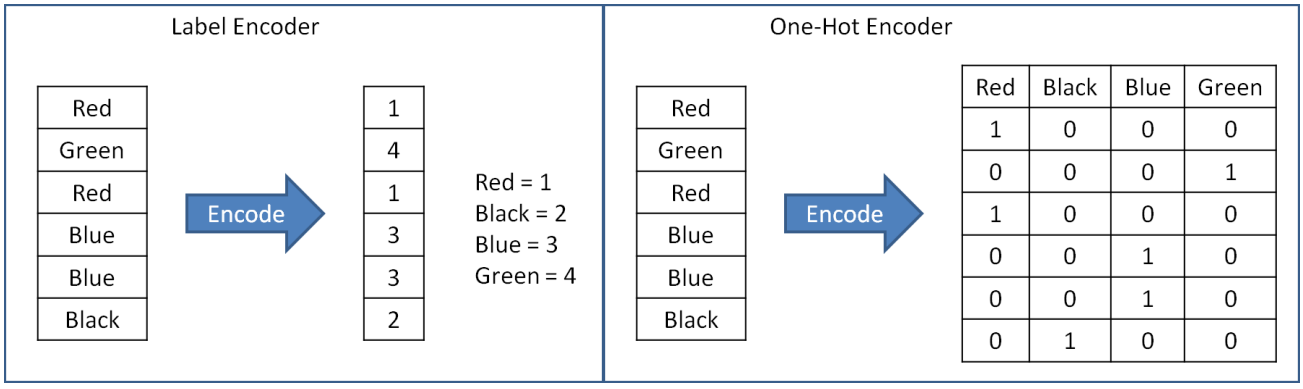

Figure 7 – Label encoder and one-hot encoder

Prediction Model Performance Indicators

Stack-up error is rarely happened, the defect rate of stack-up error is 57dppm (= 31/540,000×10^6). If we use accuracy as model performance indicator, a classifier predicts all boards as good boards will have accuracy over 99.99%. This will mislead us about thinking that this classifier performs excellently, but actually it's not. Instead of using accuracy as model performance indicator, escape rate and false alarm rate are better indicators, where

$$\text{Accuracy} = \frac{\text{Number of Actual OK and Predicted as OK Boards} + \text{Number of Actual NG and Predicted as NG Boards}}{\text{Number of Total Boards}}$$

$$\text{Escape Rate} = \frac{\text{Number of Actual NG but Predicted as OK Boards}}{\text{Number of Actual NG Boards}}$$

$$\text{False Alarm Rate} = \frac{\text{Number of Actual OK but Predicted as NG Boards}}{\text{Number of Actual OK Boards}}$$

Settings of Output Class Number, Threshold and Voting Methods

For each row of data, we collected, we had performed cross-section examination to identify the "answer" of each board is missing prepreg (value = –1), normal (value = 0) or extra prepreg (value = 1). The purpose of this project is detecting stack-up error board no matter it is missing prepreg or with extra prepreg. Will the classifiers have lower escape rate and false alarm rate if we simplified this question by setting the answer into 2 classes, normal (value = 0) and stack-up error (value = 1)? We will compare which setting is better later.

Next, when we input a set of data into classifier prediction models, each classifier will output probabilities of this board belonging to missing prepreg, normal or with extra prepreg. For example, prediction result is like (0.01, 0.67, 0.32), it means that this board belongs to missing prepreg with probability 0.01, belongs to normal with probability 0.67 and belongs to with extra prepreg with probability 0.32. Usually, we will take the class with highest probability as our final result (normal board in this case), but it is not the only way to determine which class this board belongs to. We can set a threshold, for example 0.15, and if any probability of missing prepreg or with extra prepreg is greater than threshold, whether the probability is the greatest or not, we will determine this board as missing prepreg or with extra prepreg. If both probabilities of missing prepreg and with extra prepreg are greater than threshold, we will choose the higher one. Decreasing threshold will have higher chance to trigger stack-up error board, but along with higher false alarm rate as well. On the other hand, increasing threshold will have lower chance to trigger stack-up error board and have lower false alarm rate as well. Adjusting threshold is a trade-off between escape rate and false alarm rate, it depends on how much tolerance you have to escape rate and false alarm rate. We will determine an appropriate threshold value in the later comparison.

The last item we are going to set is voting method. Voting is one of the methods to ensemble prediction results from several classifiers. There are 2 commonly used voting methods, soft voting and hard voting. The soft voting averages the probabilities of all classifiers first and then comparing the average probabilities with threshold to determine is the board normal or not. In contrast to soft voting, hard voting determines whether this board is a stack-up error board by each classifier first, and then makes the final prediction by a simple majority vote. We made a little change in this project: if any classifier determined this board is a stack-up error board then we make a final prediction that this is a stack-up error board, any escape is not allowed. Figure 8 is an example of soft voting and hard voting. After we input data into each classifier prediction model, they will output predictive probabilities. The soft voting averages probabilities of each classifier first, and then we have probability 0.02 that this board belongs to missing prepreg, probability 0.853 that this board is normal, and probability 0.127 that this board is with extra prepreg. Suppose threshold is 0.15 and none of probabilities of missing prepreg and with extra prepreg is greater than threshold, it concludes this board is normal. For hard voting, logistic regression and random forests determine this board is with extra prepreg since the probability of with extra prepreg is greater than 0.15. Support vector machine and extreme gradient boost predict this board is normal, since both probabilities of missing prepreg and with extra prepreg are less than threshold. Finally, it concludes this board is with extra prepreg. Different voting methods will cause different results, we will decide which voting method is better in this project in the following comparison. To decide appropriate output class number, threshold value and voting method, we will use collected dataset to build model in different situation. At first, we sample 20% of the dataset randomly as test set, and the other 80% is train set. We use train set to build 4 ensemble prediction models with different output class number (2 and 3) and voting methods (soft voting and hard voting), and then use test set to identify what is the maximum threshold value which will imply zero escape rates. The result is showed in Figure 9, red dotted line is escape rate versus threshold and blue solid line is false alarm rate versus threshold.

Classifier	Predictive Probability			Predict when Threshold = 0.15
	Missing PP (-1)	Normal (0)	With Extra PP (+1)	
Logistic Regression	0	0.838	**0.162**	**+1**
Support Vector Machine	0.003	0.933	0.064	0
Random Forests	0.056	0.791	**0.153**	**+1**
Extreme Gradient Boost	0.021	0.849	0.130	0
Average	0.02	0.853	0.127	

Hard Voting:
There are 2 classifiers determine this board is with extra PP, then it is a **stack-up error** board
Hard Voting Criteria:
Any classifier determines a board is stack-up error board then concluding this board is a stack-up error board

Soft Voting:
None of the averaged predictive probabilities of Missing PP and With Extra PP are greater than Threshold = 0.15, then it is a **normal** board
Soft Voting Criteria:
Any averaged probability of Missing PP or With Extra PP is greater than threshold then concluding this board is a stack-up error board

Figure 8 – Example of Soft Voting and Hard Voting

The green circle in each plot means the maximum threshold with zero escape rate, higher maximum threshold with zero escape rate means the model has higher confidence to detect stack-up error boards. From the result, hard voting with 3 output classes had highest maximum threshold with zero escape rate (bottom right part of Figure 9). We did above process for 15 times and drew the result of maximum threshold value (denoted by T) with zero escape rate and false alarm rate when threshold equals T in Figure 10 and Figure 11. In Figure 10, hard voting with 3 output classes always has larger threshold than the other 3 settings, simultaneously, it had 2nd lowest false alarm rate among 4 settings. Based on this analysis, we set output class number as 3 and ensemble results of four classifiers using hard voting. Usually, the trained model has a little bit overfitting hence we take a conservative strategy to set threshold as 0.15.

Training Models

There are some hyperparameters that we have to set to avoid the model overfitting and underfitting in every classifier listed in Table 2. Likewise, we split the dataset randomly into 2 groups, one with 80% amount of the data is train set and the other 20% is test set. We train a machine learning model with train set data, and then predict train set and test set using this model. Next, we calculate the escape rate of train set and test set. A good model will have the lower escape rate in test set and similar escape rate between train set and test set. The first condition means this model has good predictive capability and the second condition means the model is not overfitting or underfitting. Keep doing this by setting different key hyperparameters values to find an appropriate setting for every classifier. At last, the key hyperparameters setting of each classifier are listed in Table 3. Now, we are ready to train machine learning models use key hyperparameters setting listed in Table 2 and all dataset.

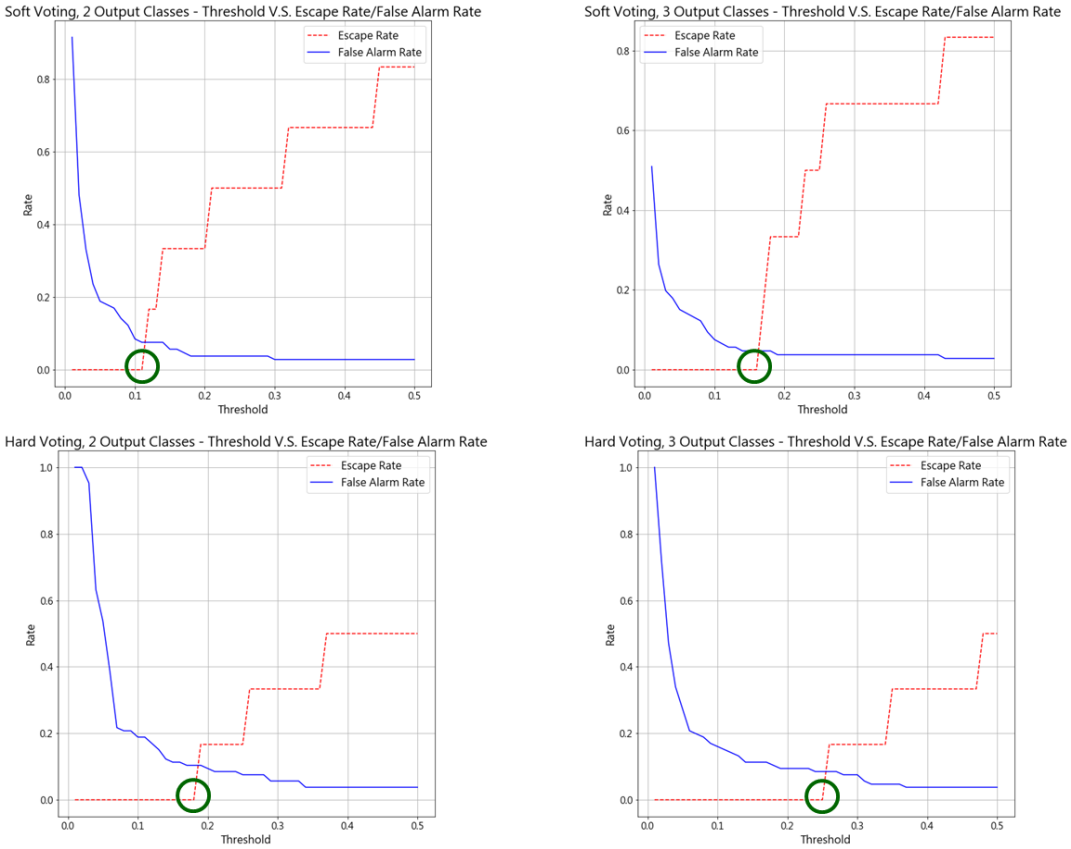

Figure 9 – One of the tests finding appropriate output class number, threshold and voting method

Figure 10 – 15 tests comparing maximum threshold with zero escape rate of different output class numbers and voting methods

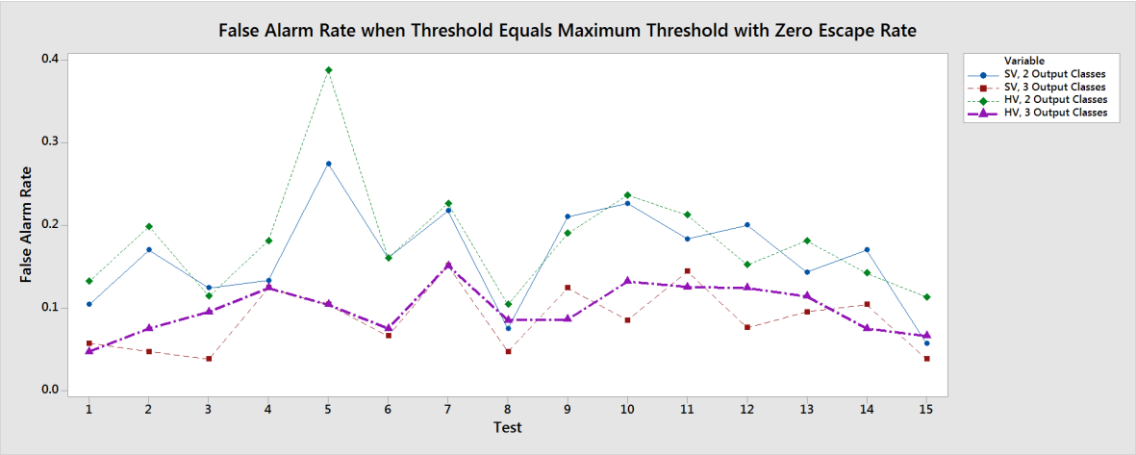

Figure 11 – 15 tests comparing false alarm rate when threshold equals maximum threshold with zero escape rate of different output class numbers and voting methods

Table 3 – Key hyperparameters setting of every classifier

Classifier	Key Hyperparameters
Logistic Regression	max_iter = 1000, C = 10
Support Vector Machine	max_iter = 2000, gamma = 0.1, C = 0.01
Random Forests	n_estimators = 1000, max_depth = 20, max_features = 'sqrt'
Extreme Gradient Boost	n_estimators = 1000, learning_rate = 0.001, max_depth = 15

Effectiveness Validation

We implemented this model online to evaluate the effectiveness from May 2019 to July 2019. During this period of time, we received 157 alarms from the 3rd statistical approach. After that, we predicted those boards with machine learning model and then performed cross-section examination to identify the real status of those boards. Finally, the performance of machine learning model was showed in a confusion matrix as Table 4.

Table 4 – Confusion matrix of evaluation results

		Actual		
		Missing PP (-1)	Normal (0)	Extra PP (+1)
Predict	Missing PP (-1)	4	15	0
	Normal (0)	0	123	0
	Extra PP (+1)	0	10	5

Note: All 9 stack-up error boards are made intentionally to test system effectiveness

From this table,

$$\text{Escape Rate} = \frac{0}{9} = 0\%$$

$$\text{False Alarm Rate} = \frac{25}{148} = 16.9\%$$

$$\text{Saving} = 1 - \frac{34}{157} = 78.3\%$$

where saving means we had to perform 157 cross-sections examination before implementing machine learning but reduced to 34 after implementing machine learning.

Summary

Although it's not a great benefit in this case, we just used the data in our database originally and coded with a free open source software Python to reduce waste in manufacturing environment. In the past few years, we've heard a lot of topics about Industry 4.0 and smart manufacturing, and then we started to collect data online. Until now, I believe most of those data are stored in database and have not been used well, this case showed us the benefits of correctly using these data. There

are lot of training programs and training materials on the internet and in real life, just get your foot in the door and follow the guidelines as follows [2], machine learning is not unreachable.

(1) Frame the problem and look at the big picture.
(2) Get the data.
(3) Explore the data and get insights.
(4) Prepare the data to better expose the underlying data patterns to Machine Learning algorithms.
(5) Explore many different models and short-list the best ones.
(6) Fine-tune your models and combine them into a great solution.
(7) Present your solution.
(8) Launch, monitor and maintain your system.

Reference

[1] Polar Application Note AP8196 What is insertion loss? http://www.polarinstruments.com/support/si/AP8196.html
[2] Hands-On Machine Learning with Scikit-Learn & TensorFlow, Aurélien Géron
[3] Intro to Machine Learning, youtube channel TensorFlow https://www.youtube.com/watch?v=KNAWp2S3w94&t=238s