**Authors**:

A Defect Prediction Case Study Featuring Printed Circuit Boards Using Ball-Grid Array Packages During Assembly

## Embracing Industry 4.0 Digital Transformation

## Published January 2019

Greg Vance (Rockwell Automation)

Mikica Cvijetinovic (Rockwell Automation)

Francisco P Maturana (Rockwell Automation)

Balamurugan Balakreshnan (Microsoft)

Michael Walton (Microsoft)

Stephen Philip (Microsoft)

David Crook (Microsoft)

Phillip M. LaCasse (Rockwell Automation)

Dr. Wilkistar Otieno (University of Wisconsin – Milwaukee)

Megan R Fink (Rockwell Automation)

Published: February 2019

Table of Contents

[Introduction 1](#_Toc1481050)

[Background 3](#_Toc1481051)

[Experimental Case Study 4](#_Toc1481052)

[Results and Discussion 6](#_Toc1481053)

[Process Overview 8](#_Toc1481054)

[Solution Overview 10](#_Toc1481055)

[Remarks 13](#_Toc1481056)

[References 17](#_Toc1481057)

[Disclaimer: 18](#_Toc1481058)

|  |
| --- |
| Introduction |
| A BGA is a type of SMT packaging used to affix devices such as microprocessors to the PCB. Pins arrayed in a grid deposit very small solder balls on the component that match a corresponding grid of copper pads on the PCB. A solder paste inspection (SPI) machine measures a set of predefined parameters on each solder paste deposit and flags for rework any out-of-tolerance boards. At this stage of the process, rework is minimally disruptive with only a trivial cost associated. After a successful SPI test and upon connecting the component to the PCB, the assembly passes through a reflow oven affix the component to the PCB.  At the end of the SMT line, each PCB undergoes an automated optical inspection (AOI), which either allows the board to pass through (“GOOD”) or flags it for review by an operator. An operator reviews each flagged board and designates it as either a PASS or FAIL. PASS boards advance with other GOOD boards, and FAIL boards undergo corrective action through prescribed processes. The purpose of the AOI station is primarily to ensure that the PCB is correctly assembled, with all necessary components present and within acceptable tolerance.  Upon successful completion of the AOI station, a PCB proceeds to an in-circuit testing (ICT) station, where it undergoes electrical testing. It then proceeds forward to final assembly (FA) where it is put through functional testing and then added to whatever product for which it is destined. The product receives a final round of functional testing before it is packaged for shipping.  The case study was motivated by two distinct but interrelated factors.  The first motivating factor is the desire to reduce downline defects. This is a persistent desire for any manufacturer, and BGA package types are of interest due to difficulties associated with inspecting internal joints and rework. These difficulties were identified early in the development of BGA technology [5], [6] and they remain a challenge for manufacturers today. Data captured and made available by the Twinsburg IIOT project can potentially mitigate the effects of the first challenge, and the second challenge can be mitigated by identifying thresholds for that data that, when exceeded, indicate increased probability of a downline defect. The key question is whether such thresholds exist. “Contact without connection” defects are especially concerning, in which the solder successfully joins the component and the PCB (“contact”) but there is not the ensuing electrical connectivity. In these circumstances, the PCB will typically pass visual or optical inspections but then fail either in electrical testing, functional testing, or after distribution to the customer, which would be the worst-case scenario. Types of contact without connection defects include “head-in-pillow” [7], “black pad” [8], and “non-wet open” [9], [10] defects.  The second motivating factor is the opportunity that it provides for continued validation of a Fuzzy Approach to Feature Reduction and Filtration (FAFRAP) framework, introduced in [11] and undergoing in-progress review and validation by the authors. This case study represents one element of that review and validation, along with additional, as-yet unpublished work. The FAFRAP process is an analyst’s tool that assists in formulating applied machine learning models, and [11] is an outcome of the same academia-industry partnership that undergirds this article. FAFRAP consists of a hierarchy of sequential filters that remove features that do not contribute to effectively solving the problem of interest. The output for each remaining feature is a crisp label for rank-ordering and qualitative description for preservation for knowledge management purposes. |

# Background

Two preliminary discussions will add value prior to proceeding to the details of the case study. The first is to provide background on a Python package, **T**ime **S**eries **F**eatu**R**e **E**xtraction on the basis of **S**calable **H**ypothesis tests (TSFRESH) [12], which relates to the methodology employed for the defect prediction case study; the second is to provide background on the FAFRAP framework.

TSFRESH, initially proposed in [13], provides an automated Python tool to extract a predefined set of up to 794 features from time series data. It then individually tests each extracted feature for statistical independence from the target. Features exhibiting a statistically significant dependency are retained, and statistically independent features are discarded. The threshold for obtaining the cutoff is determined using the Benjamini-Hochberg test to control the false discovery rate (FDR), explained in [14].

As stated in Section 1, FAFRAP is presented in [11] as an analyst’s tool to assist in the formulation of applied machine learning models by filtering out features that do not contribute to solving the problem of interest and outputting a crisp numeric value and a descriptive label for the features that remain. The filtering process is twofold. The first filter is conceptually identical to the TSFRESH filter for statistical independence and, in practical use, may be accomplished using TSFRESH so long as Python code is employed. The second filter is to assign a value to each filter that quantifies its membership in “good” solutions versus membership in “less good” solutions. This is accomplished by executing machine learning models using randomly-generated subsets of the full feature set as inputs for model training. Using whatever metric of interest is selected by the analyst, which is necessarily subject to the specific problem under study, a threshold is determined which differentiates a high-quality solution from a low-quality solution. A value is computed for each feature using Equation (1) and Equation (2), once to quantify membership in the high-performance group and once to quantify membership in the low-performance group.

In Equation (1), represents the membership of feature in group *i*, represents the number of subsets in group , and represents an indicator function equal to 1 if feature is contained in subset where and 0 if it is not.Equation (2) normalizes the values produced by Equation (1).

The values computed by Equation (1) and Equation (2) are then used as inputs to a Fuzzy Inference System (FIS), which outputs a crisp value and descriptive label. Crisp values are then used to sort the remaining features and identify the top “N” features for model inclusion. Once the top “N” features are identified, all possible combinations are checked to identify the optimal subset of features for model training.

The FAFRAP framework is in its infancy, with a nontrivial number of hyperparameters or sub-processes to fine-tune. Hyperparameters within the framework include FDR threshold in Filter #1, number and size of subsets to generate in Filter #2, threshold for top “N” features for model inclusion, and number of FIS runs to perform.

# Experimental Case Study

3.1. Description of Data

The data for this case study consists of five parametric features measured at SPI and defect data extrapolated from ICT. The five features of interest are:

* X01: Deviation of solder paste deposit from target location (x-direction).
* X02: Deviation of solder paste deposit from target location (y-direction).
* X03: Solder paste deposit volume, as a percentage of component-specific benchmark.
* X04: Solder paste deposit height.
* X05: Solder paste deposit area, as a percentage of component-specific benchmark.

Defect data consists of a binary response variable, equal to 1 if a defect is detected at ICT and 0 of not.

To organize the data, the following additional features were collected but not incorporated into any predictive models as input variables:

* F01: Unique barcode designator associated with a single panel, upon which several PCB modules might exist. Each module on a panel will be manufactured into the same style PCB.
* F02: Location reference ID for modules on a panel.
* F03: Location reference ID for component location on a PCB.
* F04: Unique designator for a specific component pin, which creates a specific solder paste deposit. This feature is used for sorting purposes in organizing the data for model inclusion.

The concatenation of each unique F01\_F02\_F03 combination provides a distinct location for a component, within a certain module, at a certain position on the panel. This information allows a specific defect to be mapped to specific records of the five parametric features.

Two datasets were collected, each containing data for a single type of PCB. Only one type of PCB was present in a dataset, but each dataset used a different type. Table 1 summarizes the two datasets.

Table 1: Data Summary

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Records** | **Total Defects** | **Training Set Non-Defect** | **Training Set Defects** | **Test Set Non-Defect** | **Test Set Defects** |
| DS01 | 4421 | 233 | 3359 | 177 | 829 | 56 |
| DS02 | 6916 | 41 | 5500 | 32 | 1375 | 9 |

As shown in Table 1, each dataset is highly unbalanced, with a fraction of the overall records exhibiting an identified defect. This presents the immediate concern that there may not be sufficient defect data for model training. Of the two datasets, DS01 might have the better prospects at first glance due to the higher proportion of defective records.

3.2. Model Approach

The nature of the available data presents some challenges associated with a traditional modeling approach in that defect information is available only to the precision of the PCB location, but SPI parametric data is available to the precision of the BGA pin level. This means that there could be scores or even hundreds of records associated with a single defect. The concern from a data perspective is that solder paste deposits that are fully within specification would be assigned a defective outcome in the training set if they reside in the same PCB location as an identified defect. This would be incorrect and could lead to confusion with interpretation of model results. A second concern, this time from a process perspective, is that a test already exists to test the individual solder paste deposits for conformity to specifications; the traditional approach to model line by line at the record level would be redundant, unless the goal is to validate the specifications, which is a different analysis altogether. Finally, and most concerning, the traditional approach of examining each record in isolation fails to capture the interaction between parametric data from different pins on the same location.

An alternative approach, employed in this case study, is to consider all pins in a location simultaneously by creating an array and sorting it in some fixed manner for each of the five features. The model is indifferent to the method of sorting, so long as the arrays are sorted in the same way every time. The default for this case study was to sort by F04 alphanumerically.

Using this approach, each record would be assigned a label of 0 or 1 depending on whether a defect was identified at that location, and that label would map to five arrays, one for each of the five parametric SPI features. The uniqueness of this approach is in applying the TSFRESH time series feature extraction algorithm to each of the five arrays, thus generating up to 794 \* 5 = 3970 features for each record. Even though TSFRESH is designed with time series in mind, there is no reason why the features that it generates could not be applied to problems in other contexts. The FAFRAP framework would then be applied to those 3970 features.

# 

# Results and Discussion

Results from this case study fall into two domains of interest. The first domain is the bottom-line, direct answer to the defect prediction question. The second domain is the application of FAFRAP to the case study and any framework-specific lessons that might be learned.

Table 2 summarizes the confusion matrices for each dataset at each level of filtration in the FAFRAP framework. The results indicate comparable results from one level of the framework to the next, suggesting that the features being removed make little to no contribution to the overall model.

Table 2: Aggregated model predicted results

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Dataset** | **FAFRAP Filtration Level** | **Number of Features** | **Correctly Predicted Non-Defective** | **Incorrectly Predicted Non-Defective (Escape)** | **Correctly Predicted Defective** | **Incorrectly Predicted Defective (False Alarm)** |
| DS01 | None | 3970 | 808 | 32 | 24 | 21 |
| DS01 | Filter #1 | 2210 | 810 | 33 | 23 | 19 |
| DS01 | Final | 15 | 807 | 34 | 22 | 22 |
| DS01 | Best Accuracy | 9 | 817 | 21 | 35 | 12 |
| DS01 | Best Precision | 9 | 821 | 29 | 27 | 8 |
| DS01 | Best Recall | 7 | 805 | 21 | 35 | 24 |
| DS01 | Best F1 | 9 | 817 | 21 | 35 | 12 |
| DS02 | None | 3970 | 1370 | 6 | 3 | 5 |
| DS02 | Filter #1 | 901 | 1372 | 7 | 2 | 3 |
| DS02 | Final | 15 | 1371 | 5 | 4 | 4 |
| DS02 | Best Accuracy | 8 | 1375 | 4 | 5 | 0 |
| DS02 | Best Precision | 8 | 1375 | 4 | 5 | 0 |
| DS02 | Best Recall | 7 | 1367 | 3 | 6 | 8 |
| DS02 | Best F1 | 8 | 1375 | 4 | 5 | 0 |

Table 3 contains scoring metrics, selected because they each carry a clear physical interpretation. Any one of them may be the metric of choice depending on the situation. Four metrics are calculated: accuracy, precision, recall, and F1 score.

Accuracy is the ratio of correct classifications to total classifications. For highly unbalanced datasets, as those used for this case study, this is a less desirable metric because the model may simply classify all records as non-defective. This is identical to decision making under the status quo, with no model at all.

Precision is a measure of how accurate the model’s predicted outcome is when it predicts a defect. This metric might be preferred if there is a high cost associated with false alarms. A false alarm occurs when the model predicts a defect for a non-defective record. Another scenario by which precision might be useful is if there are believed to be multiple root causes of defects, and the input features, in this case SPI parametric data, are not believed to contribute to all of them. In this circumstance, there would be the understanding that not all defects would be flagged by this model, but the desire would be for any flagged defects not to be false alarms.

Recall is a measure of how well the model guards against escapes. An escape is when a defective part passes undetected from one stage to the next. This metric might be preferred if there is a high cost associated with defect escapes, but a relatively low cost associated with rework if the problem is caught early. In this circumstance, precision might be sacrificed for recall.

Finally, F1 score is an attempt to capture elements of both precision and recall, and it is computed by calculating the harmonic mean of the two. This metric might be preferable if the costs associated with false alarms and escapes are similar.

Note that the number of features in the ‘best’ rows are subsets of the 15 features in the Final – Top 15 row. The final step in the FAFRAP framework, after identifying some number of top performing features, is to combinatorically iterate through all possible subsets of the final list. Note the optimal subset of features is not necessarily the same for every metric. For DS01, the same optimal subset gives the best accuracy and F1, but a different subset gives the best precision and the best recall. For DS02, the same subset of features gives optimal values for all metrics except for recall.

Table 3: Model comparison at FAFRAP levels

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Dataset** | **Filter Level** | **Number of Features** | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
| DS01 | None | 3970 | 0.9401 | 0.5333 | 0.4286 | 0.4752 |
| DS01 | Filter #1 | 2210 | 0.9412 | 0.5476 | 0.4107 | 0.4694 |
| DS01 | Final – Top 15 | 15 | 0.9367 | 0.5 | 0.3929 | 0.44 |
| DS01 | Best Accuracy | 9 | 0.9627 | 0.7447 | 0.6250 | 0.6796 |
| DS01 | Best Precision | 9 | 0.9582 | 0.7714 | 0.4821 | 0.5934 |
| DS01 | Best Recall | 7 | 0.9492 | 0.5932 | 0.625 | 0.6087 |
| DS01 | Best F1 | 9 | 0.9627 | 0.7447 | 0.6250 | 0.6796 |
| DS02 | None | 3970 | 0.9921 | 0.375 | 0.3333 | 0.3529 |
| DS02 | Filter #1 | 901 | 0.9928 | 0.4 | 0.2222 | 0.2857 |
| DS02 | Final – Top 15 | 15 | 0.9935 | 0.5 | 0.4444 | 0.4706 |
| DS02 | Best Accuracy | 8 | 0.9971 | 1.0 | 0.5556 | 0.7143 |
| DS02 | Best Precision | 8 | 0.9971 | 1.0 | 0.5556 | 0.7143 |
| DS02 | Best Recall | 7 | 0.9921 | 0.4286 | 0.6667 | 0.5217 |
| DS02 | Best F1 | 8 | 0.9971 | 1.0 | 0.5556 | 0.7143 |

With respect to the second domain, the results are encouraging and consistent with previous validation tests in [11]. The filtration mechanisms in FAFRAP worked as intended and consistent with what might be expected from this case study. Because the TSFRESH features were initially conceived to be extracted in a time-series context, there was the expectation that many of the extracted features would have little or no relationship to ICT defects. However, the question that the case study explored was if there were underlying or hidden factors that were systemically contributing to defects. If that is the case, then it is natural to expect that that factor might appear in the behavior of at least one of the 794 features extracted by the software.

Given these expectations, the FAFRAP filter results make intuitive sense. In DS01, the first filter reduced the feature set from 3970 to 2210. It is unlikely that all 2210 features individually exhibiting statistical dependence with ICT defect identification do so based on a factor that is distinct from all the others. Rather, some underlying factor might be manifesting itself in many or all those 2210 features. The final feature set bears this out because the model produced by the final feature set scores comparably to the models at previous filtration levels for each of the metrics computed. In some cases, the reduced feature set performed better. Similar reasoning may be applied to the DS02 results.

# Process Overview

Pick and place is a machine process within an automated assembly line that produces circuit boards in an electronic assembly manufacturing plant, as shown in Figure 1. These are very sophisticated machines that can be programmed to populate various sizes of circuit boards based on product requirements.

The pick and place operation consist of picking electronic components/parts from a tape or tray, taking an image of them to identify the placement centroid and then placing them in the appropriate location on the PCBs as directed by a placement program. The nozzles mount to various spindles contained within a head that moves at a high speed. Nozzle vacuum degradation (clogging or mechanical defect) increases the likelihood to miss-pick/scrap components and increases the potential of a defect in the electronic assembly. This equipment is robotically controlled and can be programed to optimally meet the board-layout requirements. The real challenge is to be able to leverage the knowledge conveyed in the generated information that is extracted during this operation to learn potential defects and other tendencies of interest, yet to be discovered.

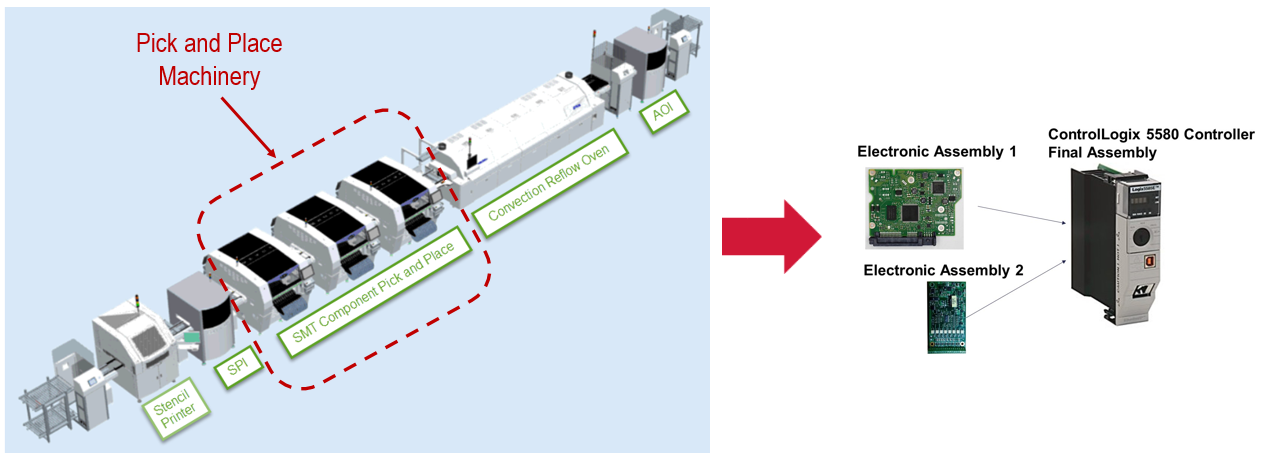


Figure 1 – Electronic assembly line

An area where improvements can be made is minimizing the rate of the failure to pick and place components. If a nozzle starts to fail, it tends to miss-pick components. As this error rate increases it can cause downtime and reduces the operation efficiencies as well. By detecting and/or predicting the failure/degradation of the nozzles, it will facilitate alerting support personal to proactively troubleshoot the nozzle(s) before they become a significant problem. Early detection also minimizes the opportunity to introduce defects into the product.

This pick and place equipment handles about 1 - 30 nozzles per head and there are one to four heads with nozzle changers with some number of nozzles. When a nozzle fails, it can take a significant amount of time in troubleshooting to determine the specific nozzles that need replacement. One goal is to be able to control downtime by anticipating the nozzle failure. Knowing when it can fail will transform the maintenance process into a proactive activity based on factual calculations and higher certainty.

One of the main concerns here is also to avoid false positives or false alarming to avoid overwhelming the operators with false alerts.

If we can avoid any defect here it will save the manufacturer on rework. Also, there are components like Integrated Circuit (IC) chips where only one side can be soldered. So, the component placement is very critical. The Pick and Place is a critical machine in the line where there may be 1 or more machines in the line based on the product physical requirements.

Sometimes the components are placed shifted or skewed on the board which will cause defective boards after permanent assembly (soldering). A whole panel holding multiple electronic assembly modules might need rework which can take 1 to 48 hours, depending on what the defects is. Position issues can be detected in downstream inspection stations such as the Automated Optical Inspection (AOI) machine while electrical issues are only detected at the final stage at functional integrated circuit testing, where the boards are already assembled. All these defects affect the OEE of the plant.

# Solution Overview

The above approach validates the time series and then build’s new features that can be used to help predict the model with better accuracy. The System uses Classification based Machine learning models which are also decision trees to better make predictions that are closer to accurate.

To build the model there are following workflow must happen to do the above analysis. Here is the detail outline of what needs to happen.

1. Collect Data
2. Process Data
3. Store Data

The above are precursor to able to build the model. The above process is already implemented as part of connected enterprise where the data from Koh Young Solder Paste Inspection machine’s data is taken and pushed into Rockwell’s Data Lake in RA private Cloud data center extension.

Now to build the model at scale and Agility we need storage and computing platform that can accommodate data and ability do machine learning compute with the data in scale. This is where the cloud opens opportunity. So to enable the high computation Time series processing the data is pushed into cloud storage called Blob Storage and then by leveraging computation like Azure Databricks we can scale the compute for the processing. The figure below shows the Architecture and the design used for the above.



Fig 5.1 – Architecture for Solder Paste Inspection – Feature Extraction.

The Architecture has 3 parts to it

1. Factory Floor
2. Rockwell RA Private Cloud in Azure for Data Lake
3. Rockwell PaaS based Cloud processing

All these 3 components are vital to build a complete end to end architecture for this solution to be deployable.

To collect data from the factory floor we use Common gateway platform built by Rockwell and renamed Factory Talk Analytics Edge. The Data collector has in built features to get the data out and push that into kafka or Iot Hub.

The Data feed that collects data every 15 minutes is pushed in to Rockwell’s Own Connected Enterprise data lake. Here is where all the IT and OT data gets converged and available for reporting. It’s more a Kudu based impala data lake that is built to store all the data.

To leverage the scale and agility we are using cloud to do Feature extraction and validation and Model Building.

Once the Model is built inference code is written with the saved model pickle file and then packaged as Docker container running in ubuntu to make it deployable to anywhere. This deployment can be in Rockwell’s private cloud or on-premise data center or factory floor.

The Storage Blob and Azure Data bricks and Azure Databricks Notebooks are used to build the model and save the pickled file. Model performance is stored in Azure SQL for auditing and reporting as well. The Model development are also connected to DevOps to maintain version and auditing of code as well.

Azure DevOps is used as a tool for DevOps and Git is used as code repository. The Machine learning Service is a PaaS offering to create docker containers and maintain then in azure container registry for deployment. For example, once the inference code is built then Azure machine learning SDK has built in feature in one line to build the Realtime docker container and push that to container registry. The Web service can also be tested as well.

Iot Hub is the primary source that manages the devices. Hub is where we can create new devices and load them into iot hub. Hub manages the device security. Hub also has features to deploy docker containers to edge and manage them on the cloud. There are options to target specific device or group of devices and what needs to be deployed as well. Hub provides bidirectional communication to manage job or send commands back to device to take action on. So for example, the image that got stored in container registry by machine learning model service is taken and pushed into devices as docker containers by creating a configuration in hub. This enables seamless container deployment which is also called functionality deployment to edge devices. Not all devices can have the same functionality, so in this way we can change as we need. The version of the image to be used can also be provided with which container registry and it’s credentials as well. This make Iot Hub and integral part of the architecture as it manages the security and the communication.

# Remarks

The approach to model SPI parametric data holistically by PCB location and extract features using a Python library initially intended for time series has been shown to produce results that merit continued exploration of the approach. Additionally, the FAFRAP algorithm to reduce and prioritize features performed as intended. From these perspectives, the case study might cautiously be deemed a tentative success.

However, additional work is necessary to truly harness the possible benefits from this approach, and additional knowledge is needed to maximize the potential of the FAFRAP.

The first area of additional work is in the interpretation of results. This study intentionally employed decision trees because they provide clear interpretability. Splits in the trees can be programmed into data tracking or visualization tools, and dashboards that correspond to the model can be created and monitored in lieu of continually computing the extracted features and running the models. However, the tradeoff in using decision trees is that they typically underperform other algorithms such as random forest or deep neural networks. This is a known, intentional limitation and tradeoff in the approach taken in this paper.

Therefore, it is necessary to analyze the specific features highlighted by FAFRAP, identify their physical interpretations as applied to time series problems, and determine what if any physical interpretation can be extended to manufacturing. This will inform subsequent decisions on how to use these models, any KPIs that may be necessary to create, or any existing KPIs to track. Absent this work, there is no reason not to use more elegant but less interpretable methods like random forest or deep neural networks.

The second area of additional work is in generalizability of results. The study should be performed again on additional datasets with a different mix of PCBs. Not only would it be value added to use additional PCBs, but there are reasons to condition datasets on factors other than PCB type. For example, each PCB contains a variety of different components. Conditioning on component part instead of PCB may allow better generalizability of the results.

With respect to FAFRAP, a known area of future research is the tuning of its hyperparameters, as alluded to in Section 2. This study revealed a potential improvement associated with the hyperparameters related to grouping the features into high-performing and low-performing solutions. There is not a known technique to determine the minimum number of subsets to generate so that there is sufficient representation of each feature for the FIS to effectively rank it. Up to this point, the analyst’s discretion has been used, with the bottom-line model result being the discriminant. However, additional knowledge would speed up the process and free the data scientist from performing redundant work.

A second hyperparameter for which a general principle would be helpful is where to set the threshold for what constitutes a “top” feature. In this case study, there were 2210 features that survived the first filter and were input into the FIS. The threshold was tested at a number of values such as 50, 100, or 200, with the guiding principle being within the top 10%.

Finally, continued replication of the study is necessary to identify if the same features highlighted by FAFRAP will continue to be prioritized in subsequent models using different PCBs or component parts. If the true underlying factor is not directly calculated by any of the time-series-style features, which is a reasonable working hypothesis, then it is possible that the algorithm will have many different “final” feature sets that all achieve comparable results.

In conclusion, the approach outlined in this research could be whimsically described as a “poor analyst’s deep learning”, where the TSFRESH-extracted features operate conceptually similarly to a single hidden layer of neurons. Like deep learning models, this approach transforms the input features into some number of new features. Unlike deep learning models, those new features can be identified and interpreted precisely because they correspond to some known calculation or physical interpretation. Also, unlike deep neural networks, model training is relatively straightforward and the lack of successive nonlinear transformations between hidden layers gives weaker prospects for uncovering hidden relationships in the data. The tradeoff between the two approaches is one of model performance for interpretability.

List of Abbreviations

AOI Automated Optical Inspection

BGA Ball Grid Array

CE Connected Enterprise

ICT In-circuit Testing

FA Final Assembly

FAFRAP Fuzzy Approach to Feature Reduction and Prioritization

FDR False Discovery Rate

FIS Fuzzy Inference System

IIOT Industrial Internet of Things

IT Information Technology

KPI Key Performance Indicator

OT Operational Technology

PCB Printed Circuit Board

RA Rockwell Automation, Inc.

SMT Surface Mount Technology

SPI Solder Paste Inspection (machine)

TSFRESH Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests

UWM University of Wisconsin – Milwaukee

Declarations

*Availability of Data and Materials*

The datasets generated in this study are currently proprietary and unfortunately are not available for public distribution.

*Competing Interests*

The authors declare that they have no competing interests.

*Funding*

The authors affirm that there are no sources of funding for the research to be declared.

*Author’s Contributions*

FM provided subject matter expert review of Section 1 and Section 2. Additionally, FM was the interface between the authors and the data source for obtaining data for the case study.

WO provided subject matter expert review of Section 3, Section 4, and Section 5.

PL provided primary content development in all sections, as well as the coordination role between authors.

Acknowledgments

Special thanks to the Data Analytics & Insights (DAI) Group, Rockwell Automation, for their expertise, data, and computing resources to execute this work.

# References

[1] Rockwell Automation, *The Connected Enterprise eBook: Bringing People, Processes, and Technology Together*. Rockwell Automation, 2015.

[2] Rockwell Automation, “The Connected Enterprise Maturity Model,” p. 12, 2014.

[3] Rockwell Automation, “The Connected Industrial Enterprise,” 2015.

[4] W. Otieno, M. Cook, and N. Campbell-Kyureghyan, “Novel approach to bridge the gaps of industrial and manufacturing engineering education: A case study of the connected enterprise concepts,” *Proc. - Front. Educ. Conf. FIE*, vol. 2017–Octob, no. November, pp. 1–5, 2017.

[5] G. Hill, “The Analysis of Manufacturing Processes for Plastic Ball Grid Arrays (PBGAs),” *Microelectron. Int.*, vol. 14, no. 1, pp. 16–19, Apr. 1997.

[6] J. Mearig and B. Goers, “An overview of manufacturing BGA technology,” in *Seventeenth IEEE/CPMT International Electronics Manufacturing Technology Symposium. “Manufacturing Technologies - Present and Future,”* 1997, vol. 40, no. 1, pp. 434–437.

[7] K. Seelig, “Head-in-Pillow BGA Defects,” *AIM Solder*, 2008. [Online]. Available: https://www.aimsolder.com/sites/default/files/head-in-pillow\_bga\_defects.pdf.

[8] J. A. Roepsch, R. F. Champaign, and M. R. Downey, “Case Study: The Effect of Severe Black Pad Defect on Solder Bonds on Ball Grid Array Components,” *J. Surf. Mt. Technol.*, vol. 15, no. 3, pp. 19–24, 2002.

[9] L. Kondrachova, S. Aravamudhan, R. Sidhu, D. Amir, and R. Aspandiar, “Fundamentals of the Non-Wet Open BGA Solder Joint Defect Formation,” in *ICSR (Soldering and Reliability)*, 2012.

[10] D. Amir, S. Walwadkar, S. Aravamudhan, and L. May, “The Challenges of Non Wet Open BGA Solder Defect,” in *SMTA International*, 2012.

[11] P. M. LaCasse, W. Otieno, and F. P. Maturana, “A hierarchical, fuzzy inference approach to data filtration and feature prioritization in the connected manufacturing enterprise,” *J. Big Data*, vol. 5, no. 1, p. 45, Dec. 2018.

[12] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, “Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package),” *Neurocomputing*, vol. 307, pp. 72–77, 2018.

[13] M. Christ, A. W. Kempa-Liehr, and M. Feindt, “Distributed and parallel time series feature extraction for industrial big data applications,” Oct. 2016.

[14] Y. Benjamini and D. Yekutieli, “The control of the false discovery rate in multiple testing under dependency,” *Ann. Stat.*, vol. 29, no. 4, pp. 1165–1188, 2001.

[15] Deep learning - <https://en.wikipedia.org/wiki/Deep_learning>

[16] Azure Databricks - <https://docs.microsoft.com/en-us/azure/azure-databricks/>

[17] Azure SQL - <https://docs.microsoft.com/en-us/azure/sql-database/>

[18] Azure functions - <https://docs.microsoft.com/en-us/azure/azure-functions/functions-overview>

[19] Notification hub - <https://docs.microsoft.com/en-us/azure/notification-hubs/notification-hubs-push-notification-overview>

[20] Power BI - <https://docs.microsoft.com/en-us/power-bi/service-azure-and-power-bi>

[21] Artificial Intelligence. - <https://www.microsoft.com/en-us/ai/default.aspx>

[22] Iot Edge - <https://azure.microsoft.com/en-us/services/iot-edge/>

[23] Docker Containers - <https://www.docker.com/resources/what-container>

[24] Azure Iot Reference Architecture - <https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/iot/>

[25] Azure DevOps - <https://docs.microsoft.com/en-us/azure/devops/index?view=azure-devops&viewFallbackFrom=vsts>

[26] Azure Machine learning service - <https://azure.microsoft.com/en-us/services/machine-learning-service/>

[27] Azure Storage Blob - <https://docs.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction>

[28] Azure container registry - <https://azure.microsoft.com/en-us/services/container-registry/>

# Disclaimer:

(c) 2019 Rockwell Automation/Microsoft Corporation.  All rights reserved.  This document is provided "as-is." Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.   
   
This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes. You may modify this document for your internal, reference purposes.