Research Article

Open Access

Ana Elsa Hinojosa Herrera*, Stoyan Stoyanov, Chris Bailey, Chris Walshaw, and Chunyan Yin

Data analytics to reduce stop-on-fail test in electronics manufacturing

https://doi.org/10.1515/comp-2019-0014
Received November 30, 2018; accepted July 25, 2019

Abstract: The use of data driven techniques is popular in smart manufacturing. Machine learning, statistics or a combination of both have been used to improve processes in electronic manufacturing. This paper presents the application of classical techniques to reduce production cycle time by compacting a production test sequence. This set of tests is run on stop-on-fail scenario for quality assurance of an electronical device. Data generated in the production test-set on stop-on-fail scenario challenges the traditional application of the data driven techniques, because of the missing data characteristic. The developed computational procedures handle this application-specific data attribute. The novelty of this work is in the algorithm developed, which applies classical techniques in an iterative environment, as a strategy to analyse incomplete datasets. Results show that the method can reduce a production test set with parametric and non-parametric tests by building an accurate prognostic model. The results can reduce production cycle time and costs. The paper details and provides discussions on the advantages and limitations of the proposed algorithms.

Keywords: decision tree, logistic regression, random forest, incomplete dataset, electronic device

1 Introduction

As part of intelligent production processes, electronic devices are tested after assembly for quality assurance. Automatic testing equipment is popular to execute parametric and non-parametric sets of tests, generating massive and valuable information. On the other hand, the testing process impacts the cost of production due to the time and resources needed. Cost reduction can be achieved by mining test data for predicting the quality of a batch, improving process robustness, or by shortening the production test sequence.

There are successful business cases where testing processes were reduced using data analysis. Parametric analyses are widely used to predict the outcome of a test set, for instance in [1] the authors analysed correlation between each pair of tests items. Variations of the Group LASSO method were used in [2] and [3] to identify tests which could be predicted by a linear combination of other tests. Furthermore, [3] covers the stop-to-fail scenario where the test program stops as soon as a device fails a test and the remaining tests will not be applied.

In [4] the authors applied Chi-Square to discard tests with very small significance values and then used Support Vector Machine (SVM) methods, in particular C-Support Vector Classification (C-SVC), to carry out the test process compaction. Logistic regression is used in [5] to predict results of test sequences where the data is quantitative or qualitative. Another approach used frequently to reduce test dimension is based on Principal Components Analysis (PCA) [6].

Different Artificial Intelligence (AI) techniques are useful for this application. In [7], a combination of a multi-objective Genetic Algorithm (GA) for feature selection, and k-Nearest-Neighbours (k-NN) with an Ontogenic Neural Network (ONN) for prediction is presented. Reference [8] details a successful application of a Feed-Forward Neural Network (FFNN) for the reduction of production tests and detection of defective assets.

Other research reports on the development of a statistical methodology based on Binary Decision Trees (BDT) to reduce test sets by eliminating tests whose output could

***Corresponding Author: Ana Elsa Hinojosa Herrera:** School of Computing and Mathematical Sciences, University of Greenwich. London SE10 9LS. United Kingdom. E-mail: aehinojosa@ieee.org
Stoyan Stoyanov: School of Computing and Mathematical Sciences, University of Greenwich. London SE10 9LS. United Kingdom. E-mail: s.stoyanov@gre.ac.uk
Chris Bailey: School of Computing and Mathematical Sciences, University of Greenwich. London SE10 9LS. United Kingdom. E-mail: c.bailey@gre.ac.uk
Chris Walshaw: School of Computing and Mathematical Sciences, University of Greenwich. London SE10 9LS. United Kingdom. E-mail: c.walshaw@gre.ac.uk
Chunyan Yin: School of Computing and Mathematical Sciences, University of Greenwich. London SE10 9LS. United Kingdom. E-mail: c.yin@gre.ac.uk

be predicted using the results from another test [9]. This methodology is advantageous from the computational time point of view, since considerably less training time is required to build a tree than to train a network. However, sometimes the BDT model is less accurate. In addition, it is difficult to say how the classification model accuracy is affected when the test data is characterised with missing values. Finally in [10] Binary Decision Forests (BDF) were used to identify redundant tests.

In this paper we propose a novel strategy to handle missing data when building classification models to compact production test sets in a stop-on-fail scenario. The approach could be applied with both numerical and non-numerical test results. The data mining approach was written in an R script which covers data gathering, data pre-processing, variables association analysis, iterative within-set model building, its verification and reliability assessment. The algorithm is evaluated by building Logistic Regression, Decision Tree, and Random Forest models.

The remainder of this paper is organized as follows. In Section 2 we cover different data analysis techniques such as logistic regression, decision trees, and random forests. In addition, performance metrics that could be used to contrast models are also included and discussed. The challenges in building models when using incomplete datasets are outlined. In Section 3 we present our novel algorithm for model building. The method is evaluated in Section 4 using historical production test data of an electronic device, and Section 5 concludes the paper.

2 Data analysis techniques

Data Analytics applications are developed both to describe a phenomenon, and for prognosis or prescriptive purposes. In this paper we are going to cover predictive analytics whose aim is to make forecasts based on historical information. However, a descriptive analysis is included as part of the preliminary steps of building a data analytic service. This step is relevant to understand the historical data, before building a model.

2.1 Qualitative comparison of techniques

Examples of different techniques to reduce test sequences in electronics manufacturing are summarized in Table 1. In addition, an evaluation of whether methods could be used with datasets containing numerical and non-numerical variables is included. Furthermore, missing data was evaluated only in [3] where the case of stop-on-fail scenario was covered. Data pre-processing is needed to handle missing values when using any of the other methods.

Different methods for linear correlation analysis have been used for testing reduction [1-3]. However, these methods cannot be used for non-parametric variables.

The PCA approach followed in [6] was applied to a dataset of numerical variables. However, PCA is not recommended for non-numerical variables. On the other hand, the application reviewed in [4] deals with parametric variables, while Chi-Square could be used for non-parametric variables such as categorical. The limitation is that Support Vector Classification (SVC) cannot be applied to non-numerical data.

Logistic regression is an approach that could be applied to numerical and non-numerical variables. However missing values should be handled before running logistic regression. In this context, reference [11] provides a comparative analysis of five popular methods to handle missing data. Sub-section 2.4 details how the stop-on-fail scenario challenges logistic regression modelling.

The approaches discussed in [7] and [8] could be used with datasets from stop-on-fail test applications. GA, *k*-NN, ONN, and FFNN support numerical and non-numerical variables. One disadvantage of these methods is that the training of a NN model is time consuming. In addition, the biggest limitation of GA is that it cannot guarantee optimality.

Finally, BDT is a useful classification method that works well with numerical and non-numerical variables but is not suitable for stop-on-failure scenarios. This will be detailed in Sub-section 2.4. A similar conclusion is valid for BRF, which is comprised of a collection of BDTs.

2.2 Classification models

SVM, ANN, *k*-NN, GA, fuzzy sets, Decision Tree (DT), and Random Forest (RF) are some data mining techniques used in the electronics industry [12]. DT is commonly used for classification because it is efficient, simple to implement and easy to understand. In the following we cover and compare logistic regression, decision tree and random forest models.

2.2.1 Logistic regression

Logistic Regression is a classification algorithm and is part of the general linear models (GLM) family where logit is the

Table 1: Qualitative Comparison of Analytical Methods

Method	*Variable Type*	*Missing Values*
Tests Correlations [1]	Parametric only	No
Weighted Group LASSO [2]	Parametric only	No
Variations to Group LASSO [3]	Parametric only	Stop-on-Fail
Chi-Square & C-SVC [4]	Parametric (+ numerical)	No
Logistic Regression [5]	Numerical and non-numerical	No
PCA [6]	Numerical only	No
GA, k-NN & ONN [7]	Numerical (+ non-numerical)	No
FFNN [8]	Numerical (+ non-numerical)	No
BDT [9]	Numerical (+ non-numerical)	No
BRF [10]	Continuous (+ non-numerical)	No

link function. The LR is formulated as:

$$\ln\left(\frac{P_L}{1-P_L}\right) = \beta_0 + \sum_{j=1}^{J} \beta_j x_{j,L} + \varepsilon$$

where P_L is the probability that the response variable has a value equal to 1, or {pass} class; $\{\beta\}$ are the coefficients to be estimated; while $\{x\}$ represents input variables; and ε is the predicted error.

Stepwise is a method to find a model with the smallest Akaike Information Criterion (AIC) by removing or adding input variables. The both ways mode has been selected for the study reported with this paper, where input variables could be added or replaced in each iteration.

The glm2 [13] function was used for logistic regression models, and stepAIC [14] for stepwise algorithm.

2.2.2 Decision tree

BDT can be applied to different data, for example to sample populations that consist of n observations made on m variables. The n observations correspond to 2 classes. For example, Table 2 illustrates 4 observations, 2 classes {pass, fail} and 3 variables {Test 1, Test 2, and Test 3}.

Table 2: Dataset from Production Tests

Overall Result	*Test 1*	*Test 2*	*Test 3*
Pass	-76	9A	1
Fail	-80	9A	5
Pass	-66	0	2
Fail	-74	-9A	6

The final model will break the observations into groups, where each of these groups is assigned a predicted class.

The rpart routine [15] branches a tree based on the Gini index, as in:

$$f(p) = p(1-p)$$

where p is the probability of {pass} class and $(1-p)$ the probability of {fail} class, [16].

In order to avoid overfitting the DT built needs to be evaluated using prune.rpart [15]. The prune function evaluates the nested sequence of subtrees supplied by rpart object and recursively snipping off the least important splits based on the amount by which splitting a node improves the relative error, called complexity parameter (CP), [15]. Plots can be generated using rattle R package [17] for easy visualization of the splitting rules and architecture of the tree built.

2.2.3 Random forest

Random Forest generates a robust classification model, it involves building different decision trees and assembling the outputs to make a classification decision. In [18] it was concluded that Boosting and Bagging are two sampling methods that generate accurate results. Furthermore, the Bagging method requires less computational time to build a BDF.

The randomForest [19] routine is an implementation of Breiman's random forest algorithm for classification and regression. The sampling method could be implemented with or without replacement. The bagging sampling method has been selected and used for the study reported with this paper. The use of this RF model enables the calculation of a predicted probability for the classes.

For this paper a threshold of 0.5 was selected to define classes, as below:

$$\text{if } (p > 0.5) \Rightarrow \text{the class is predicted as } \{\text{pass}\}$$

where p is the predicted probability of class{pass}.

2.3 Performance metrics

Models could be evaluated not only from an accuracy point of view also how long it takes to be trained or built, and how simple it is to implement the model as a service. Different performance metrics could be calculated as discussed in [20] and [21]. In this paper we included the Confusion Matrix (CM), Overall Accuracy (OA), Recall (R), Precision (P), F-Score, and Specificity as metrics for model performance evaluation.

1. Confusion Matrix: Contains the calculation of True Pass (TP), True Fail (TF), False Pass (FP), and False Fail (FF) as shown in Table 3. Where N is the number of total instances.

Table 3: Confusion Matrix

		Predicted		
		Pass	Fail	Total
Actual	Pass	TP	FF	$TP + FF$
	Fail	FP	TF	$FP + TF$
	Total	$TP + FP$	$FF + TF$	N

2. Overall Accuracy (OA): Is the proportion of devices that were correctly classified by the model. It is calculated as:

$$OA = (TP + TF)/N$$

3. Recall (R): The ratio of failed devices that were correctly predicted to the actual size of the {fail} class. It is calculated as:

$$R = TF/(TF + FP)$$

4. Precision (P): The ratio of failed devices that were correctly predicted to the predicted size of the {fail} class. It is calculated as:

$$P = TF/(TF + FF)$$

5. F-Score: The harmonic mean of recall and precision. It is calculated as:

$$\text{F-Score} = (2 * P * R)/(P + R)$$

6. Specificity: The ratio of passed devices that were correctly predicted to the actual size of the {pass} class. It is calculated as:

$$\text{Specificity} = TP/(TP + FF)$$

2.4 Challenges of incomplete datasets

When working on big data analytics the presence of missing values is common because of low data quality or data recording processes. Missing data could be randomly distributed across the data or following some pattern. One popular technique to deal with missing values is known as imputation, where data gaps are filled using the available data recorded, for example using the mean value to fill the missing records. Another technique is complete cases method which eliminates the records with incomplete data. Both approaches could generate biased or inaccurate models.

For production tests on stop-on-fail scenario the test program stops as soon as an asset fails a test in the sequence, generating incomplete datasets (Table 4). The risk in using imputation methods is the lack of information to quantify the error. In addition, complete cases method reduces the dataset to a sample with {pass} devices only. The cleaned data generated by this approach is not useful for prognosis. For instance, the binary decision tree built with data cleaned using complete cases consists of 1 node in which all elements are {pass} class. Similar results are obtained when building a logistic regression model or training a random forest.

The next section describes the novel algorithm proposed as a solution to build classification models from data recorded in a stop-on-fail scenario. In Section 4 this algorithm is applied to build a logistic regression model, train a BDT and a BRF, which are algorithms that require complete datasets for its training.

Table 4: Dataset Before Cleaning. Stop-on-Fail Test Sample

Overall Result	*Test 1*	*Test 2*	*Test 3*
Pass	-76	9A	1
Fail	-80	9A	
Pass	-66	0	2
Fail	-74		

3 Iterative within-set model building

When using a subset of tests, it is expected that the cleaned dataset has an adequate mixture of classes as illustrated in Table 5. Note that this data is the result of cleaning with complete cases method the dataset in Table 4 but considering Test 1 and Test 2 only. With the data in Table 5 it is feasible to build a classification model in a stop-on-fail scenario. The classification model could be used to reduce the test set by predicting the output of a specific tests by using the result of previous tests in the sequence. This idea motivated the proposed algorithm where DTs are built by adding one test in the sequence in each iteration until a model with defined target accuracy level is built. The same algorithm could be used to build random forest, logistic regression or other similar models.

Table 5: Dataset After Using First 2 Tests Subset for Cleaning.

Overall Result	*Test 1*	*Test 2*	*Test 3*
Pass	-76	9A	
Fail	-80	9A	
Pass	-66	0	

Figure 1 illustrates the method proposed to build an iterative within set models and its evaluation. This algorithm consists of three main steps:

1. Data gathering and pre-processing;
2. Model building and metrics calculation;
3. Model evaluation and improvement.

1. Data gathering and data pre-processing
   (a) Test to be Modelled: From the test sequence select a test, ***Test_x***, to which a model would be built in order to eliminate that test in the sequence. We recommend selecting a test that the assets frequently fail
   (b) Response Variable: Generate the response variable, ***Y_Test_x***, associated to *Test_x*. Where:

$$Y\_Test\_x = \begin{cases} Pass & \text{if the device pass } Test\_x \\ Fail & \text{in other case} \end{cases}$$

   (c) Subset of Tests: The first iteration consists of the data recorded for the first two tests in the sequence, ***D_2***. Similar ***D_k*** contains a subset of the first *k* tests.

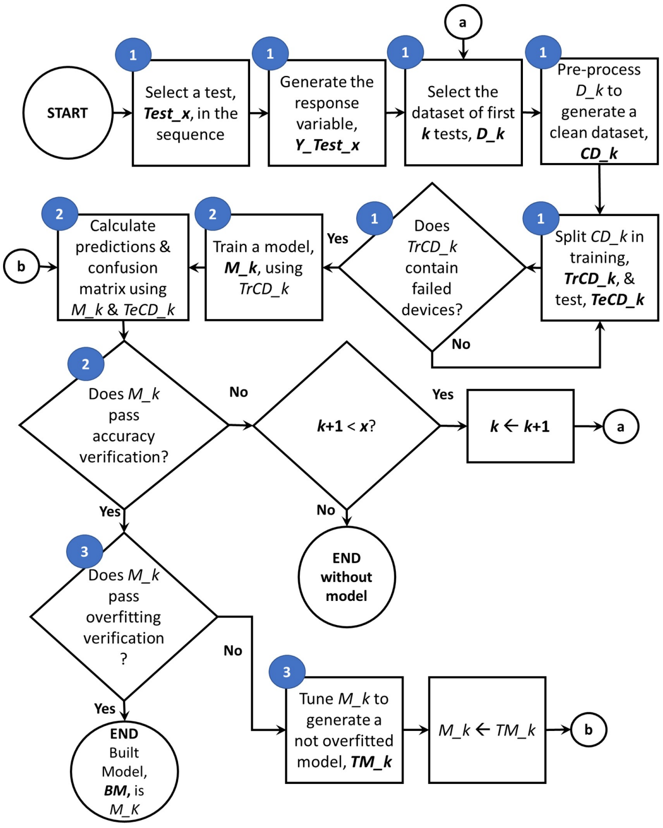

START
1
Select a test, Test_x, in the sequence
1
Generate the response variable, Y_Test_x
a
1
Select the dataset of first k tests, D_k
1
Pre-process D_k to generate a clean dataset, CD_k
2
Calculate predictions & confusion matrix using M_k & TeCD_k
b
2
Train a model, M_k, using TrCD_k
Yes
1
Does TrCD_k contain failed devices?
No
1
Split CD_k in training, TrCD_k, & test, TeCD_k
2
Does M_k pass accuracy verification?
No
k+1 < x?
Yes
k ← k+1
a
Yes
No
END without model
3
Does M_k pass overfitting verification ?
No
Yes
END Built Model, BM, is M_K
3
Tune M_k to generate a not overfitted model, TM_k
M_k ← TM_k
b

Figure 1: Model Building Algorithm Flowchart.

   (d) Data Cleaning: The complete cases method was used to clean *D_k*. The cleaned dataset is called ***CD_k***.
   (e) Dataset Split into Training and Test Datasets: *CD_k* dataset is split in two independent sets. The training dataset, ***TrCD_k***, is used to build the model, ***M_k***, and the test dataset, ***TeCD_k***, is used to calculate model accuracy.
   (f) Training Sample Evaluation: As shown in Subsection 2.4 it is important that *TrCD_k* contains {pass} and {fail} devices. When *TrCD_k* does not contain an adequate mixture of classes the step 1.c must be revisited.
2. Model building and metrics calculation
   (a) Model Building. Use *TrCD_k* to build *M_k* model for *Test_x*. The algorithm is flexible to build other models but here is illustrated for:
      i. Logistic Regression
      ii. Decision Tree
      iii. Random Forest
   (b) Accuracy Calculation: We recommend calculating metrics discussed in Sub-section 2.3:
      i. Confusion Matrix
      ii. Overall Accuracy

        iii. Recall
        iv. Precision
        v. F-Score
        vi. Specificity

    The steps 1.c to 2.b are repeated until *M_k* accuracy is at least as good as the target level set or until there are no more tests before *Test_x* in the sequence.

3. Model evaluation and improvement
    (a) Over-fitting Evaluation:
        i. For Logistic Regression: The model *M_k* has the smallest AIC.
        ii. For Decision Tree: All complexity parameter values in vector *cp_M_k* should be above 0.01.
        iii. For Random Forest. This step is not needed as was detailed in Sub-section 2.2.3.
    (b) Tuning the Model:
        i. For Logistic Regression: The model *M_k* should be tuned using stepwise method.
        ii. For Decision Tree: If at least one element of *cp_M_k* is below 0.01, the tree should be pruned. During pruning the least important splits of *M_k* are snipped off.
        iii. For Random Forest. This step is not needed as was detailed in Sub-section 2.2.3.

    The step 2.b should be revisited with tuned model, ***TM_k***, and *TrCD_k* dataset.

The built model, ***BM***, generated with this algorithm could be used to predict the result of a specific test based on the results of previous tests. When having an accurate model to predict the result of a specific test, it could be eliminated from the sequence and hence reduce the time cycle of testing process. However, it is possible that the previous tests to *Test_x* are not significant predictors to model *Test_x*, hence the algorithm would terminate without a model and *Test_x* should not be eliminated from the sequence.

It is relevant to highlight that model reliability and its maintenance are important when the model is used as a service, because the process could change due to modification in the production line, materials, or wear of machinery or assets, for instance. A model maintenance phase is illustrated in Figure 2. and consists of three phases:

1. Current data gathering and pre-processing:
    (a) New dataset, ***ND***, should be gather
    (b) Response Variable: Generate the response variable, ***Y_Test_x***, associated to *Test_x*. Where:

$$Y\_Test\_x = \begin{cases} Pass & \text{if the device pass } Test\_x \\ Fail & \text{in other case} \end{cases}$$

    (c) Pre-process *ND* dataset following the same data cleaning strategy as when building the model *BM*. The clean dataset is called ***CND***
    (d) Split *CND* in two independent sets: re-training, ***RTrCND***, and validation, ***VCND***, datasets
2. Model validation and metrics calculation. Metrics and confusion matrix are calculated using *BM* which is the available model and dataset *VCND*.
3. Model re-train. If *BM* model is not as accurate as the target a new model should be generated using the *RTrCND* dataset and algorithm in Figure 1.

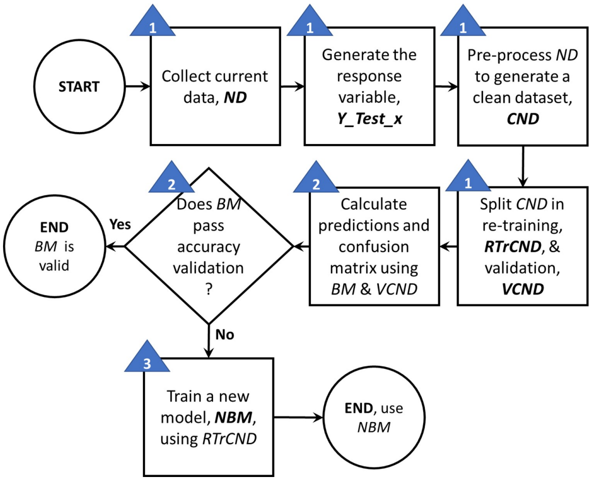

START
1
Collect current data, ND
1
Generate the response variable, Y_Test_x
1
Pre-process ND to generate a clean dataset, CND
2
Does BM pass accuracy validation ?
Yes
END BM is valid
2
Calculate predictions and confusion matrix using BM & VCND
1
Split CND in re-training, RTrCND, & validation, VCND
No
3
Train a new model, NBM, using RTrCND
END, use NBM

Figure 2: Model Validation Algorithm Flowchart.

4 Stop-on-fail production test application

Historical data from a production test process of an electronic device, in a stop-on-fail scenario, is used to demonstrate and validate how the proposed algorithm can compact a test set by building a predictive classification model for one of the sequence elements.

The methodology followed to analyse the production test set includes:

1. To understand the problem and define the objective of the analysis
2. To define the strategy to pre-process the data for improving data quality
3. To evaluate association between tests and other variables recorded in the production test process

4. To build the model and its evaluation
5. To validate the reliability of the model

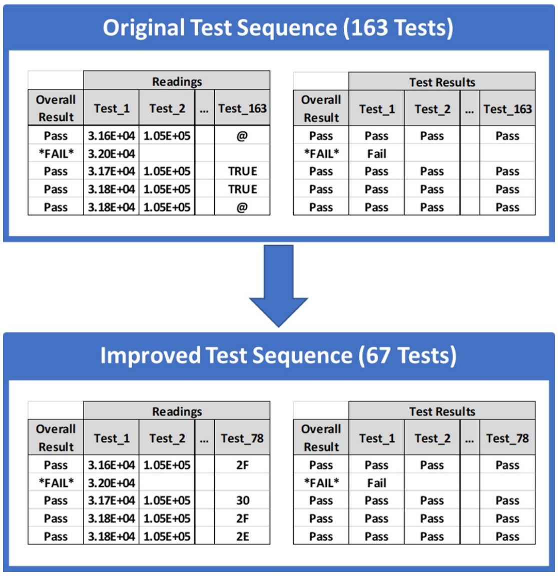
More details about data pre-processing strategy followed and variables association analysis can be found on the conference paper preceding this paper [22].

Three different models were generated for one test in the sequence that the faulty devices frequently fail. Logistic regression, binary decision tree, and random forest algorithms were used. The models were compared based on performance metrics (Sub-section 2.3), and the computational time required to build the model.

4.1 Problem understanding and objective definition

As part of the manufacturing process each asset is tested by an automated sequence, which performs 163 readings including resistance, voltage, current, or a logical test. Each reading is compared to a upper limit, lower limit, or a constant value. If the value is outside the respective limit then the test is failed. The sequence is interrupted after a fail in one of the test items and is not executed for the remaining tests of the test set.

The main objective is to evaluate how the testing process could be reduced, by identifying redundant tests, but also by using an analytic to predict whether a test will fail based on results of previous tests in the sequence without performing the reading (Fig. 3).



Original Test Sequence (163 Tests)

	Readings			
Overall Result	Test_1	Test_2	...	Test_163
Pass	3.16E+04	1.05E+05		@
FAIL	3.20E+04			
Pass	3.17E+04	1.05E+05		TRUE
Pass	3.18E+04	1.05E+05		TRUE
Pass	3.18E+04	1.05E+05		@

	Test Results			
Overall Result	Test_1	Test_2	...	Test_163
Pass	Pass	Pass		Pass
FAIL	Fail			
Pass	Pass	Pass		Pass
Pass	Pass	Pass		Pass
Pass	Pass	Pass		Pass

Improved Test Sequence (67 Tests)

	Readings			
Overall Result	Test_1	Test_2	...	Test_78
Pass	3.16E+04	1.05E+05		2F
FAIL	3.20E+04			
Pass	3.17E+04	1.05E+05		30
Pass	3.18E+04	1.05E+05		2F
Pass	3.18E+04	1.05E+05		2E

	Test Results			
Overall Result	Test_1	Test_2	...	Test_78
Pass	Pass	Pass		Pass
FAIL	Fail			
Pass	Pass	Pass		Pass
Pass	Pass	Pass		Pass
Pass	Pass	Pass		Pass

Figure 3: Test Sequence Reduction.

4.2 Data pre-processing

Data from running production tests to electronic devices was collected from large number of .csv (3,766) or .fil (12,727) files, each one containing information for one or more devices. For each device there are records for the respective test results of each test item in the sequence, and also the overall test result. When a device fails, no records for the subsequent tests are obtained as the testing stops at that point. Data format and quality was improved for better data handling and analysis results (more details in [22]). After following the cleaning process, the dataset used for model building contains the results of testing 68,168 devices, where 50,882 assets passed and 17,286 failed.

More current data was collected for model reliability assesment. It contains the results of 235,132 devices, where 171,131 assets passed and 64,001 failed the production test (Fig. 4). It is important to highlight that these datasets contain missing values, hence additional cleaning steps are needed before analysing data.

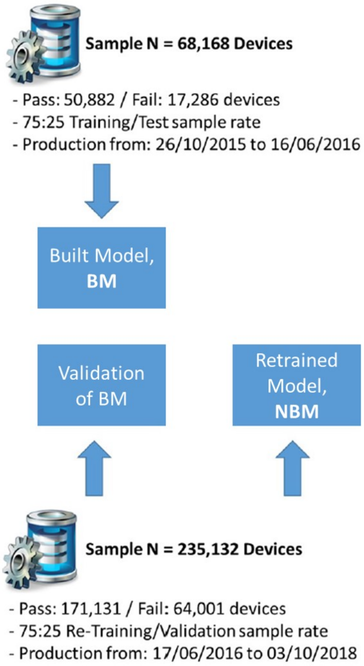

Sample N = 68,168 Devices
- Pass: 50,882 / Fail: 17,286 devices
- 75:25 Training/Test sample rate
- Production from: 26/10/2015 to 16/06/2016
Built Model, BM
Validation of BM
Retrained Model, NBM
Sample N = 235,132 Devices
- Pass: 171,131 / Fail: 64,001 devices
- 75:25 Re-Training/Validation sample rate
- Production from: 17/06/2016 to 03/10/2018

Figure 4: Datasets.

4.3 Variables association analysis

Identifying relations between variables is useful not only to determine redundant tests, but also because while applying machine learning classification or clustering techniques the attributes with the least contributions to the resulting classification or clustering will act as noise. Hence removing least contributors will improve the model built. Chi-Square Test and Pearson Correlation Coefficient were used to analyse the association between categorical variables and continuous variables, respectively. 26 pairs of tests were identified as highly correlated [22], hence model over-fitting evaluation is needed.

4.4 Model predictive building and evaluation

The dataset comprised of 68,168 devices was used for this analysis. After pre-processing the dataset and evaluating the association between variables or tests we applied the proposed algorithm to compact the test sequence. The ratio used to split the data in Training/Test is 75:25. The model accuracy targets are:

- The OA ≥ 90%
- Recall ≥ 90%
- Precision ≥ 90%
- F-Score ≥ 90
- Specificity ≥ 90%

The algorithm was used to generate LR, DT and RF models for *Test*_114, which is one of the tests that frequently fail low-quality devices.

4.4.1 Logistic regression

The first model that successfully achieved all accuracy metrics is:

glm2(formula = Y_Test_x ∼ Test_106, family =
binomial(link = logit), data = train)

Using the confusion matrix (Table 6) the performance of this model is as follows:

- The OA achieved is 100%
- Recall = 100%
- Precision = 100%
- F-Score = 100%
- Specificity = 100%
- Time to build the model: 23.85 hours

.

The model LR_106 satisfies all performance metrics, but it is computationally complex, it took 23.85 hours to build LR_106, in addition to the time taken to build the models of previous iterations.

Table 6: Confusion Matrix. LR_106

		Predicted		
		Pass	Fail	Total
Actual	Pass	11232	0	11232
	Fail	0	97	97
	Total	11232	97	11329

4.4.2 Decision tree

In iteration 106 a model was built that satisfies all accuracy metrics targets. It consists of a root node and two leaves (Fig. 5). This BDT classifies the devices as follows:

$$Y\_Test\_114 = \begin{cases} Pass & \text{If } Test\_106 < -62 \\ Fail & \text{In other case} \end{cases}$$

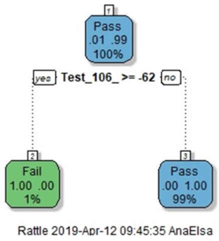


Figure 5: Decision Tree DT_106.

Using the confusion matrix (Table 7) the performance of this model is as follows:

- The OA achieved is 100%
- Recall = 100%
- Precision = 100%
- F-Score = 100%
- Specificity = 100%
- Time to build the model: 2.98 seconds

From the associated confusion matrix (Table 7) and accuracy metrics results we conclude that the model DT_106 could be used to predict Test_114, hence this test could be eliminated from the test sequence. Furthermore the complexity of this model was low.

Table 7: Confusion Matrix. DT_106

		Predicted		
		Pass	Fail	Total
Actual	Pass	11232	0	11232
	Fail	0	97	97
	Total	11232	97	11329

4.4.3 Random forest

In each iteration was generated a forest of 100 trees, using the dataset with size *N*=68,168. Sampling method of the Training set was done with replacement. In iteration 106 an accurate random forest was built for Test_114.

From the associated confusion matrix (Table 8) the accuracy metrics results are as follows:

- The OA achieved is 100%
- Recall = 100%
- Precision = 100%
- F-Score = 100%
- Specificity = 100%
- Time to build the model: 1.125 minutes

Table 8: Confusion Matrix. RF_106

		Predicted		
		Pass	Fail	Total
Actual	Pass	11232	0	11232
	Fail	0	97	97
	Total	11232	97	11329

We conclude that the RF_106 model could be used for predicting Test_114. Its training was not complex and satisfied performance targets.

4.4.4 Models comparison

The three methods were able to build an accurate model for Test_114, in addition they were consistent in identifying Test_106 as predictor. In Figure 6 we see how the model overall accuracy increased to 100% in the last iteration for the three approaches. This because in the iteration 106 was included a significant predictor. Furthermore, the OA was close to 100% in each iteration. This is because the imbalance in proportion Pass/Fail, and that the models in all iterations are accurate to prognosis {pass} class, as can be seen in Figure 7.

When having a highly imbalanced dataset it is recommended to analyse the model performance for each class. We recommend using Specificity (Fig. 7) for {pass} class and for {fail} class the usage of Recall (Fig. 8), Precision (Fig. 9), and F-Score (Fig. 10).

In Figures 8-10 we can see that the three methods were improving its capabilities to correctly classify faulty devices when the iterations increase up to reach 100% accuracy in iteration 106.

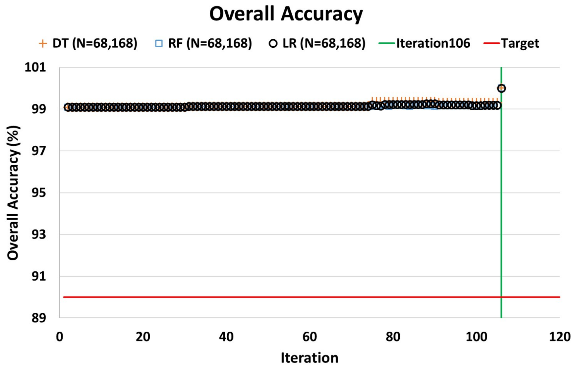


Figure 6: LR, DT and RF Overall Accuracy by Iterations

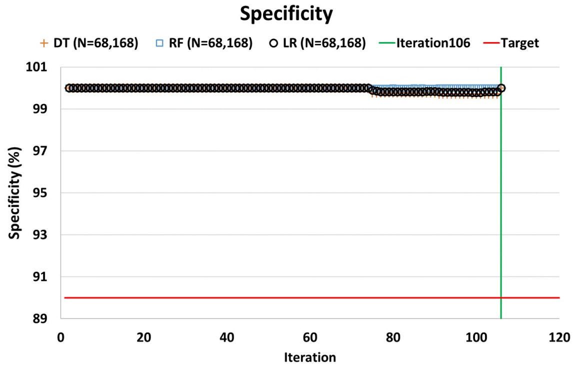


Figure 7: LR, DT and RF Specificity by Iterations

With regards computational complexity, in Figure 11 we can see that LR is much more complex than the other two approaches. Note that LR complexity is illustrated in minutes, while DT and RF are in seconds. In addition, for LR the time complexity grows exponentially when adding more potential predictors.

Figure 11 shows how the computational complexity of DT and RF fall in iteration 106, which is the last one. When

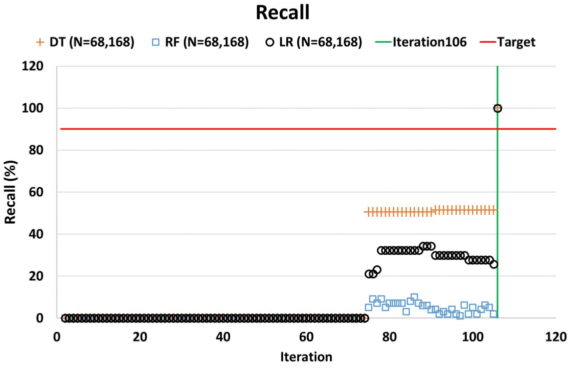


Figure 8: LR, DT and RF Recall by Iterations

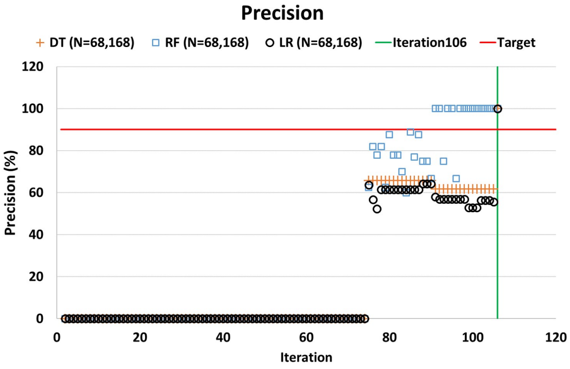


Figure 9: LR, DT and RF Precision by Iterations

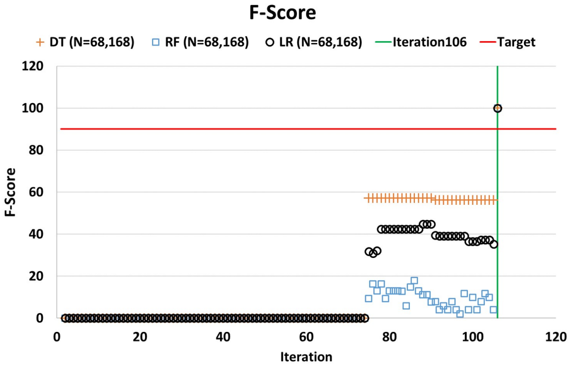


Figure 10: LR, DT and RF F-Score by Iterations

including a highly significant predictor (Test_106) in the tree(s), there are no more improvement in the Gini index when adding other variables, hence the training ends. A similar effect can be seen for Test_75 in iteration 75th and 76th for Logistic Regression method, where the stepwise phase for improving the Logistic Regression model was not complex. In contrast, iteration 106 was highly computationally complex. Since the association between predictive candidates impacted the model complexity.

From the three methods, we do not recommend using logistic regression because its complexity increases exponentially as the number of predictors increase when the predictors are interrelated. In our application the potential predictors are 163 and some tests are highly associated. On the other hand, decision tree and random forest methods performed very well in complexity and accuracy. Furthermore, a single tree is simpler than a forest, hence we recommend to use decision trees.

The accurate and simple model proposed to predict Test_114 is as follows:

$$Y\_Test\_114 = \begin{cases} Pass & \text{If } Test\_106 < -62 \\ Fail & \text{In other case} \end{cases}$$

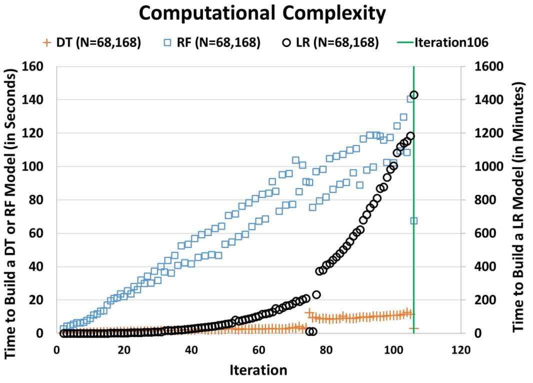


Figure 11: LR, DT and RF Computational Complexity by Iterations

4.5 Model predictive validation and re-building

The software used to automate testing procedures is frequently subject to software certification requirements, furthermore any change to it must require a re-certification of the new software version. Hence before implementing any change it is important to assure that the improvement is robust. Based on that, the reliability of model DT_106 was assessed using up-to-date data (Fig. 4). The new dataset was pre-processed as detailed in Sub-section 4.2.

The algorithm for model validation proposed in Figure 2 was followed using the up-to-date data and a confusion matrix was calculated (Table 9). The model accuracy is as follows:

- The OA achieved is 100%
- Recall = 100%
- Precision = 100%

- F-Score = 100%
- Specificity = 100%

Table 9: Confusion Matrix. Validating DT_106

		Predicted		
		Pass	Fail	Total
Actual	Pass	20954	0	20954
	Fail	0	47	47
	Total	20954	47	21001

We conclude that model DT_106 is still valid for the actual production process conditions and we recommend implementing changes in the testing software to eliminate the Test_114. In addition, more tests could be analysed following the algorithm proposed (Fig. 1) and if accurate predictive models are built, more reduction could be achieved in the testing procedure.

5 Conclusions

In this paper we presented an algorithm to generate robust models that could be used for reducing a test set in a stop-on-fail test scenario with parametric and non-parametric tests. This algorithm is a strategy to manage incomplete datasets. The development was enabled through performing a successful data mining analysis on the production test data. The analysis covered data gathering in two different raw format, data pre-processing for data quality improvement, an association analysis of input variables, the proposed novel algorithm application, its verification and reliability assessment. The flexible algorithm proposed was evaluated by building logistic regression, decision tree, and random forest models. In addition, the benefit of using Decision Tree method over the other two approaches was discussed.

The outlined algorithm, based on the use of classification models, is found to be adequate for applications with incomplete datasets and can be employed in real production lines to offer accurate and reliable models to compact a test sequence even for stop-on-fail scenario. Embedding the model generated with the presented algorithm has the potential to enable substantial cost savings as a result of production test set compaction. For the discussed data and application, we have illustrated a robust and accurate model that enables the elimination of one test that is frequently failed by faulty components.

Note

This paper is an extended version of a presentation given at the University of Essex, Southend, UK in August 2018, as part of the IEEE International Conference on Computing, Electronics & Communications Engineering 2018 (iCCECE '18) conference [23].

Acknowledgment: The authors would like to acknowledge the support from Microsemi (A Microchip company) for providing the dataset and for sharing their time and expertise with us.

We would like to express our deep gratitude to Prof. Marc Cavazza (Head of the School of Computing and Mathematical Sciences at the University of Greenwich) for his valuable and constructive suggestions during the planning and development of this research work.

References

[1] Chen M., Orailoglu A., Test cost minimization through adaptive test development, Proceedings of IEEE International Conference on Computer Design (2008, California, USA), 2008, 234-239

[2] Hsu C., Lin F., Cheng K., Zhang W., Li X., Carulli J. M., et al., Test data analytics — exploring spatial and test-item correlations in production test data, Proceedings of IEEE International Test Conference (2013, California, USA), 2013, 1-10

[3] Lin F., Hsu C. K., Cheng K. T., Learning from production test data: correlation exploration and feature engineering, Proceedings of IEEE 23rd Asian Test Symposium (2014, Hangzhou, China), 2014, 236-241

[4] Sumikawa N., Drmanac D. G., Wang L. C., Winemberg L., Abadir M. S., Forward prediction based on wafer sort data — a case study, Proceedings of IEEE International Test Conference (2011, California, USA), 2011, 1-10

[5] Pham H. V., Demidenko S. N., Merola G. M., Eliminating re-burn-in in semiconductor manufacturing through statistical analysis of production test data, Proceedings of IEEE International Instrumentation and Measurement Technology Conference (2017, Turin, Italy), 2017, 1-6

[6] Nahar A., Daasch R., Subramaniam S., Burn-in reduction using principal component analysis, Proceedings of IEEE International Conference on Test (2005, Texas, USA), 2005, 155-165

[7] Stratigopoulos H. G., Drineas P., Slamani M., Makris Y., RF specification test compaction using learning machines, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2010, 18(6), 998-1002

[8] Záluský R., Ďuračková D., Stopjaková V., Brenkuš J., Mihálov J., Majer L., Production test-based classification of antennas using the feed-forward neural network, Proceedings of 24th International Conference Radioelektronika (2014, Bratislava, Slo-

vakia), 2014, 1-4

[9] Biswas S., Blanton R. D., Statistical test compaction using binary decision trees, IEEE Design & Test of Computers, 2006, 23(6), 452-462

[10] Biswas S., Blanton R. D., Reducing test execution cost of integrated heterogeneous systems using continuous test data, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2011, 30(1), 148-158

[11] Lv S., Kim H., Zheng B., Jin H., A review of data mining with big data towards its applications in the electronics industry, Applied Sciences, 2018, 8(582), 1-34

[12] Meeyai S., Logistic regression with missing data: a comparisson of handling methods and effects of percent missing values, Journal of Traffic and Logistics Engineering, 2016, 4(2), 128-134

[13] Marschner I., Donoghoe M., glm2: fitting generalized linear models, R package version 1.2.1, August 2018, https://CRAN.R-project.org/package=glm2

[14] Venables W., Ripley B., stepAIC {MASS}, Package MASS version 7.3-51.1, https://stat.ethz.ch/R-manual/R-devel/library/MASS/html/stepAIC.html

[15] Therneau T., Atkinson B., Ripley B., rpart, R package version 4.1-13, February 2018, https://cran.r-project.org/web/packages/rpart/rpart.pdf

[16] Therneau T., Atkinson B., Ripley, B., An introduction to recursive partitioning using the RPART routines, Mayo Clinic, February 2018. https://cran.r-project.org/web/packages/rpart/vignettes/longintro .pdf

[17] Williams G., Culp M. V., Cox E., Nolan A., White D., Medri D., et al., rattle, R package version 5.1.0, September 2017, https://CRAN.R-project.org/package=rattle

[18] Quinlan J., Bagging, boosting and C4.S, Proceedings of the thirteenth national conference on Artificial intelligence (1996, Oregon, USA), 1996, 725-730

[19] Breiman L., Cutler A., Liaw A., Wiener M., randomForest, R package version 4.6-14, March 2018, https://www.stat.berkeley.edu/∼breiman/Random Forests/

[20] Elhamahmy M., Elmahdy H., Saroit I., A new approach for evaluating intrusion detection system, Artificial Intelligent Systems and Machine Learning, 2010, 2(11), 290-298

[21] Ashraf N., Ahmad W., Ashraf R., A comparative study of data mining algorithms for high detection rate in intrusion detection system, Annals of Emerging Technologies in Computing, 2018, 2(1), 49-57

[22] Hinojosa A., Stoyanov S., Data driven predictive model to compact a production stop-on-fail test set for an electronic device, Proceedings of International Conference on Computing, Electronics & Communications Engineering (2018, Southend, United Kingdom), 2019, 59-64