Data Driven Predictive Model to Compact a Production stop-on-fail Test Set for an Electronic Device

Authors: Ana Elsa Hinojosa Herrera, Stoyan Stoyanov

Conference: 2018 International Conference on Computing, Electronics &Communications Engineering (iCCECE), University of Essex, Southend, UK, 16-17 August 2018.

Paper Acceptance Date: 21/06/2018

GALA Repository Upload Date: 16/07/2018

Data Driven Predictive Model to Compact a Production stop-on-fail Test Set for an Electronic Device

Ana Hinojosa
Computational Mechanics and Reliability Group
University of Greenwich
London, United Kingdom
aehinojosa@ieee.org

Stoyan Stoyanov
Computational Mechanics and Reliability Group
University of Greenwich
London, United Kingdom
S.Stoyanov@greenwich.ac.uk

***Abstract*— Decision Tree is a popular machine learning algorithm used for fault detection and classification in the industry. In this paper, the modelling technique is used to compact a production test set defined for quality assurance of an electronic asset. The novelty of this work is in the proposed method that builds in an iterative way decision trees until an accurate predictive model that meets classification accuracy target in a stop-on-fail test scenario. Generated test data is characterized with missing values which is a major challenge to the traditional use of decision trees. The developed computational procedure handles this application-specific data attribute. Exemplary results show that the method is able to significantly reduce a production test set with parametric and non-parametric tests, and generate a truthful prognostic model. In addition, the method is computationally efficient and easy to implement. It could also be combined with another test compaction strategies such as variables association analysis. Furthermore, the method proposed offers the flexibility of exploring the trade-off between the number of removed tests from the production test set and the prediction accuracy. The results can enable production costs reduction without impacting quality detection accuracy. The paper details and provides discussions on the advantages and limitations of the proposed algorithm.**

Keywords*— *Decision Tree, Production test set compaction, Incomplete dataset, stop-on-fail test, Electronic device

I. INTRODUCTION

The fourth industrial revolution is transforming electronic industries into smart factories, where machines are connected to another machines, people or assets. Different type of data is generated at high speed, big volume and with uncertainty. Cloud technology, Internet of Things and Artificial Intelligence are pillars of these transformations.

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

There are successful business cases where testing processes were compacted 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 Group LASSO method were used in [2] and [3] to identify tests which could be predicted by a linear combination of other tests. Furthermore, the last one covers 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

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

Different Artificial Intelligence (AI) techniques are useful for this application. In reference [8], a combination of a multi-objective Genetic Algorithm (GA) for feature selection, and *k*-Nearest-Neighbours (*k*-NN) with Ontogenic Neural Network (ONN) for prediction is presented. Reference [9] details a successful application of 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 which output could be predicted using the results from another test [10]. This methodology is advantageous from computational time point of view, since considerably less training time is required to build a tree than to train a network. In contrast, sometimes 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.

With regards the computational resources used, the most common software for data mining and big data applications in electronics industry are Excel, Visual Basic, C, C++, Python, MATLAB, WEKA, SAS, SPSS, Minitab, Statistics, and R [11].

In this paper we propose a novel classification method based on Decision Trees to compact production test sets in a stop-on-fail scenario. The approach could be applied with both numerical and non-numerical test results. In addition, the method proposed offers the flexibility of exploring the trade-off between the number of removed tests and the prediction accuracy. The data mining approach was written in an R script which covers data gathering, data pre-processing, variables association analysis, and iterative within-set decision tree model building.

The remainder of this paper is organized as follows. In Section II we compare different data analysis techniques, in particular decision trees, and challenges of using these

methods for modelling stop-on-fail test scenarios. In Section III we present our novel method. The method is illustrated in Section IV using historical production tests data of an electronic device, and Section V concludes the paper.

II. DATA ANALYSIS TECHNIQUES

A. Qualitative Comparison of Techniques

The usage of different techniques for reducing test sequence in electronics manufacturing is summarized in Table I. In addition, an evaluation if those methods could be used with datasets containing numerical and non-numerical variables is included. On the other hand, missing data was evaluated only in reference [3] where the case of stop-on fail scenario was covered. A data pre-processing is needed to handle missing values when using any of the other methods.

TABLE I. QUALITATIVE COMPARISON WITH TECHNIQUES

Method	***Variable Type***	***Missing Values***
Correlations between each pair of tests [1]	Parametric only	Not evaluated (a handling method is needed)
Weighted Group LASSO [2]	Parametric only	Not evaluated (covered in [3])
Variations to Group LASSO [3]	Parametric only	Stop-on-Fail
PCA [4]	Numerical only	Not evaluated (a handling method is needed)
Chi-Square & C-SVC [5]	Parametric (useful for numerical)	Not evaluated (a handling method is needed)
Logistic Regression [6]	Numerical and non numerical	Not evaluated (a handling method is needed)
GA, k-NN & ONN [8]	Numerical (useful for non-numerical)	Not evaluated (a handling method is needed)
FFNN [9]	Numerical (useful for non-numerical)	Not evaluated (a handling method is needed)
BDT [10]	Numerical (useful for non-numerical)	Not evaluated (a handling method is needed)

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 [4] 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 [5] deals with parametric variables, while Chi-Square could be used for non-parametric variables, in particular categorical ones. The limitation is that SVM 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 [7] provides a comparative analysis of five popular methods.

The approaches discussed in [8] and [9] could be used with datasets from stop-on-fail test applications because 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, furthermore, the solution quality deteriorates on big datasets.

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 II.C.

B. Classification Models

SVM, ANN, *k*-NN, GA, fussy sets and Decision Tree (DT) are some data mining techniques used in electronics industry [11]. DT is commonly used for classification because its efficiency, simplicity to be implemented and easy to be understood by humans.

BDT can be applied to different data, in particular to sample populations that consist of *n* observations made on *m* variables. The *n* observations correspond to 2 classes. For example, Table II illustrate 4 observations, 2 classes {pass, fail} and 3 variables {Test 1, Test 2, and Test 3}. The final model will break the observations into groups, each of these groups is assigned a predicted class as in Fig. 1. The method is composed of rules which are built recursively by repeated splits of the training dataset following these 2 steps:

1. Calculate the impurity of each node measuring the mixture of classes in the sample covered by a node. Looking for the major reduction of the impurity metric is selected the variable which best splits the data into two groups.
2. Data is split in two sub-groups based on the rule identified in previous step.

Step 1 and 2 is applied recursively to each sub-group until the subgroups reach a minimum size or there is no more improvement in the measurement of impurity.

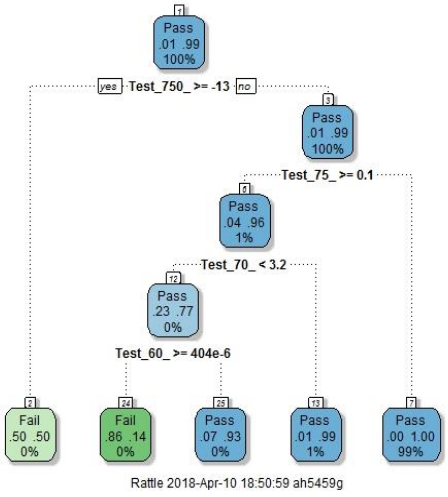

Fig. 1. Example of Binary Decision Tree

The ***rpart*** routine [13] branches a tree based on the Gini index, as in:

$$f(p) = p\,(1 - p) \tag{1}$$

where p is the probability of {pass} class and $(1\text{-}p)$ the probability of {fail} class [12].

In order to avoid overfitting the DT built needs to be evaluated using ***prune.rpart*** [13]. 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 improved the relative error, called complexity parameter (cp). [13]

For easy visualization of the splitting rules and architecture of the tree built, DT plots can be generated using rattle R package [14].

C. *Stop-on-fail Challenges with Decision Tree Model*

When working on big data analytics is common the presence of missing values because of data low quality or process definition. This data attribute is present in production test on stop-on-fail scenario, where the test program stops as soon as an asset fails a test in the sequence, and the remaining tests will not be executed.

There are some methods proposed to deal with missing values. One common technique, known as imputation, fills the data gaps, for example by using the most frequent value. On the other hand, the complete cases method eliminates the records with incomplete data. Both approaches could generate biased or inaccurate models. In stop-on-fail production tests, the second approach reduces the dataset to a sample with pass devices only. Table III is the result of cleaning the dataset provided in Table II. Furthermore, dataset of Table III is not useful for prognosis, because the model consists of 1 node for which all elements are {pass} class (Fig. 2).

The next section describes the novel algorithm proposed to build a BDT in a stop-on-fail scenario.

TABLE II. DATASET BEFORE CLEANING

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

TABLE III. DATASET AFTER CLEANING

Overall Result	**Stop-on-fail Test Sample**		
	Test 1	***Test 2***	***Test 3***
Pass	-76	9A	1
Pass	-66	0	2

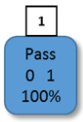

Fig. 2. BDT Clean Dataset on stop-on-fail Test Set

III. ITERATIVE WITHIN-SET DECISION TREE

Using a subset of tests in the sequence the cleansed dataset is prepared to have adequate mixture of classes as illustrated in Table IV. Note that this is the result of cleaning dataset of Table II but taking into account Test 1 and Test 2 only. With the data in Table IV it is feasible to build a BDT in a stop-on-fail scenario. The decision tree could be used to compact the test set by identifying the subset of tests needed to build the classification model and dropping the consecutive ones. This idea motivated the proposed algorithm where DTs are built adding one test in the sequence in each iteration until a model with defined target accuracy level is built.

TABLE IV. DATASET AFTER CLEANING – FIRST 2 TESTS SUBSET

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

The novel method proposed to build an iterative within-set BDT and its evaluation is illustrated in Fig. 3. This algorithm consists of three main phases: (1) data gathering and data pre-processing, (2) BDT building and its accuracy calculation, and (3) evaluation of the BDT generated after running iterations of phases (1) and (2).

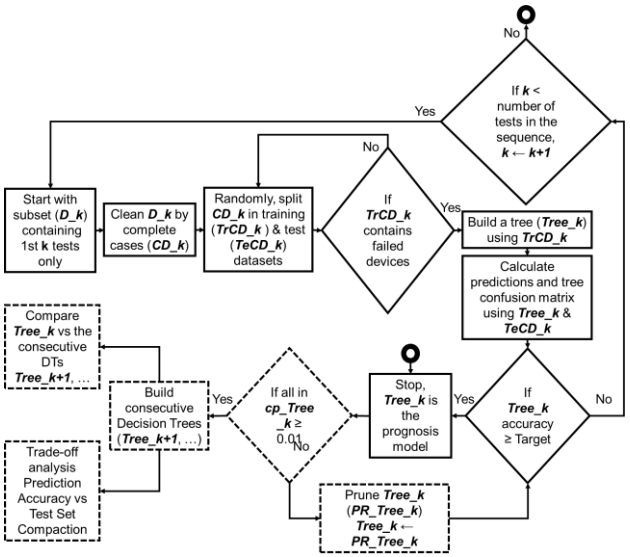

Fig. 3. Algorithm Flowchart

A. *Data Gathering and Pre-Processing*

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

2) Data Cleansing: Complete cases were used to clean *D_k*. The cleansed dataset is called *CD_k.*

3) Dataset Split in Training and Test Datasets: CD_k dataset is split in two independent sets. The training dataset (*TrCD_k*) is used to build the decision tree, and the test dataset (*TeCD_k*) is used to calculate a confusion matrix to evaluate the DT accuracy.

4) Training Sample Evaluation: As shown in Section II is important that training dataset (*TrCD_k)* contains pass

and fail devices. When *TrCD_k* does not contain an adecuate mixture of classes the step 3 should be revisited.

B. Decision Tree Building and Accuracy Calculation

5) Decision Tree Building: Using rpart routine a DT *(Tree_k)* is built with *TrCD_k* training ataset.

6) Accuracy Calculation: The accuracy of *Tree_k* is the % of matches between Predicted Class and Actual Class, i.e. true positives + true negatives in a confusion matrix.

The steps 1 to 6 are repited until the *Tree_k* accuracy is at least at the target level set or until there are no more tests in the sequence.

C. Decision Tree Evaluation

7) Overfitting Evaluation: All complexity parameter values in vector *cp_Tree_k* should be above 0.01.

8) Pruning Tree: If at least one element of *cp_Tree_k* is below 0.01, the tree should be pruned using prune function, included in rpart package. During pruning the least important splits of *Tree_k* are snipped off. After prunning, Step 6 should be revisited with *PR_tree_k* dataset.

9) Consistency Evaluation: To evaluate the robustness of *Tree_k* built in the last iteration, its architecture is compared against the architecture of the consecutive ones.

10) Trade-off Analysis: The accuracy target could be evaluated visualizing its growth over the iterations.

IV. STOP-ON-FAIL PRODUCTION TEST APPLICATION

Historical data from a production test process of an electronic device, in a stop-on-fail scenario, used to demonstrate and validate the proposed approach. This dataset enables to illustrate the algorithm proposed to compact a test set by building a predictive classification model with a subset of the tests.

The methodology followed to analyse a production test set includes firstly the problem understanding and definition of the analysis objective, secondly data pre-processing strategy to improve the quality of the data, thirdly an analysis to evaluate association between tests and other variables recorded in the production test process, to identify redundant tests but also to detect data noise generated by the test process, and finally model building and its evaluation

A. Problem Understanding and Objective Definition

As part of the manufacturing process each asset is tested by an automated sequence which, in this instance, consists of 163 consecutive tests. The sequence is interrupted after a fail in one of the test items and are not executed for the remaining items of the test set. The main objective is to evaluate how the testing process could be compacted, identifying redundant tests, but also to find a method to prognosis if a test will fail based on results of previous tests in the sequence.

B. Data Pre-processing

Data from running production tests to electronic devices was recollected from large number of .csv 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.

These are key aspects included in the data pre-process scope to increase data quality and improve format for a better data handling:

- Because the sequence is interrupted for a certain device after one test is failed (when that is the case), some files do not have the same format, i.e. there are files containing less columns.
- Files are not tidy datasets. The first rows of every file contain general information that is not related to the test result.
- There are different test sequences, and this analysis is on a particular test (denoted as ‘p’) sequence only.
- Test results variables are of different types: categorical, logical, integer, decimal or hexadecimal.
- Some records contain a special character ‘@’, which is used as an indicator that the value of the test result is outside expected limits.
- Some records contain the character ‘*’, which indicates failure has occurred at that test.

On the other hand, for each test information with regards the overall test result, batch number, cell number, temperature, tester ID, operator ID, computer ID, date time, Match and Flex Type is available. This information is taken into account to evaluate if the test conditions added significant noise to the tests result.

These are the features covered in the R script written for cleansing pre-process:

- Removing rows that do not correspond to ‘p’ test.
- Removing rows where the test was aborted before a 'Fail' or 'Pass' result.
- For records containing the character '@' or ‘*’ the values were kept but removed the character '@' or ‘*’.
- Two complementary tables for those values with '@' or '*' were generated. For joining purposes, a key indicator was defined containing: batch number, cell number, date, and time of the record.
- The ‘p’ test sequence stops after a fail, but some consecutive tests consists of calculations from previous tests values. Those values were removed because were calculated with not truthful information.
- Inconsistency was found in the value format for some character variables, for example 'True' and 'TRUE'. Standardization is needed because these values could be taken as different by the R scripts.
- The hexadecimal records were converted to decimal. Having variables in the same numerical system basis will help to compare them but also for a better result when standardizing variables.

After following this cleaning process the dataset contains the results of testing 68168 devices, where 50882 assets passed and 17286 failed. Furthermore, there were found 10510 values with '@' and 16495 values with '*'. Is important to highlight that this dataset contains missing values, hence additional cleansing steps are needed before analysing data.

C. *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.

a) Chi-Square Test: Before appling Chi-Squre test this second cleansing process was ran:

Step 1) Eliminate from the analysis the tests items with variance 0: Test_9, Test_59, Test_62, Test_87, Test_151, Test_152, Test_156, Test_157, and Test_163.

Step 2) Complete cases approach was applied to each pair of test before running Chi-Square test.

The pairs of tests with p-value < 0.05 are listed in Table V. Those pairs are considered as significant associated.

We can conclude that surrounding conditions (Cell number, Tester ID, Computer ID, and Operator ID) affect overall test result. Furthermore, Test_162 result is associated to operator's interaction (Tester Id, Operator ID, and Match). We recommend to isolate the operator's interaction.

TABLE V. VARIABLES ASSOCIATED (CHI-SQUARE TEST)

Test A	Test B
OverallResult	CellNumber, TesterID, Comp_ID, Op_ID, and Test_162
BatchNumber	Op_ID, Match, and Flex_Type
CellNumber	TesterID, Comp_ID, and Match
TesterID	Op_ID, Match, Flex_Type, and Test_162
Comp_ID	Flex_Type
Op_ID	Test_162
Match	Flex_Type, and Test_162

b) Pearson's Correlation Coefficient: The Pearson's correlation coefficient was calculated for each pair of numerical continuous variables only, but firstly complete cases approach was applied to each pair of test items.

Table VI lists the 26 pairs of tests highly correlated (absolute value of Pearson's Correlation Coefficient > 90%), in particular some tests that are ran twice like Test_106. Based on the results of this analysis is recommended to drop 20 tests from the original test set:

- The second trial of Test_106, Test_122, Test_130, and Test_138.
- For the other 16 pairs of correlated tests we recommend to eliminate the more expensive and time consuming test of each pair.

TABLE VI. VARIABLES ASSOCIATED (PEARSON'S CORRELATION COEFFICIENT)

Test A	Test B
Test_1	Test_2, and Test_4
Test_2	Test_4
Test_3	Test_5
Test_12	Test_27, and Test_92
Test_13	Test_52
Test_14	Test_28, and Test_30
Test_17	Test_20
Test_27	Test_92
Test_28	Test_30
Test_70	Test_80
Test_84	Test_85
Test_88	Test_89
Test_93	Test_94
Test_98	Test_99
Test_106	Test_106B, Test_130, and Test_130B
Test_106B	Test_130, and Test_130B
Test_122	Test_122B
Test_130	Test_130B
Test_138	Test_138B
Test_149	Test_154

D. *Model Predictive Building and Evaluation*

In this analysis the model accuracy target is set to 90%, the ratio used to split dataset in Training/Test is 75:25. In addition, we omitted the tests with null variance that were identified in the previous Chi-Square test analysis.

After pre-processing the dataset and evaluating the association between variables or tests we are going to apply the proposed algorithm to compact the test set 'p'.

In the first iteration ($k = 2$) *Tree_2* using Test_1 and Test_2 only, was built. This tree consists of 3 nodes and its accuracy is 75.8%. During consecutive iterations, tests items were added one by one to the subset used to build DTs.

Finally, in iteration 102 Test_106 was added and built the *Tree_102* (Fig. 4) which consists of a root node and two leafs. This BDT classifies as Pass whenever Test_106 < -62. The model accuracy is 90.6%. On the other hand, when using this BDT to prognosis overall test fail could be omitted Test_107 to Test_163 of the sequence which correspond to 35% of the original sequence.

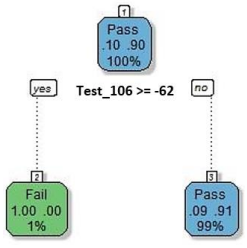

Fig. 4. Decision Tree – Iteration 102 (*Tree_102)*

Model misclassifies actual pass devices only (Fig. 5), which means that warranties levels will not be increased, only production costs. Nevertheless, if the test sequence is completed for the devices predicted as Fail the cost of misclassification could be reduced.

		Predicted Class	
		Pass	Fail
Actual Class	Pass	0.899	0.090
	Fail	0	0.011

Fig. 5. Confusion Matrix of *Tree_102*

The *Tree_102* is not over fitted, all cp values are at least 0.01 (Fig. 6). In addition, the architecture of the tree does not change in the consecutive 7 iterations.

```
          CP nsplit rel error    xerror       xstd
1 0.0919246      0 1.0000000 1.0000000 0.01446264
2 0.0100000      1 0.9080754 0.9080754 0.01385302
```

Fig. 6. Complexity Parameter Calculation Tree_102

E. Production Test Set Compactation

Combining results of Variables Association Analysis and Iterative Within-set BDT, it is considered possible to omit 75 tests of the tests sequence, which corresponds to 46% of the original test set (Table VII). The accuracy of classification is above 90%, warranties levels and quality costs are not impacted.

TABLE VII. TEST SET COMPACTION

Approach	Test Omitted
Iterative within-set DT (all tests after Test_106)	52
Test with variance = 0	9
Test associated (before Test_106)	14

F. Trade-off Analysis between Prediction Accuracy and Test Set Compactation

The proposed method provides flexibility to evaluate between model accuracy and test set compaction (Fig. 7) when the algorithm is executed until the last test is included. For example, Tree_55 model has an accuracy of 86.2% but in the next iteration the accuracy grows 1.5%. On the other hand, from iteration 72 to 101 the growth is 0.87% only.

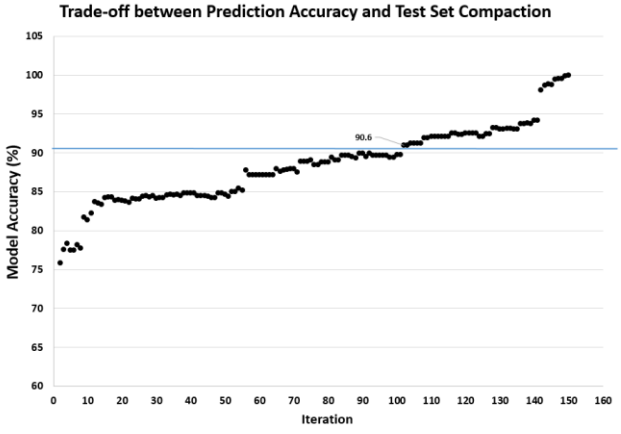

Fig. 7. Trade-off between Prediction Accuracy and Test Set Compaction

V. CONCLUSIONS

In this paper we presented a novel algorithm to compact test set on a stop-on-fail test scenario with parametric and non-parametric tests. This development was enabled through performing a successful data mining analysis on the production test data which covered data gathering in raw format, data pre-processing, and integration of the proposed novel algorithm and variables association analysis. The outline algorithm, based on use of decision tree, is found to be adequate for applications with incomplete datasets and can be employed in real production lines to offer flexible trade-off between the model accuracy and test set compaction. Advantages associated with ease of implementation and computational efficiency were also illustrated. Embedding the presented data driven predictive model 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 46% test reduction with prediction accuracy above 90% for faulty components.

ACKNOWLEDGMENT

The authors would like to acknowledge the support from Microsemi Corporation (MSCC) for providing the dataset and test process information.

REFERENCES

[1] Chen M., and Orailoglu A., "Test Cost Minimization through Adaptive Test Development," 2008 IEEE International Conference on Computer Design, Lake Tahoe, CA, 2008, pp. 234-239.

[2] C. K. Hsu et al., "Test data analytics — Exploring spatial and test-item correlations in production test data," 2013 IEEE International Test Conference (ITC), Anaheim, CA, 2013, pp. 1-10.

[3] F. Lin, C. K. Hsu and K. T. Cheng, "Learning from Production Test Data: Correlation Exploration and Feature Engineering," 2014 IEEE 23rd Asian Test Symposium, Hangzhou, 2014, pp. 236-241.

[4] A. Nahar, R. Daasch and S. Subramaniam, "Burn-in reduction using principal component analysis," IEEE International Conference on Test, 2005., Austin, TX, 2005, pp. 155-165.

[5] N. Sumikawa, D. G. Drmanac, L. C. Wang, L. Winemberg and M. S. Abadir, "Forward prediction based on wafer sort data — A case study," 2011 IEEE International Test Conference, Anaheim, CA, 2011, pp. 1-10.

[6] H. V. Pham, S. N. Demidenko and G. M. Merola, "Eliminating Re-Burn-In in semiconductor manufacturing through statistical analysis of production test data," 2017 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), Turin, 2017, pp. 1-6.

[7] S. Meeyai. "Logistic Regression with Missing Data: A Comparisson of Handling Methods, and Effects of Percent Missing Values," Journal of Traffic and Logistics Engineering vol. 4, no. 2, December 2016.

[8] H. G. Stratigopoulos, P. Drineas, M. Slamani and Y. Makris, "RF Specification Test Compaction Using Learning Machines," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 18, no. 6, pp. 998-1002, June 2010.

[9] R. Záluský, D. Ďuračková, V. Stopjaková, J. Brenkuš, J. Mihálov and L. Majer, "Production test-based classification of antennas using the feed-forward neural network," 2014 24th International Conference Radioelektronika, Bratislava, 2014, pp. 1-4.

[10] S. Biswas and R. D. Blanton, "Statistical Test Compaction Using Binary Decision Trees," in IEEE Design & Test of Computers, vol. 23, no. 6, pp. 452-462, June 2006.

[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, pp. 1-34.

[12] Therneau, T., Atkinson, B., and Ripley, B., "An Introduction to Recursive Partitioning Using the RPART Routines," Mayo Clinic. February 2018.

[13] Therneau, T., Atkinson, B., and Ripley, B., 'rpart,' R package version 4.1-13, February 2018.

[14] Williams, G., et al, 'rattle,' R package version 5.1.0, September 2017