POLITECNICO DI TORINO
Repository ISTITUZIONALE

Feature Selection for Cost Reduction in MCU Performance Screening

Original
Feature Selection for Cost Reduction in MCU Performance Screening / Bellarmino, N., Cantoro, R., Huch, M., Kilian, T., Schlichtmann, U., Squillero, G.. - (2023), pp. 1-6. (24th IEEE Latin-American Test Symposium (LATS) Veracruz (MEX) 21-24 Marzo 2023) [10.1109/LATS58125.2023.10154495].

Availability:
This version is available at: 11583/2976251 since: 2023-02-21T16:28:29Z

Publisher:
IEEE

Published
DOI:10.1109/LATS58125.2023.10154495

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2023 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

09 November 2026

Feature Selection for Cost Reduction in MCU Performance Screening

Nicolò Bellarmino*, Riccardo Cantoro*, Martin Huch†, Tobias Kilian†‡,
Ulf Schlichtmann‡ and Giovanni Squillero*

*Politecnico di Torino
Torino, Italy

†Infineon Technologies AG
Munich, Germany

‡Technical University of Munich
Munich, Germany

***Abstract*—In safety-critical applications, microcontrollers must satisfy strict quality constraints and performances in terms of $F_{\max}$, that is, the maximum operating frequency. It has been demonstrated that data extracted from on-chip ring oscillators, the so-called speed monitors, can model the $F_{\max}$ of integrated circuits using machine learning models. Those models are suitable for the performance screening process, and they use speed monitors are features, while the target is the $F_{\max}$. But if the number of features used for building a machine learning model is huge, the risk of over-fitting or curse of dimensionality is high, leading to a high generalization error. Also, devices with a high number of ring-oscillator are costly to be produced. This paper copes with supervised feature selection in microcontroller performance screening during the early phase of prototyping and presents methodologies to reduce the number of monitors needed to build efficient machine learning models without losing in accuracy. We propose a methodology to rank features according to their importance in the performance prediction, able to extract a subset of them drastically reduced in size, but still able to well solve the underlying task. Experiments showed that the chosen subset of features leads to simpler ML models that can achieve lower prediction error, reducing overfitting. This permits avoiding inserting the full set of sensors in the final product, with a huge saving of money and physical space in the silicon.**

***Index Terms*—Fmax, Speed Monitors, Ring Oscillators, Speed Binning, Machine Learning, Device Testing, Manufacturing**

I. Introduction

The purpose of Microcontroller (MCU) performance screening is to detect underperforming devices that do not fully meet the characteristics described in the datasheet in terms of maximum operating frequency $F_{\max}$. Literature has proven that machine learning (ML) models trained on data that can be correlated with the device's $F_{\max}$ can be used to predict the operating frequency of the device with good results [1]–[3]. The accuracy of supervised ML models depends on the quality and the quantity of labeled data. Previous works have proposed the usage of Speed Monitors (SMONs) to predict $F_{\max}$ values derived from the execution of functional patterns on single devices [2], [4]–[6]. In this context, unlabeled data are relatively inexpensive to acquire, and the number of sensors deployed may be huge (hundreds or thousands), thus leading to an extremely high number of features for the ML models. This may lead to over-fitting problems or the curse of dimensionality: when the dimensionality increases, also the volume of the space increases, and data become sparse. In this context, in order to obtain reliable ML models, it would be necessary to have an amount of data that grows exponentially with the dimensionality of the space. But since the process of acquiring the labels is costly in terms of time needed, also the amount of labeled data available is limited (often only hundreds of samples). In this paper, we focus on feature selection on an MCU test chip, with the goal of reducing the features needed to build machine learning models and thus decreasing the number of sensors to be placed on board. We rank the feature on the basis of their importance in the supervised performance prediction task by means of repeated features-selection steps. These techniques allowed us to drastically reduce the feature space dimension without a noticeable loss in prediction performances and thus save area on the silicon. The methodology also gives test engineers useful information on the correlation between sensors and $F_{\max}$, for diagnosing purposes. Experiments showed that this approach can lead to an acceptable prediction error even with a fraction of the original features (only 15% of the original features).

The rest of the paper is organized as follows. Section II presents related works on the topic. Section III describes theory and concept useful for understanding successive experiments; in particular, Section III-A describes the characterization process used to derive the dataset for ML algorithms; Section III-B describes the SMONs used as features, while Section III-C introduces the concepts of ML and Feature Selection. In Section IV, the motivations why we need feature selection are given. In Section V, details on the proposed approach are given. Section VI presents the experimental evaluation. Finally, Section VII draws the conclusions.

II. Related Work

Several approaches to performance prediction have been proposed in the past [7]–[10]. The use of ML models to relate structural and functional $F_{\max}$ was first introduced in [4].

Using indirect measures to predict circuit parameters is called in literature 'alternate test', and has been widely studied for analog circuits [11]–[14]. The core idea is to learn a mapping between indirect measurements and circuit parameters and to use only the indirect low-cost measurements to predict device parameters during production testing.

Authors are listed in alphabetical order.

The authors of [1]–[3] worked on building an ML model for $F_{\max}$ prediction, to be used in MCU performance screening. In [2], they correlated the values of 27 speed monitors to functional $F_{\max}$ measured on more than 4,000 packaged devices extracted from 26 corner-lot wafers. In [3], Active Learning metrics were used in order to consider the effects of process variations during training. In [1], they evaluated the effectiveness of several outlier detection techniques in identifying anomalous, noisy data, and outliers. Feature reduction is a very well topic in ML community [15]. There exist two main techniques for dimensionality reduction: *feature selections* selects a subset of features from the original ones on the basis of some criteria (for example, the effectiveness in predicting the target label). In *feature extractions*, we build a new and smaller set of features as a combination of the original ones, compacting the information on the dataset. Principal Component Analysis (PCA), a feature extraction technique invented in the early 1900s by Karl Pearson [16], has been effectively used in related works on SMONs [17].

III. Background

A. Microcontroller Characterization Process

In the context of alternate tests, an ML model is trained on data that can be correlated to certain circuit parameters or characteristics. In our case, the output characteristic is the device's performance $F_{\max}$ (labels), and the alternative test input parameters are the SMON frequencies (features). The features used are the SMONs, or on-chip ring oscillators (ROs), and their frequencies are measured during production with high accuracy in a stable, fast, and easy process. Measuring SMONs' frequencies is part of the regular production test, and thus the features are potentially available for every produced MCU. However, training the ML models requires the acquisition of an adequately labeled dataset and, thus, the MCU characterization. The labeling process is a time-consuming procedure performed mostly manually, in which each MCU is measured individually with functional test patterns [2]. Therefore, due to the high effort, the process is performed on a small subset of the manufactured devices. The labeling involves precise steps: each MCU is mechanically mounted on a board that mimics the in-field application. Then, the MCU starts executing a certain functional pattern (e.g., a test program) with low frequency, which is slowly stepped up until a functional failure happens [18] (e.g., a crash in the application, an erroneous response, etc.); the last working frequency $F_{\max}$ is then stored. The procedure is typically repeated using various functional test patterns, thus leading to a multi-label dataset. For each MCU, the $F_{\max}$ values collected with the various patterns can have a particular spread, due to devices' process variation which usually follows a Gaussian distribution [19], or bias in the label measurements. The bias could be either a defective device or a measurement affected by uncertainty due, for example, to minor changes in voltage or temperature condition, mechanical vibration of the board, or white noise. In order to guarantee robustness in the measurement process and to avoid including outliers in the training procedure, those devices that present at least one functional pattern in which the $F_{\max}$ deviates for more than 2.5 standard deviations from the wafer median are eliminated. The desired outcome is a high-quality set of labeled devices that are used for ML training. At the end of the labeling process, each labeled device has multiple labels for different functional patterns.

B. The SMONs

As already stated - the SMONs are on-chip ROs and are used as features to predict the performance of the MCU. Thus, the SMONs structure and sensitivity also significantly impact the performance model. Therefore, the aim is to place a large variety of different types of SMONs in the MCU. In addition, the SMONs have to be spatially distributed to cover the Within-die (WID) process variation that is increasingly emerging in shrinking technologies [20].

In order to handle such requirements, a SMON module is developed such as it contains a heterogeneous set of different SMONs. Different SMON modules are placed in multiple locations on the MCU to ensure spatial coverage.

A SMON module consists of different generic and design-dependent ROs. The generic ROs consists of inverter gates, NAND gates, and NOR gates from different cell libraries, respectively. The design-dependent ROs are replicated functional paths from the design. Several equal SMONs modules were placed in different locations of the MCU under test.

In addition to the SMON modules, the MCU contains several functional paths RO introduces in [21]. Such functional path ROs (fpath) uses functional paths in the design and use them as ROs and do not add dedicated ROs to silicon.

The used feature set of SMONs from multiple SMON modules and functional path ROs gives us a solid foundation for the chip behavior in every corner and enormous benefit for data processing. In the end, thousands of sensors are placed on board, divided into several identical SMON modules.

C. Machine Learning

Training an ML model means finding an input-output relation on the basis of the available data. The model analyzes some *features* and produces an output on the basis of these values. Simple linear regression algorithms, for example, perform a linear combination of the input features on the basis of some weights, assigned to each feature during the training process. But if the number of observation samples is lower than the number of features, there exists the risk of massively overfitting our model, which would be not able to generalize on new data, unseen at training time. The number of samples needed to estimate an arbitrary function with a given level of accuracy grows exponentially with respect to the number of input variables (i.e., the dimensionality of our data) [22]. This situation is called *the curse of dimensionality* (COD). In simple words, the more features we have and the more data we need to build robust ML models. Dimensionality reduction techniques can be used to deal with COD. Recursive Feature Elimination (RFE) [23] is a popular feature selection algorithm. RFE is

effective at selecting the features the more related to predicting the target variable on the basis of an underlying ML model. The hyperparameters to be chosen are the number of features to select and the ML model to solve the underlying supervised problem. Given an external estimator that assigns weights to features (for example the coefficients of a linear model), RFE selects features by recursively considering smaller and smaller sets of these. First, the estimator is trained on the whole set of features and the importance of each feature is obtained. Then, the least important features are pruned. That procedure is recursively repeated on the pruned set until the desired number of features to select is eventually reached. RFE with Cross Validation (CV) performs RFE in a cross-validation loop to find the optimal number of features.

IV. Motivations

As indicated in Section III-B, placing more SMONs on board (and in different locations) could help to better deal with the WID process variation, and thus give us more information about the performances of the MCU. So, to have excellent performance predictions, it might make sense to have as many SMONs as possible. But this cannot be feasible for two main reasons:

1) Each SMON that is added to the design occupies a physical area on the chip. Adding hundreds of SMONs causes a non-negligible area overhead and contributes to current leakages. Moreover, the SMONs are for testing purposes only and have no functional value for the customers. Therefore, the occupied area should be kept as small as possible.
2) From an ML perspective, the more SMONs we have, the more features our ML models will use. Nevertheless, when the feature space dimensions increase, the risk of the *curse of dimensionality* becomes high, and the ML models can become significantly inaccurate on new samples because of overfitting. This circumstance is also amplified by the limited amount of labeled data available.

The motivations for reducing the features are thus clear: from a technological point of view, a reduced number of features allows the production costs of each device to be reduced, as only the most informative sensors are incorporated into future products. A feature ranking is obtained as an additional outcome of the feature reduction proposed technique. Such feature ranking is also helpful for diagnostic purposes, as it provides test engineers with relevant information about the correlation between individual SMONs and patterns. It also indicates the SMONs modules with which we can achieve optimal performance prediction, and thus where to place the SMONs on board. Since the SMONs in the SMONs modules are the same at each location, the different contribution of the SMONs modules is only related to the spatial positioning on the chip.

Also, a reduced feature set permits an ideal increase in the generalization performances of the ML models and permits achieving simpler models that work on fewer inputs.

V. Proposed Approach

The approach followed in this work aims to extract the most relevant feature set for each pattern by means of a classical feature extraction method (RFE). Applying directly the RFE or RFECV is not possible, because these methods are based on estimating the importance of each feature by means of the coefficients of an ML model. If we use some feature transformations (for example, the polynomial features), the coefficients of the ML model are no more directly linked to each original input feature, but to the artificial features created after the transformation. Thus, we need to develop a method that permits us to rank the original features, selecting the most informative ones. The developed approach can be resumed in 5 steps, explained in detail in this section:

1) Identify a good baseline with which to compare the successive models.
2) Identify the best subset of location in which place the SMONs by brute force approach.
3) Identify a reasonable number of features.
4) Obtain a ranking for the SMONs, both for ML and diagnosis purposes.
5) Train the models with the best SMONs of the best location(s).

We first trained our machine learning models on all the features available, measuring the baseline performances. To reduce the dimensionality, it is possible to use PCA as a preprocessing step. The PCA extracts a reduced features-set by computing the principal components, performing a change of basis on the data. The new components are linear combinations of the ones in the original space that maximize variance and that are uncorrelated with each other. Results by including PCA are satisfactory. This means that it is possible to successfully compact information in our dataset since many independent variables are highly correlated. We can collect 99% of the variance of a single SMON module with only 10% of the features and 99.99% with 15% of the features, meaning we have a high redundancy in the input space. If we consider all the SMONs modules (in general, n), these numbers drop, respectively, to $\frac{10\%}{n}$ and $\frac{15\%}{n}$ (and this is absolutely reasonable, since the different SMONs modules have the same SMONs, and the only difference is the spatial positioning on the chip). However, the PCA approach would reduce only the input of the ML models, but not the monitors mounted on board. We still need all the monitors to compute the principal components. So PCA is used to obtain a baseline to compare the results of subsequent experiments. PCA will be dropped after finding the best features set, and will not be inserted in the final models. Thus, we need a feature selection algorithm rather than feature extraction. We are interested in selecting the best feature from the original feature set, with negligible (or no) loss in prediction error with respect to the feature extraction method. Since monitors are placed in more than one location, we first need to find the best possible combination of the locations or check if only one of them is enough to reach satisfactory performances. We used a combinatorial brute force

approach to compute all the possible location subsets, and for each of them, we computed the performance by using the chosen model. We then ranked the sets on the basis of the prediction error obtained for each pattern. We sorted, for each pattern, the prediction error obtained with a certain set (Fig. 1), and we gave each set a score inversely proportional to the position on the rank (the first set, the one with the lowest prediction error, have a score equal to n, with n the number of sets. The second one have a score of $n-1$ and the latest have a score of 0). This rank permits test engineers to have a clear idea of the importance of each location for each pattern, and if it is the case to place monitors in more than one location. Summing up the importance of each location in all the patterns, we obtain the final ranking (Fig. 2), which informs us about the mean performance of locations set on the different patterns/tasks. However, due to technological constraints, we decide to keep only one location and thus the best set is for us the one with a single location that appears first in the rank. To find a good number of features to select for every location, we used the RFECV, which performs RFE in a cross-validation loop to find the optimal number of features. We used three different simple linear algorithms with regularization as base estimators (Ridge, Lasso, ElasticNet), with no feature transformation. These algorithms associate a weight to each feature, and thus it is possible to extract the importance of the features by looking at these weights (Fig. 3). The lack of feature transformation leads to higher prediction error, but this phase is only intended to select the SMONs the most related to the performance prediction task. The selected SMONs will be used as features for the final models, with non-linear feature transformation. Once the optimal or a reasonable number of features is found, we can proceed to rank the SMONs: to do this we run several steps of RFE with the selected number of features, with different training sets (in a CV fashion), with many different base estimators and for all the patterns and different locations. The repeated run permits the validation of the results, avoiding that these have happened by chance. At the end we obtain a ranking of all the SMONs, counting how many times each SMON was selected by the algorithm (Fig. 4). The rank permits to test engineer to have a clear idea of the contribution of each SMON in the performance prediction task. It is also possible to check the importance of each SMON for each different pattern, finding that there are some SMONs that contribute more to one pattern rather than another (Fig. 4). We compared the results obtained with the reduced features set on a single location, finding that we were able to have effectively increased the prediction performance of our model (Table I). We were able to reduce the number of features-set size to only $\frac{15\%}{n}$ of the original one (by considering all the SMONs in all the n SMONs modules). In the case in which we selected more than one location, or if we have an already developed product in which SMONs were placed in more locations, it is possible to compact information by using aggregating functions for the SMONs values: for each SMON in the different modules, we extract a single value (for example, averaging them). This strategy, called

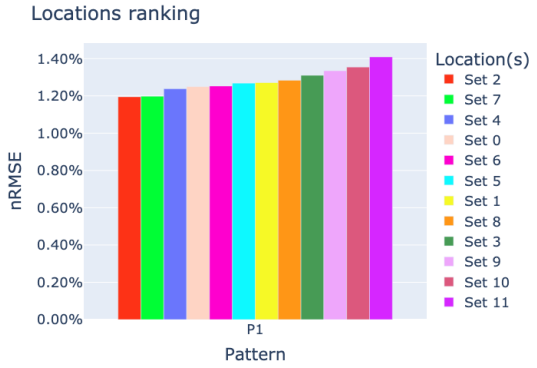

Fig. 1. Location ranking for a pattern (example showing 12 different location sets).

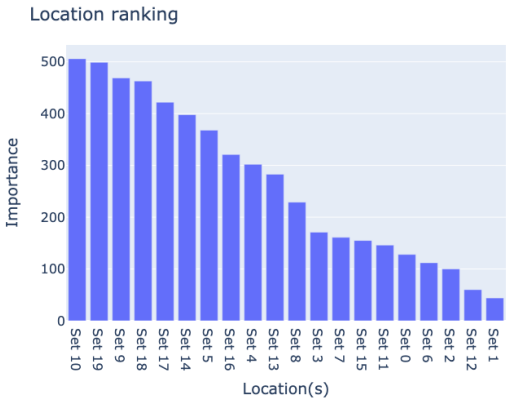

Fig. 2. Location Ranking (toy example). On the x-axis is the set of locations used for training the model and on the y-axis is the correspondence importance (20 sets are shown).

"Virtual Module (VM)", permits both the reduction of the number of SMONs that the ML analyzes and the exploitation of the correlation and SMONs variations among the different locations. This approach permits keeping the ML algorithm as simple as possible even in the case of multiple SMONs modules, and with this technique we are able to reach the best performance. In general, the developed techniques permits both to 1) find the best combination of features/modules, that lead to the lowest prediction error, if we are interested in reducing the error as much as possible and 2) find a good subset of features, that lead to a little drop in performances but with a great save of money/space in the chip, since we will place fewer sensors on the final product.

VI. Experimental Evaluation

A. Experimental Setup

The proposed methodology has been validated on a dataset composed of few thousand samples with about a thousand of features (SMONs), divided into several SMONs modules plus tens of independent features (fpath). The evaluation of the model was performed with a 5-folds CV stratified per wafer.

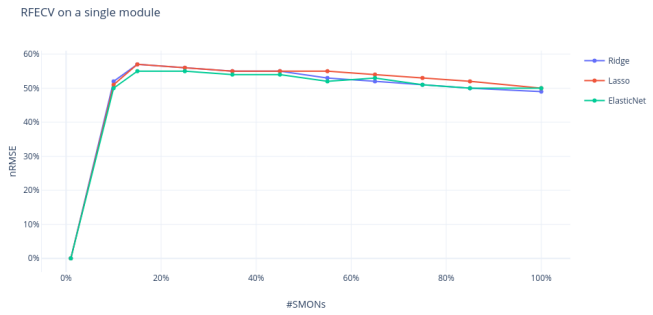

Fig. 3. RFECV algorithm run on a SMONs location with fpath. On the x-axes the number of features, on the y-axes the R2 score. Going over 15% of SMONs, linear methods do not have any great advantages in terms of reducing the prediction error.

TABLE I
POLY RIDGE PREDICTION ERROR WITH DIFFERENT FEATURES SETS

Method	Used SMONs (%)	Input Features (%)	nRMSE
n modules (no PCA)	100%	100%	2.02%
n modules (PCA)	100%	$15\%/n$	1.42%
1 module (no PCA)	100%	100%	1.92%
1 module (PCA)	100%	15%	1.40%
1 module (best SMONs)	15%	15%	1.37%
VM (median)	100%	$15\%/n$	1.28%
VM (mean)	**100%**	$\mathbf{15\%/n}$	**1.25%**

All experiments were performed in Python. Experiments run on a server equipped with an AMD ®EPYC 7301 16-Core CPU @ 3.20GHz × 16, 128GB of RAM.

The model chosen is a Ridge Regressor (Linear Regression with L2 regularization). Each column of the dataset was scaled by subtracting the mean and dividing it by the variance (*Standard Scaler*). Literature suggests the use of polynomial features, to project the data into a higher-dimensional space. This transformation enables non-linear interactions among features, and thus permits the use of simple linear regressions to solve the performance prediction task [1], [3]. In some of the experiments, PCA was inserted before the polynomial transformation as a feature reduction step (highlighted in the tables). We will call the pipeline composed by the Standard Scaler, the eventual PCA, the polynomial transformation, and the Ridge Regressor the Polynomial (or Poly) Ridge.

Results are presented in terms of normalized Root Mean Square Error (nRMSE) and R2 score. RMSE is a popular regression performance index [24] but in this context we normalized it by the mean value of $F_{\max}$ in the test set, i.e. $nRMSE = RMSE(y_{true}, y_{pred})/mean(y_{true})$ to obtain a percentage of the error with respect to the mean frequency of the samples. The coefficient of determination R2 is the proportion of the variation in the dependent variable that is predictable from the independent variable(s) [25]. A regressor that perfectly fits the data would have an R2 score of 1.

B. Results

Baseline performances were computed by mean of 5-folds CV by using the Polynomial Ridge described in the previous section. The best subsets of location were computed as described in section Section V, by extracting all the possible combinations of location. The best set is the one that appears first in the rank. However, we are interested in reducing as much as possible the number of locations, and thus we can take as the best location the first one that appears as a single location in the rank. With the best single location, we can obtain a mean error of 1.92%, with a negligible loss in prediction performance with respect to taking more than one SMONs module. To find a good number of features to select, we run the RFECV algorithm (5-fold CV and 3 linear base-regressors). Experiments showed that approximately 15% of the features of a single SMON module is a good number of features to extract since we have no evident advantages by using more features (Fig. 3). The reached R2 score with simple linear algorithms becomes quite steady ($\approx$ 0.8) with that number of features, and we have no advantages by increasing them. This result is supported both by RFECV algorithm and by PCA (Section V).

To select the best feature set in all the locations, repeated runs of RFE were performed by considering different settings: 5 different regressors were used in a 6-folds CV, for every pattern and for each of the locations, resulting in about 2 thousand runs. The regressors used were 3 linear regressors with L1-L2 regularization and 5-folds CV to hyper-parameter tuning (i.e. Lasso, Ridge, ElasticNet) and 2 tree-based regressors (i.e. Random Forest, Decision Tree). For each run, 15% of the features were select, and we increased a counter for each SMON every time this was selected. We build the rank on the basis of this score. We can compute the score of the SMONs grouping by patterns, obtaining the importance of SMONs for each pattern Fig. 4 or we can group by location, or, to compute a general score, we can sum up all the results. This gives us a general view of the importance of each SMON, and we can select the best subsets of SMONs that should work well on average for each pattern Section VI-B. The first 15% of the SMONs were selected, and the results were validated on a proper test set. The "Virtual Module" was created by aggregating the SMONs value by means of different functions (mean and median values of the same SMONs in the different locations). Results showed that this solution is the most effective one in terms of prediction error Table I. Also, using some aggregation function permits making the SMONs measurements robust to possible noise, missing value, or faults (while this would not be possible with a single SMON measurement). With this approach. we were able to reach an R2 score $R2 \approx 0.93$

VII. CONCLUSIONS

We presented a feature reduction framework to be used in MCU performance screening. SMONs values are good alternate measures for performance prediction, and industries/test engineers may use a high number of SMONs to

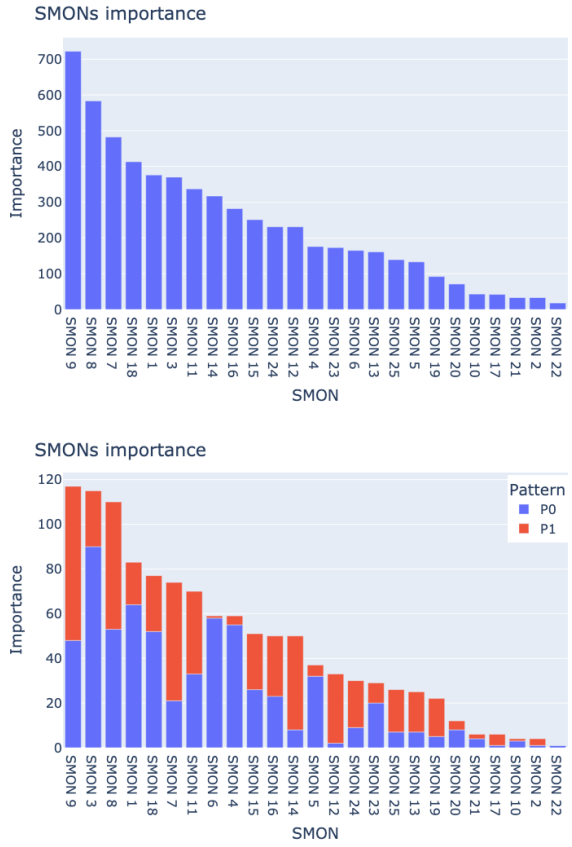

Fig. 4. Toy example of the SMONs ranking one should obtain. On the y-axis, the number of times the SMON was chosen by the repeated RFE run. On the top, is the final ranking by summing up all the results. On the bottom, the ranking considering the same SMONs on two patterns. The value of some SMONs may be more important for a patterns rather than others (see SMON 4, 12).

catch performance variation. However, if this number is huge, the ML models may be prone to overfitting, and a high number of labeled devices would be necessary to train the models and achieve good generalization performances. Our framework permits the selection of the best features, the ones that contribute the most to the performance prediction task. The experiments showed that this approach reveals the optimal number of features for each location, and the best location(s) in which place the SMONs in the early stage of product development, to reduce as much as possible the cost of the final product. It can be applied also in already developed products, by aggregating information using the mean or the median value of each SMONs in a different location, creating a "Virtual Module" that compacts information and make the SMONs measurements more robust to possible noise or faults. Our experiments showed that the use of a reduced features set permits the achievement of lower prediction error (from 2.02% to 1.25% of nRMSE), and thus, higher generalization capabilities. This approach is in general employable in the context of MCU (of any type) performance prediction by measuring the $F_{\max}$ using functional testing benchmarks, and by means of alternative measurements for performance (SMONs, in our case).

REFERENCES

[1] N. Bellarmino *et al.*, "Microcontroller Performance Screening: Optimizing the Characterization in the Presence of Anomalous and Noisy Data," in *IEEE International Symposium on On-Line Testing and Robust System (IOLTS)*, 2022.

[2] R. Cantoro *et al.*, "Machine Learning based Performance Prediction of Microcontrollers using Speed Monitors," in *2020 ITC*, 2020.

[3] N. Bellarmino *et al.*, "Exploiting active learning for microcontroller performance prediction," in *2021 IEEE European Test Symposium (ETS)*, 2021.

[4] J. Chen *et al.*, "Data learning techniques and methodology for Fmax prediction," in *2009 ITC*, 2009.

[5] J. Chen *et al.*, "Selecting the most relevant structural Fmax for system Fmax correlation," in *2010 VTS*, 2010.

[6] M. Sadi *et al.*, "SoC Speed Binning Using Machine Learning and On-Chip Slack Sensors," *IEEE TCAD*, 2017.

[7] K. von Arnim *et al.*, "An effective switching current methodology to predict the performance of complex digital circuits," in *2007 IEEE International Electron Devices Meeting*, 2007.

[8] D. Blaauw *et al.*, "Razor II: In situ error detection and correction for PVT and SER tolerance," in *2008 IEEE International Solid-State Circuits Conference - Digest of Technical Papers*, 2008.

[9] G. Sannena *et al.*, "Low overhead warning flip-flop based on charge sharing for timing slack monitoring," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2018.

[10] T. B. Chan *et al.*, "DDRO: A novel performance monitoring methodology based on design-dependent ring oscillators," May 2012.

[11] H. Ayari *et al.*, "Making predictive analog/rf alternate test strategy independent of training set size," in *2012 IEEE International Test Conference*, 2012.

[12] P. Variyam *et al.*, "Prediction of analog performance parameters using fast transient testing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2002.

[13] H.-G. Stratigopoulos *et al.*, "Error moderation in low-cost machine-learning-based analog/rf testing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2008.

[14] J. Brockman *et al.*, "Predictive subset testing: Optimizing ic parametric performance testing for quality, cost, and yield," *IEEE Transactions on Semiconductor Manufacturing*, 1989.

[15] W. Jia *et al.*, *Feature dimensionality reduction: A review*, en, Jan. 2022.

[16] I. T. Jolliffe *et al.*, "Principal component analysis: A review and recent developments," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, Apr. 2016.

[17] C. Chang *et al.*, "Accurate performance evaluation of vlsi designs with selected cmos process parameters," *IET Circuits, Devices Systems*, 2018.

[18] R. McLaughlin *et al.*, "Automated Debug of Speed Path Failures Using Functional Tests," in *2009 27th IEEE VLSI Test Symposium*, 2009.

[19] K. Maragos *et al.*, "In-the-Field Mitigation of Process Variability for Improved FPGA Performance," *IEEE Transactions on Computers*, 2019.

[20] S. Asai, Ed., *VLSI Design and Test for Systems Dependability*. Springer Japan, 2019.

[21] T. Kilian *et al.*, "A scalable design flow for performance monitors using functional path ring oscillators," in *2021 IEEE International Test Conference (ITC)*, 2021.

[22] R. E. Bellman, *A Guided Tour*. Princeton: Princeton University Press, 1961.

[23] A. A. Megantara *et al.*, "Feature importance ranking for increasing performance of intrusion detection system," in *2020 3rd International Conference on Computer and Informatics Engineering (IC2IE)*, 2020.

[24] T. Chai *et al.*, "Root mean square error (rmse) or mean absolute error (mae)?– arguments against avoiding rmse in the literature," *Geoscientific Model Development*, Jun. 2014.

[25] D. Chicco *et al.*, *The coefficient of determination r-squared is more informative than smape, mae, mape, mse and rmse in regression analysis evaluation*, en, Jul. 2021.