Embedded Feature Selection in MCU Performance Screening

# Embedded Feature Selection 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 in terms of maximum operating frequency ($F_{\max}$). Data from on-chip ring oscillators, the so-called Speed Monitors, can be used as features of Machine Learning models to predict $F_{\max}$. Increasing the number of ring oscillators on the chip can increase the information retrieved about the device's speed. However this may also lead to overfitting, and a lack of generalization capabilities.**

**This paper focuses on supervised feature selection in performance screening during the early phase of prototyping. The aim is to reduce the number of features while maintaining the accuracy of machine learning models. Two distinct approaches for obtaining feature rankings based on ring oscillators' significance in predicting performance are compared: one based on Recursive Feature Elimination, and one on regularized linear models. 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, saving money and physical space in the silicon.**

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

## I. Introduction

Microcontroller (MCU) performance screening aims to identify devices that do not meet the specified characteristics, such as the maximum operating frequency ($F_{\max}$) indicated in the datasheet. Existing literature has demonstrated the effectiveness of machine learning (ML) models trained on data correlated with $F_{\max}$ in accurately predicting the operating frequency [1]–[4]. Previous research has advocated for the use of on-chip ring oscillators (ROs), also denoted as Speed MONitors (SMONs) to predict $F_{\max}$ values [4]–[7]. In principle, having numerous predictor sensors may increase the information about the device's speed. However practically speaking, having hundreds of features for ML models potentially causes overfitting because of the *Curse of Dimensionality* (COD): in higher dimensional space, data tends to be sparse, and more and more labeled data are needed to estimate a reliable relation between features and target. But since obtaining accurate $F_{\max}$ value is a time-consuming process, we often have only a limited availability of labeled data, in the scale of hundreds of samples.

This paper addresses this challenge by focusing on feature selection, aiming to reduce the number of features (and thus, the number of SMONs) required for building ML models. The approach involves ranking features based on their importance in the supervised performance prediction task. We compared two different methods, both based on repeated feature-selection: the first, from previous work [8], relies on state-of-the-art Recursive Feature Elimination (RFE). The proposed approach, instead, is based on Embedded Feature Selection and regularized linear models.

Experimental results showcase the viability of this approach, demonstrating to be effective in significantly reducing the dimensionality of the feature space without compromising prediction performance. Acceptable prediction error can be reached even with a fraction of the original SMONs (17%). Additionally, since a SMONs ranking is a natural outcome of the proposed procedure, they provide valuable insights for test engineers regarding the correlation between SMONs and $F_{\max}$. Reducing the SMONs contributes both to efficient prediction models and cost savings and optimized use of silicon space in the microcontroller.

The rest of the paper is organized as follows. Section II presents related work 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 and Section III-D describes the regularized linear models setting. 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 [9]–[11]. In the literature, using indirect measures to predict circuit parameters is called 'alternate test' and has been widely studied for analog circuits [12]–[15]. The core idea is to learn a mapping between indirect measurements and some circuit parameters, and to use only the indirect

Authors are listed in alphabetical order.

979-8-3503-6312-8/24/$31.00 ©2024 IEEE

low-cost measurements to predict circuits parameters during production testing.

The authors of [1]–[3] worked on building ML models for $F_{\max}$ prediction, to be used in MCU performance screening. In [2], they correlated the frequency values of 27 on-chip SMONs to functional $F_{\max}$. Dimensionality reduction is well-covered topic both in the ML [16], [17] and CAD communities, with several works in the realm of alternate tests for analog circuits [18], [19]. The importance of feature selection in MCU performance screening was firstly addressed in [8]. However, existing methods often rely on filtering or wrapper approaches. Filtering approaches are usually univariate (considering only one feature at once). Wrapper methods like Sequential with Forward and Backward feature selection or RFE, instead, usually have a high computational cost and dependency on the model and data, with the risk of overfitting [20].

Two main techniques exist for dimensionality reduction: *features selection* (FS) and *features extraction* (FE). FS selects subsets of features based on some criteria (like the effectiveness in predicting the target label). FE builds 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 [21], has been effectively used in related works on SMONs [4].

## III. Background

### A. Microcontroller Characterization Process

Frequencies from on-chip ROs, referred to also as SMONs, provide features for the ML model. These frequencies are accurately measured during production using a stable, fast, and straightforward process. As measuring SMON frequencies is part of the regular production test, these features are potentially available for every produced Microcontroller Unit (MCU). However, training ML models necessitates a properly labeled dataset, requiring MCU characterization.

The labeling process is time-consuming, involving measuring each MCU individually with functional test patterns [2]. This process is conducted on a small subset of manufactured devices. The labeling procedure includes mounting each MCU on a board and executing a specific functional pattern with a low frequency, incrementally increasing until a functional failure occurs, with the last working frequency $F_{\max}$ being recorded [22]. This process is repeated using various functional test patterns, resulting in a multi-label dataset. For each device, the most critical pattern is the one with the lowest $F_{\max}$ value. In order to ensure robustness in the measurement process and exclude outliers from the training procedure, devices with $F_{\max}$ deviating by more than 2.5 standard deviations from the wafer median in at least one functional pattern are eliminated. This ensures a high-quality set of labeled devices for ML training.

### B. The SMONs

The structure and sensitivity of the SMONs significantly influence the performance prediction model. To address this, the goal is to incorporate a diverse array of SMON types into the MCU. They also need to be spatially distributed to account for Within-die (WID) process variation, particularly prominent as feature sizes of manufacturing technologies are reduced further [23].

Thus, an SMON module is designed to contain a heterogeneous set of various SMONs. Multiple instances of identical SMONs modules are strategically placed throughout the MCU to ensure comprehensive spatial coverage.

An SMON module comprises generic and design-dependent ROs. Generic ROs include inverter gates, NAND gates and NOR gates, which are taken from cell libraries used in the design of the MCU. Meanwhile, the design-dependent ROs are replicated functional paths derived from the design.

The feature set includes SMONs from multiple SMON modules and functional path ROs, offering a robust foundation for understanding the chip's behavior across various scenarios.

In order to streamline information from different SMON modules, aggregation can be employed. For each SMON in various modules, a single value is extracted by an aggregation function (like the mean or the median). This approach, referred to as the "Virtual Module (VM)" [8], serves the dual purpose of reducing the number of SMONs analyzed by the ML model and capitalizing on the correlations and variations among SMONs at different locations. The VM strategy allows for a simplified ML algorithm, even in the presence of multiple SMON modules, and permits decreasing variance and removing possible outliers from the measured SMON values.

### C. Machine Learning

Training an ML model involves establishing a relationship between inputs and outputs based on available labeled data. The model evaluates certain features to generate an output. For instance, simple linear regression algorithms combine input features linearly using weights assigned during training.

The number of samples required to accurately estimate a function increases exponentially with the number of input variables, also known as the dimensionality of the data [24]. In simpler terms, having more features necessitates a larger dataset to construct robust ML models. To address this, dimensionality reduction techniques such as FS [17] can be employed. RFE [25], [26] stands out as a popular algorithm for FS. RFE effectively identifies the features the more relevant for predicting the target variable based on an underlying ML model. The hyperparameters involved in RFE include the number of features to select and the choice of the ML model to address the underlying supervised problem. RFE operates by utilizing an external estimator that assigns weights to features, such as coefficients in a linear model or feature importance scores derived from impurity measurements in decision trees [27]. It recursively considers smaller sets of features, starting with the entire set, and prunes the least important features until the desired number of features is reached. RFE with Cross Validation (RFE-CV) incorporates cross-validation to determine the optimal number of features automatically [26].

### D. Regularized Linear Models

Some ML algorithms can handle FS internally: these are called *embedded methods*. There, the feature selection is automatic as part of the model training process, eliminating the need for separate feature selection steps. As an example, Regularized linear models can perform embedded feature selection. These models add a penalty term (usually the $L_p$ norm) computed on the coefficients' vector in the ordinary least-square linear regression objective function:

$$\min_{w} \frac{1}{2n_{\text{samples}}} ||Xw - y||_2^2 + \alpha \left( \sum_{i=1}^{n} |w_i|^p \right)^{\frac{1}{p}} \quad (1)$$

Some examples of embedded FS based on regularization are Lasso Regression, ElasticNet, and Orthogonal Matching Pursuit (OMP) [28]–[30]. Lasso is a linear regression model with $L_1$ regularization. It tends to drive some coefficients to exactly zero, effectively performing FS. ElasticNet is an extension of Lasso that combines $L_1$ and $L_2$ regularization. While $L_0$ regularization is non-convex and computationally challenging, some optimization methods like OMP approximate the fit of a linear model with constraints imposed on the number of non-zero coefficients (ie. the $L_0$ pseudo-norm).

## IV. Motivations

As discussed in Section III-B, incorporating a greater number of SMONs at various locations on the MCU could enhance the handling of WID process variation, providing more comprehensive insights into MCU performance. While having a multitude of SMONs might contribute to superior performance predictions, practical constraints limit the feasibility of this approach for two key reasons:

- **Physical Area and Overhead**: Each additional SMON integrated into the design occupies physical space on the chip. Incorporating hundreds of SMONs leads to a noticeable area overhead, contributing to current leakages. Given that SMONs serve testing purposes exclusively and hold no functional value for customers, minimizing the occupied area is essential.
- **Curse of Dimensionality**: From an ML perspective, an increased number of SMONs results in a higher feature count for the predictive models. This may lead to suboptimal models due to the COD, or models with a higher footprint than needed (in terms of number of coefficients).
- **Interpretability** : The higher the number of SMONs, the more difficult is to catch which of them is relevant for the performance prediction task.

Technologically, a reduced feature set lowers production costs by incorporating only the most informative sensors into future products and also mitigates the associated physical space and current leakage concerns. Additionally, a feature ranking emerges as a byproduct of the proposed FS techniques, offering valuable insights for diagnostic purposes. This ranking informs test engineers about the correlation between individual SMONs and patterns, guiding selection and design decisions.

Reducing the feature set also contributes to enhanced generalization performances of ML models, allowing for simpler models that operate on fewer inputs. This strategic reduction aligns with both technological and machine learning considerations, facilitating cost-effective and efficient integration of SMONs into MCU designs.

Also, opting for a linear model over more sophisticated methods has several advantages including, simplicity and interoperability, computational efficiency, avoidance of overfitting, and small data requirements.

In a previous work [8], authors have used a Feature Selection approach based on Recursive Feature Elimination (RFE). As stated in Section III, RFE involves iteratively fitting a model and removing the least important features until a predetermined number of features is reached. Typically, the number of features to retain is not known in advance. To identify the optimal number of features, RFE can be used in a cross-validation loop (RFE-CV) to score various feature subsets and select the set that yields the best performance. RFE relies on an underlying model: we can use more than one ML model [8], benefitting from the different kinds of feature ranking they perform (Ridge Regressor, Lasso, Elasticnet, Random Forest [31], Gradient Boosting [32]). This procedure, based on different types of learners, permits reaching an optimal feature set, but at the cost of a high computational time: since this approach involves training several models on different feature sets of different sizes.

Specifically, the process outlined in [8] might require a dedicated server for approximately a week, and need to be repeated for each new device family. Additionally, as the number of SMONs under consideration rises, the computational time significantly escalates.

Thus, a simpler procedure is needed, that can lead to an acceptable feature subset in a reasonable time.

## V. Proposed Approach

The approach followed in this work using regularized linear models to select the most relevant feature set for each available functional testing pattern, finally ranking the SMONs based on their importance in the supervised performance screening task. The developed approach can be summarized in 3 steps, explained in detail in this section:

1) Obtain a ranking for the SMONs
2) Identify a reasonable number of features to keep
3) Train the models with the best SMONs

The SMON ranking was obtained by repeatedly applying steps of feature selection. We looped over all the SMON modules (avoiding location bias in the result), over all the 10 functional testing patterns considered, for 6 different training-test splits and using 3 different regularized linear models (Lasso, ElasticNet, OMP). This was done to find feature subsets most likely independent by physical location and choice of training samples.

For each step of feature selection, if an SMON was selected by the underlying feature selection method, we increase a counter for that SMON. SMONs ranking could give test

engineers insights on the importance of each SMONs for the downstream performance screening task. Algorithm 1 sketches how the ranking of the SMONs based on linear models is computed.

We compared the proposed method with a more sophisticated one that relies on RFE, from previous work [8].

For the RFE procedure, as underlying estimators, we also included non-linear models (Decision Trees and Random Forests). This can make the SMON ranking creation more accurate, but it comes at the cost of a not negligible increase in the computation time.

The RFE procedure starts considering the whole feature subset and pruning feature until a pre-determined number of features is reached. Thus, we need to fix it as an hyper-parameter. To do this, we can use the RFE-CV procedure. We used four different algorithms: Random Forest and three Linear models with regularization (Ridge, Lasso, ElasticNet) as base estimators, with no feature transformation (Fig. 1).

For the second step, we can still use a regularized linear model like Lasso Regressor or OMP as a Feature Selector. We trained these two models on all the features (directly on SMONs frequency, no feature transformation) on 5 different train splits, and we recorded the mean number of non-zero coefficients $n$ among the run.

Finally, we selected the first $n$ SMONs from the ranking and we built a Polynomial Ridge Regressor model: we applied a degree-2 polynomial transformation of the SMONs frequency.

The Virtual SMON module strategy [8] was implemented: for each selected SMON, its value was computed by considering the median values among all the identical SMONs from the different SMON modules (as stated in Section III-B).

## VI. Experimental Evaluation

### A. Experimental Setup

The proposed methodology has been validated on a dataset composed of a thousand samples with hundreds of SMONs, divided into several identical SMON modules.

The evaluation was performed with a 5-folds CV stratified per wafer. All experiments were performed in Python. Experiments run on a server equipped with a dual-socket AMD ®EPYC 7301 16-Core CPU @ 3.20GHz, 128GB of RAM. Each column of the dataset was scaled by subtracting the mean and dividing it by the variance (*Standard Scaler*). As the final model, we used a pipeline composed of the Standard Scaler, a polynomial transformation of the input features, and a Ridge Regressor. This model is called Polynomial (or Poly) Ridge [4]. Results are presented in terms of normalized Root Mean Square Error (nRMSE) and the coefficient of determination ($R^2$). RMSE is a popular regression performance index [33] 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 concerning the mean frequency of the samples. $R^2$ is the proportion of the variation in the dependent variable that is predictable from the independent variable(s) [34]. A regressor that perfectly fits the data would have an $R^2$

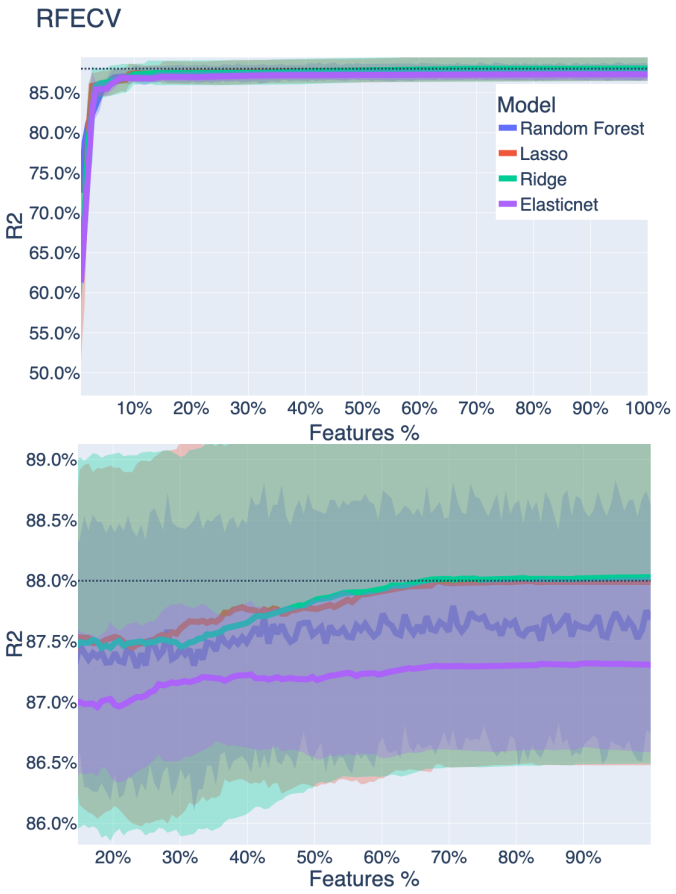


Fig. 1. RFECV algorithm run on the full features set, mean error across 5-folds plus error band. On the x-axes the number of features, on the y-axes the R2 score. The maximum performance (dotted line) is reached with about 65% of SMONs, but going over 15% of SMONs, the gain increases only by 0.5% of R2 scores (lower plot, with a close-up in the range 15%-100% of features.

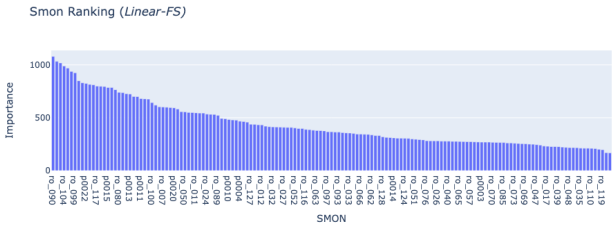


Fig. 2. Example of SMONs ranking obtained. On the Y axis, the importance of each SMONs (RO or replicated functional paths)

score of 1 (or 100%). A dummy regressor that always predicts the mean value of the target has an $R^2$ score of 0.

To find a good number of features to select, we used Lasso Regressor on 5 different train splits (step 2, as stated in Section V).

The regularized linear regressors used for the SMON ranking (i.e. Lasso, ElasticNet, OMP) used an internal 5-fold CV to tune hyper-parameter $\alpha$ parameters in Eq. (1).

To validate this (and implement the RFE-based comparison, as stated in Section V), we run the RFE-CV algorithm (5-fold

**Algorithm 1:** SMONs Ranking creation

```
Result:
SMONs ranking SR
Data:
S = Set of Regularized Linear Models
M = SMONs modules
m = # of SMONs per modules
k = Number of Training Splits
Y = Testing programs
D = SMONs Frequencies;
Init:
SR[m] = 0
for currentSmons in M do
    for X in D.split(k) do
        for y in Y do
            for model in S do
                model.fit(X[currentSmons], y)
                selectedSmons = [1 if abs(c) > 0 else 0
                  for c in model.coef]
                SR[selectedSmons] += 1
            end
        end
    end
end
```

CV and 4 base-regressors) on the median-VM (Section III-B).

### B. Results

We can compute the score of the SMON grouping by patterns, obtaining the importance of SMONs for each pattern or we can group by location, or, to compute a general score, we can sum up all the results (Fig. 2). This latter approach 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.

From step 2 of the proposed approach (Section V), we obtained about 17% ± 1.33% of non-zero coefficients for Lasso models. We also use an OMP algorithm with an internal 5-folds CV to automatically find the best number of features. This latter model chose about 14% ± 4%. The two results are similar. The outcome of the models was, in our setting, practically immediate (few seconds). As a comparison, we can use the RFE-CV procedure: from Fig. 1, we can see that we practically find a plateau in terms of $R^2$ score with about 66% of the total number of SMONs, for each algorithm. This number is thus the outcome of the RFE-CV procedure. But going over 15-17% of SMONs, the gain increases only by 0.5% of $R^2$ scores. Notably, the size of each feature set tried in RFECV procedure (x-axis of Fig. 1) does not reflect the actual number of non-zero coefficients in the underlying model: this is especially true if we use a regularized linear model as the underlying estimator. With Lasso, RFE-CV procedure stated that the best possible performance (about 88% of $R^2$ score) is obtained with 94% of the available SMONs. But if we look at the non-zero coefficients of the model trained with that number of SMONs, only a few of them are non-zero entries (i.e. about 17% of the total: the same obtained without the RFE-CV, as stated previously). This fact, combined with the analysis of the RFE-CV curves, confirms that Lasso was able to find the correct number of coefficients, even lower than the RFE-CV procedure. Notably, in our setting, a whole day (about 24 hours) was required to obtain Fig. 1, while only few seconds were needed to fit Lasso and OMP-CV on 5 train-test splits. 17% of the SMONs are sufficient to retrieve relevant information about the performance of the device. Also, this justifies the fact that relying only on regularized linear models, letting them choose among the whole feature without inserting an additional step of optimization loop (i.e. the RFE step) is beneficial because they can effectively choose a good feature set among the whole SMONs set. This led to a simpler and faster feature selection method: the proposed SMON ranking procedure comes with a high speed-up in the computation time. While more than an entire week of computation and CPU-time was needed to obtain the feature ranking from [8], dropping the RFE step decreases the time to only a few hours(i.e. from about 168 hours for SMONs ranking plus 24 hours for choosing the size of the feature set (192 hours) to only 4: roughly 2% of the time). The features set from the RFE procedure [8] (namely, RFE-FS) come with higher accuracy with respect to performing no FS, but with lower results concerning the proposed approach (namely, Linear-FS): this may be due to the higher number of ROs chosen. But even decrease the number of chosen SMONs up to 17% (number found by regularized linear models), we still have lower performance concerning Linear-FS (Table I). The simpler alternative, based only on regularized linear models is the best in terms of execution time and prediction accuracy: 89.69% $R^2$ and a gain of several days. Pruning a high number of SMONs is beneficial when further feature interaction transformations are computed (like the Polynomial transformation). Imagine having 150 SMONs on a chip. Since we are using a Polynomial Regressor, computing feature interaction would lead to 11475 features, and thus to 11475 coefficients. By computing a feature ranking, retaining only 17% of the SMONs, we would have 377 polynomial terms. But, if needed, for example, to reduce further the model footprint in terms of the number of coefficients, it is possible to add a further step of $L_1$ regularization after the polynomial transformation, and before feeding the ridge regressor. This would reduce the number of coefficients thanks to Lasso ability to find a sparse solution. We called this solution Lasso+Ridge (Table I). Starting from 43% of Poly terms, Lasso keeps only 0.46% non-zero coefs (Table I), with only a little drop in performance. However, as the number of features increases, also the time needed to train a Lasso model increases: starting from the whole features set, about 2 hours are needed to train the models, while it takes seconds on the reduced features set into account. This is due to polynomial transformation, in which the dimensionality of the outcome increases polynomially with the number of SMONs. We mentioned the ability of Regularized Linear models to crucially decrease the risk of overfitting. We can state this

TABLE I
PREDICTION ERROR WITH DIFFERENT FEATURE SETS AND MODELS (AVERAGE RESULTS ON 5-SPLITS)

| FS Method | Model. | Used SMONs(%) | Input Features(%) | nRMSE% | $R^2$% | Time (h) |
|---|---|---|---|---|---|---|
| No-FS | Ridge | 100% | 100% | 1.53 | 88.07 | – |
| RFE-FS | Ridge | 66% | 43% | 1.47 | 88.85 | 192 |
| RFE-FS (2) | Ridge | 17% | 3% | 1.42 | 89.61 | 192 |
| Linear-FS | Ridge | 17% | 3% | 1.42 | 89.69 | 4 |
| No-FS | Lasso+Ridge | 100% | 0.80% | 1.50 | 88.54 | – |
| RFE-FS | Lasso+Ridge | 66% | 0.62% | 1.44 | 89.36 | 192 |
| RFE-FS (2) | Lasso+Ridge | 17% | 0.33% | 1.44 | 89.37 | 192 |
| Linear-FS | Lasso+Ridge | 17% | 0.46% | 1.43 | 89.63 | 4 |
| No-FS | Linear | 100% | 100% | 2.24 | 74.39 | – |
| RFE-FS | Linear | 66% | 43% | 2.44 | 69.50 | 192 |
| RFE-FS (2) | Linear | 17% | 3% | 1.86 | 82.29 | 192 |
| Linear-FS | Linear | 17% | 3% | 1.75 | 84.30 | 4 |

by substituting Ridge with a simpler Linear Regression (no $L_2$ regularization). Final prediction accuracies are much lower (Table I). Introducing some kind of regularization enhances the generalization capabilities of the model.

## VII. CONCLUSIONS

We presented a feature selection framework to be used in MCU performance screening. SMON values are good alternate measures for performance prediction, and industries/test engineers may use a large number of SMONs to catch performance variation. Our framework permits to rank SMONs based on their importance in the performance prediction task. The experiments showed that linear models with regularization can effectively reduce the number of SMONs automatically, without relying on RFE loop. They also can compact information, reducing the number of coefficients of the final linear model. The obtained feature ranking obtained superior accuracy with respect to RFE, and can be produced quickly. In general, this approach is useful for every kind of situation in which we have features highly correlated with each other and relations feature-target linear or pseudo-linear.

## 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 *IEEE International Test Conference (ITC)*, 2020.

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

[4] N. Bellarmino *et al.*, "A Multi-Label Active Learning Framework for Microcontroller Performance Screening," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2023.

[5] J. Chen *et al.*, "Data learning techniques and methodology for fmax prediction," in *IEEE International Test Conference (ITC)*, 2009.

[6] J. Chen *et al.*, "Selecting the most relevant structural fmax for system fmax correlation," in *28th VLSI Test Symposium (VTS)*, 2010.

[7] M. Sadi *et al.*, "SoC Speed Binning Using Machine Learning and On-Chip Slack Sensors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2017.

[8] N. Bellarmino *et al.*, "Feature Selection for Cost Reduction In MCU Performance Screening," in *IEEE 24th Latin American Test Symposium (LATS)*, 2023.

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

[10] 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.

[11] T. B. Chan *et al.*, "DDRO: A novel performance monitoring methodology based on design-dependent ring oscillators," in *Thirteenth International Symposium on Quality Electronic Design (ISQED)*, May 2012.

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

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

[14] 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 (TCAD)*, 2008.

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

[16] W. Jia *et al.*, "Feature Dimensionality Reduction: a Review," *Complex & Intelligent Systems*, Jun. 2022.

[17] I. Guyon *et al.*, "An Introduction to Variable and Feature Selection," *The Journal of Machine Learning Research*, Mar. 2003.

[18] S. Larguech *et al.*, "Efficiency evaluation of analog/rf alternate test: Comparative study of indirect measurement selection strategies," *Microelectronics Journal*, 2015.

[19] M. J. Barragan *et al.*, "A procedure for alternate test feature design and selection," *IEEE Design and Test*, 2015.

[20] U. M. Khaire *et al.*, "Stability of feature selection algorithm: A review," *Journal of King Saud University - Computer and Information Sciences*, 2022.

[21] 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.

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

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

[24] R. Bellman, *"Adaptive Control Processes: A Guided Tour"*, 1961.

[25] 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.

[26] I. Guyon *et al.*, "Gene Selection for Cancer Classification Using Support Vector Machines," *Machine Learning*, Jan. 2002.

[27] L. Breiman, "Random forests," en, *Machine Learning*, Oct. 2001.

[28] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, 1996.

[29] H. Zou *et al.*, "Regularization and Variable Selection via the Elastic Net," *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 2005.

[30] J. A. Tropp *et al.*, "Signal Recovery From Random Measurements Via Orthogonal Matching Pursuit," *IEEE Transactions on Information Theory*, 2007.

[31] Q. Lv *et al.*, "Enhanced-Random-Feature-Subspace-Based Ensemble CNN for the Imbalanced Hyperspectral Image Classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2021.

[32] Z. He *et al.*, *Gradient Boosting Machine: A Survey*, 2019.

[33] T. Chai *et al.*, "Root Mean Square Rrror (RMSE) or Mean Absolute Error (MAE)?– Arguments Against Avoiding RMSE in the Literature," *Geoscientific Model Development*, Jun. 2014.

[34] 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.