COSMO: COmpressed Sensing for Models and logging Optimization in MCU Performance Screening

COSMO: COmpressed Sensing for Models and logging Optimization in MCU Performance Screening

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

***Abstract*—In safety-critical applications, microcontrollers must meet stringent quality and performance standards, including the maximum operating frequency $F_{\max}$. Machine learning models have proven effective in estimating $F_{\max}$ by utilizing data from on-chip ring oscillators. Previous research has shown that increasing the number of ring oscillators on board can enable the deployment of simple linear regression models to predict $F_{\max}$. However, the scarcity of labeled data that characterize this context poses a challenge in managing high-dimensional feature spaces; moreover, a very high number of ring oscillators is not desirable due to technological reasons. By modeling $F_{\max}$ as a linear combination of the ring oscillators' values, this paper employs Compressed Sensing theory to build the model and perform feature selection, enhancing model efficiency and interpretability. We explore regularized linear methods with convex/non-convex penalties in microcontroller performance screening, focusing on selecting informative ring oscillators. This permits reducing models' footprint while retaining high prediction accuracy. Our experiments on two real-world microcontroller products compare Compressed Sensing with two alternative feature selection approaches: filter and wrapped methods. In our experiments, regularized linear models effectively identify relevant ring oscillators, achieving compression rates of up to 32:1, with no substantial loss in prediction metrics.**

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

I. Introduction

In safety-critical sectors such as automotive and aerospace, ensuring the reliability of microcontrollers (MCUs) is crucial. This involves identifying devices that fulfill specific criteria. Within this context, the maximum operating frequency ($F_{\max}$) holds particular significance due to its impact on the overall performance of the MCU. $F_{\max}$ influences the speed at which the MCU can process data, which is crucial in applications where timely processing is essential. If the $F_{\max}$ is not reached, this can lead to problems such as timing violations and system instability, which can endanger the reliability and safety of the entire system. Traditional approaches to determine $F_{\max}$ require extensive testing at varying clock frequencies (*Speed Binning*, [1], [2]). This method is time-intensive, relies on costly test setups, and merely yields binary pass/fail outcomes. In response to these challenges, machine learning (ML) regression models have been proposed to predict $F_{\max}$ of MCUs based on alternative easy-to-acquire on-chip measurements, offering significant time savings compared to traditional methods [1]–[3]. In particular, previous research showed that frequency values from ring oscillators (ROs) can be linked with the device's speed $F_{\max}$ and used as features for ML models [3], [4].

Increasing the number of ROs could enhance the information available regarding the eventual speed of the device [5], potentially improving the accuracy of ML models. However, this approach involves costs from both predictive and technological standpoints: possible reductions in accuracy, increased need for training samples to efficiently fit data distributions, higher monetary production costs, and increased current leakage [5].

This paper focuses on exploiting Compressed Sensing (CS) and regularized linear models for optimizing ML models for MCU performance screening. By considering ROs as a proxy for the $F_{\max}$, the challenge of constructing an ML model for $F_{\max}$ is here approached as a linear system identification (or linear regression) problem. We propose CS and regularized linear models as a unifying theory used to both build ML models and perform Feature Selection. By pruning irrelevant or uninformative features/SMONs in the context of MCU performance screening, the goal is to reduce the models' footprint, specifically in the number of coefficients/features, while maintaining satisfactory prediction error

Feature selection techniques are categorized into three types: filtering, wrapper, and embedded approaches [6]. The paper highlights the use of CS principles as an embedded feature selection approach to promote sparsity in the coefficients of the models. Specifically, we concentrate on linear techniques with regularization, that incorporate penalty terms within the error metric used for building the linear model.

The primary objectives and contributions of this study are as follows:

- Propose a novel mathematical approach for modeling the relationship between ROs and $F_{\max}$, facilitating the use of more straightforward and computationally efficient methods for solving the regression problem.
- Establish that CS theory, when implemented via reg-

* Politecnico di Torino (Turin, Italy). † Infineon Technologies AG (Munich, Germany). ‡ Technical University of Munich (Munich, Germany). Authors are listed in alphabetical order.

ularized linear models, can be effectively utilized for predictive performance screening.
- Demonstrate that the use of regularized linear models contributes to a reduction in model size without significantly compromising prediction accuracy.
- Extend the application of this methodological framework across various MCU products, even in cases where the relationship between SMONs and $F_{\max}$ is not strictly linear.
- Evaluate the effectiveness of employing both convex and non-convex penalty terms within the context of regularized linear models.
- Conduct a comparative analysis of regularization techniques against filter and wrapper methods for feature selection.

This innovative approach achieves a reduction in complexity and computational cost while maintaining or enhancing model performance.

The rest of the paper is organized as follows: Sections II and III provide the necessary background information, including data collection processes for ML algorithms and an introduction to CS (Section III-A) and regularization (Sections III-B to III-E). Section IV presents related works on the topic. In Section V, the motivations for deploying compressed sensing are given. In Section VI, we detailed the proposed approach. Section VII presents the experimental evaluation. Section IX provides a discussion about the key takeaways and findings. Finally, Section X draws the conclusions.

II. Background: Testing, SMONs, and Labels

Testing is a crucial aspect throughout the integrated circuits (ICs) life cycle, from design to in-field evaluations [7]. Specifically, MCU Performance Screening evaluates ICs focusing on their maximum operating frequency $F_{\max}$. This process entails applying test patterns at varying clock frequencies to determine the operational capabilities of the ICs, screening out those with frequencies below predetermined thresholds. This process is called *Speed-Binning*, and it is carried out in discrete-step [2], giving binary information about the test outcome (Pass/Fail).

In order to facilitate performance screening, ROs are utilized. ROs consist of in-series cells from standard libraries, with an overall inverting behavior; thus, these structures oscillate. ROs' oscillation frequency can be linked with the device's speed $F_{\max}$ [1]–[3]. In particular, these can be used as *features* for an ML *regression* model [1], [3], [4], [8], able to predict *continous* value for $F_{\max}$. ROs used to be linked with devices' $F_{\max}$ speed are called *Speed MONitors* (SMONs). The number of SMONs on a chip varies depending on the product type, ranging from tens to hundreds. Each SMON on board is slightly different from the others. In the design phase of MCU, engineers should decide which type of SMONs to place on board. This choice can be made based on previous information from legacy products or by data-driven approaches [5]. While the number remains constant within a technology family (e.g., family A), it may differ between products of different families (e.g., family B).

Labeling MCUs based to obtain a continuous (not discrete) $F_{\max}$ measurement is not integrated into the standard production test flow (Fig. 1). This necessitates individual measurements, by mounting MCUs on specific evaluation boards and running functional test patterns. This process is resource-intensive and conducted on only a limited subset of manufactured devices. Consequently, labeled data is scarce [4], [9], posing a challenge for ML applications.

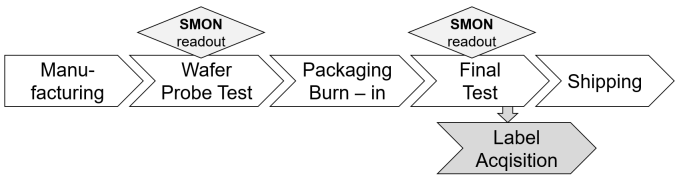

Fig. 1. Data collection steps through the manufacturing

III. Dimensionality Reduction, Feature Selection and Compressed Sensing

Dimensionality reduction is crucial in ML. It involves reducing the number of features an ML model has to analyze. The goals are: improving model performance, reducing computational complexity, and mitigating the risk of overfitting. Two primary methods for dimensionality reduction exist: Feature Extraction (FE) [10] and Feature Selection (FS) [10]. FE aims to produce a new set of features, combining the original ones. Principal Component Analysis (PCA) is a popular technique for FE [10]. FS, instead, involves selecting the most relevant subset of features from the pool of input variables and discarding redundant ones. Methods for FS can be categorized into three main groups: Filter, Wrapper, and Embedded methods [6]. Filter methods assess the features' relevance ranking them based on certain statistical measures. For example, correlation-based FS techniques measure the statistical relationship between each feature and the target variable. Common metrics include the Pearson correlation coefficient and Spearman Rank [11]. Features with the highest correlation with the target are selected [10], in a univariate supervised fashion. Alternatively, it is possible to filter out highly correlated features, in an unsupervised fashion. Wrapper methods utilize an underlying ML model to evaluate the performance of different feature subsets, selecting the best based on error metrics [12]. These methods are computationally heavier than filter methods, but often yield better results. Recursive Feature Elimination (RFE) is a widely-used wrapper method [10]. Embedded methods incorporate feature selection in the model training process, eliminating the need for a separate feature selection step (as in Decision Trees and Random Forests, [13]). Other types of ML models with embedded feature selection are the Linear Models that introduce some regularization in the training procedure. These methods are deeply explained in Sections III-B to III-E.

A. Compressed Sensing

Signal processing has traditionally involved acquiring, analyzing, and reconstructing signals through methods such as Fourier analysis and sampling theory. Traditional techniques assume that signals are uniformly and densely sampled, leading to Nyquist-Shannon sampling requirements. However, in

many applications, acquiring and storing large amounts of data can be impractical and costly [14]. To address this point, CS [15] emerged as a theory that proves that signals admitting sparse representations can be accurately reconstructed from a reduced set of linear measurements. Large-dimensional systems often exhibit essential behavior describable by sparse models, with a number of parameters significantly lower than the number of state variables [16].

More in detail, CS provides conditions for recovering a sparse vector $x \in \mathbb{R}^n$, with $k < n$ non-zero components, from linear regression $y = Ax + \eta \in \mathbb{R}^m$, with $m < n$, and η a measurement-related noise factor [17]. The development of efficient algorithms for recovering sparse signals, such as Basis Pursuit and Orthogonal Matching Pursuit [18], has contributed to the practical implementation of CS. Also, regularized linear models (deeply discussed in the next section) are suitable for promoting sparsity, particularly the LASSO (Least Absolute Shrinkage and Selection Operator) [19]. Over the years, CS has found applications in diverse fields, including medical imaging, radar, communications, and more [14], [20].

B. Regularized Linear Models

Regularized linear models are a class of linear regression models that incorporate regularization techniques to improve accuracy and prevent overfitting. These techniques add a penalty term to the ordinary least-square (OLS) linear regression objective function, influencing the optimization process to reduce the number of active parameters. The optimization problem is

$$\min_w \frac{1}{2n} \|Xw - y\|_2^2 + \alpha \sum_{j=1}^{m} \text{pen}(|w_j|) \tag{1}$$

where the first component is the OLS objective and the second is a penalty term on the coefficient of the model. Here, $X \in R^{n,m}$ is the observations' matrix, $y \in R^n$ is the regression target, $w \in R^m$ is the coefficients vector, and α is the regularization strength hyper-parameter. The penalty is usually an ℓ_p norm of the coefficient vector w, defined as

$$\|w\|_p = \left(\sum_{i=1}^{m} |w_i|^p \right)^{\frac{1}{p}}. \tag{2}$$

The convex ℓ_1 and ℓ_2 norms are popular regularizations, and they give rise to ridge regression [21] and LASSO [19], respectively. While ℓ_2 regularization has mainly a smoothing effect, ℓ_1 regularization has a sparsity promoting effect, that yields an effective variable selection in linear regression. More recently, non-convex regularization penalties [22] have been introduced, which improve the variable selection performance with respect to classic ℓ_1 regularization. Examples of effective non-convex regularization penalties are Log Sum Penalty (LGP, [23]), Minimax Concave Penalty (MCP, [24]), Smoothly Clipped Absolute Deviation (SCAD, [25]) and the ℓ_p norm with $0 < p < 1$ [22].

C. Convex Regularization

The LASSO problem is (1) with ℓ_1 regularization, i.e., $\text{pen}(|w_j|) = |w_j|$. The ℓ_1 regularization encourages sparsity in the model, leading several coefficients to become exactly zero. LASSO is a convex problem, therefore it is mathematically affordable. It represents an effective recovery strategy for CS, i.e., for linear regression problems with $w \in \mathbb{R}^m$, $y \in \mathbb{R}^n$ and $n < m$.

In ridge regression [21], the optimization problem is (1) with $\text{pen}(|w_j|) = w_j^2$. The ℓ_2 regularization helps in reducing the impact of multicollinearity in the data, penalizing large coefficients and shrinking them towards zero, but it does not lead to sparsity in the model.

ElasticNet [26] is a linear regression model trained with both ℓ_1 and ℓ_2 norm regularization of the coefficients. This combination allows for both learning a sparse model (like in LASSO) and maintaining the regularization properties of ridge regression. The corresponding optimization problem is

$$\min_w \frac{1}{2n} \|Xw - y\|_2^2 + \alpha\rho \|w\|_1 + \frac{\alpha(1-\rho)}{2} \|w\|_2^2.$$

The convex combination of ℓ_1 and ℓ_2 is controlled using α and ρ parameters.

D. Orthogonal Matching Pursuit

Beyond convex regularization, greedy algorithms are traditionally used for CS. In particular, OMP [27] approximates the fit of a linear model with constraints imposed on the number of non-zero coefficients (i.e., the ℓ_0 pseudo-norm), finding a coefficients vector with a fixed number of non-zero elements:

$$\arg\min_w \|y - Xw\|_2^2 \text{ subject to } \|w\|_0 \leq n_{\text{nonzero\_coefs}}. \tag{3}$$

OMP is based on a greedy algorithm that includes at each step the features most highly correlated with the current residual, i.e. the difference between an observed value y_i and its corresponding predicted value $\hat{y}$. Once the feature is selected, the signal is orthogonally projected to the span of the selected feature, the residual is recomputed, and the process repeats. OMP can be considered as a convex approach because each step in the algorithm involves solving a convex least squares problem. However, the overall optimization problem solved by OMP is non-convex.

E. Non-Convex Regularization

In the last years, continuous non-convex sparsity-promoting regularization has been attracting substantial attention [22]. While solving LASSO is affordable thanks to its convexity, the global minimum suffers from bias with respect to the true solution, due to the penalty term. In contrast, non-convex regularization, with shape closer to the ℓ_0 norm, may reduce the bias [24]. Due to the non-convex nature of the objective function, no theoretical assurances are provided regarding the attainment of the global minimum [22]. However, research has demonstrated that non-convex linear models exhibit superior performance within a fixed time frame, achieving higher accuracy levels in less time compared to convex optimization

methods [22], [28]. Moreover, it can produce sparser models, which is desirable for interpretability and feature selection, and it may accelerate the rate of convergence of the solution algorithms [29]. The use of non-convex regularization has become popular in CS, in the framework of ℓ_1 reweighting techniques [30], while recently it has been also analyzed for pruning purposes in deep neural networks [31].

Among the most popular non-convex regularizers, MCP approximates the ℓ_1 penalty when α is arbitrarily large. In particular, MCP maintains the shape of the ℓ_1 norm around the origin while approximating the scaled ℓ_0 norm away from the origin [24]. The MCP penalty is

$$\text{pen}(w) = \begin{cases} \alpha w - \frac{w^2}{2\gamma} & \text{if} \quad w \leq \alpha\gamma \\ \gamma \frac{\alpha^2}{2} & \text{if} \quad w > \alpha\gamma \end{cases} \tag{4}$$

in which $\alpha > 0$ is the regularization strength and $\gamma > 0$ is an hyper-parameter.

SCAD [25] penalizes large coefficients less severely than LASSO, leading to more stable and accurate model estimation, especially when dealing with correlated predictors. The SCAD penalty is

$$\text{pen}(w) = \begin{cases} \alpha w & \text{if} \quad w \leq \alpha \\ 2\alpha\gamma w - w^2 - \frac{\alpha^2}{2}(\gamma - 1) & \text{if} \quad \alpha < w \leq \alpha\gamma \\ \alpha^2 \frac{\gamma+1}{2} & \text{if} \quad \alpha\gamma < w \end{cases} \tag{5}$$

where $\alpha > 0$, $\gamma > 0$.

LGP is often adopted as a replacement for the ℓ_0 pseudo-norm in CS and low-rank optimization. [23] The LGP penalty is

$$\text{pen}(w) = \sum_{j=1}^{m} \log\left(1 + \frac{|w_j|}{\epsilon}\right) \tag{6}$$

where $\alpha > 0$, $\gamma > 0$.

Among the ℓ_p regularizers, with $0 < p < 1$, the case $p = \frac{1}{2}$ is often used [17], [22].

IV. Related Work

In recent years, the utilization of ML in both design and testing has garnered considerable attention, with a multitude of data analytic methods based on ML being extensively investigated [32].

Also, the interest in predicting the maximum operating frequency $F_{\max}$ of MCUs in safety-critical applications has grown. The early utilization of ML models to establish a relationship between structural and functional $F_{\max}$, first introduced in [1], was studied extensively by several researchers. While previous studies have explored the possibility of mapping indirect measurements and $F_{\max}$ using small sample sizes or simulated data [1], [2], [33], this approach has now been validated using real-world data from MCU characterization [3], [4], [34]. Additionally, various ML models, including both linear [3] and non-linear [4], [34], have been presented in the literature. Active Learning was employed in [9] to reduce the training set size by selecting informative samples for model derivation, and outlier detection techniques were evaluated to identify noisy data and outliers [35]. In analog circuits, several studies have been done on feature selection [36], [37], while the importance of feature selection in MCU performance screening was firstly addressed in [5]: however, these methods often rely on filter or wrapper approaches. Filtering approaches are usually univariate, considering only one feature at once. Wrapper methods, instead, have a high computational cost and depend on the model and data on which they are trained, with the risk of overfitting [38].

V. Reasoning behind Compressed Sensing and Regularized Models

In recent years, significant research efforts have focused on developing predictive models for MCU performance screening [3]–[5], [9], [35]. In this field, the number of SMONs on board is positively correlated with the amount of information related on the devices' $F_{\max}$ [5]. However, a higher number of SMONs presents challenges from both technological and data-analysis perspectives.

From a technological standpoint, having a large number of SMONs is undesirable, as they are solely used for testing purposes and contribute to increased current leakage, occupy more physical space on the die, and raise production costs [4]. Moreover, implementing ML-based performance screening at scale leads to significant storage challenges. Storing thousands of features for data logging across millions of devices could result in excessive storage requirements.

From a predictive modeling perspective, a large number of features can lead to the "Curse of Dimensionality" (CoD) [10], which necessitates a larger training set for effective model development. However, obtaining a large labeled dataset can be prohibitively expensive and time-consuming, often limiting the number of samples to just hundreds or thousands (Section II). Additionally, this may not always be feasible since critical decisions must be made during the MCU design phase when data availability is limited. Consequently, techniques such as Data Augmentation [35], Transfer Learning [8], and Active Learning [9], which rely on diverse and abundant training samples to improve model robustness, may not be viable.

Having a large number of features or SMONs allows approximating their relationship with $F_{\max}$ using linear models [5]. These are favored for their simplicity, interpretability, and lightweight nature, and can potentially be deployed directly on the MCU for in-field evaluation. A linear regression model requires storing only $n+1$ coefficients, where n is the number of features and the additional coefficient is the intercept term. This results in a limited memory footprint, making linear regression models efficient in terms of storage requirements for embedded systems.

However, this creates a trade-off: although more SMONs provide greater insight into device speed, they also introduce higher costs. Some SMONs sets have SMONs very similar to each other, and thus highly linear correlated (Fig. 2). For this reason, not all the features may be important for the ML models, or not all of these ROs may be of interest for the downstream tasks. Therefore, the number of SMONs should be minimized while maintaining predictive accuracy.

Dimensionality reduction techniques like PCA can be used, but they do not directly reduce the number of SMONs required

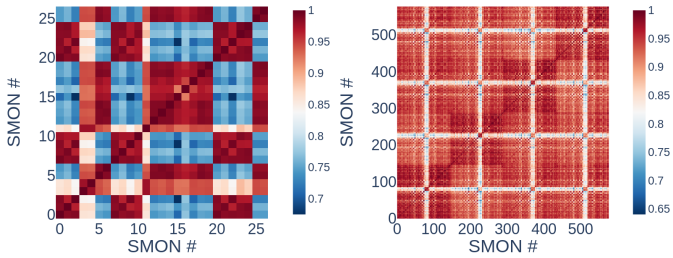

Fig. 2. Spearman Rank Correlation matrices for two products: *A*, with 27 SMONs (left) and *B*, with 579 SMONs (right). In *B*, SMONs are highly correlated. In *A*, clusters of features can be identified.

on board. Therefore, FS is necessary. Given the preference for linear models due to their simplicity and interpretability, CS emerges as an appealing solution. CS offers an efficient approach to feature selection within the framework of linear models. By leveraging the sparsity assumption inherent in real-world datasets, CS identifies and retains only the most informative features, discarding redundant or less relevant ones, making feature selection an intrinsic part of the model training process. This approach simplifies the overall workflow while ensuring that the resulting model remains interpretable and lightweight, which aligns with the domain's preference for linear models.

The CS framework (and the use of regularized linear models) relies on the assumption of a linear relationship between SMONs and $F_{\max}$ (or between features and the target). While this assumption may not hold strictly for all MCU products, preprocessing the SMONs values—such as applying a polynomial transformation—can project the values into an alternate space where linear models can still be effectively applied. This enables the modeling of the relationship between the transformed SMONs values and $F_{\max}$ as linear, enhancing the general applicability of the CS framework for MCU performance screening.

VI. Comparative analysis

It is worth noticing that, as described in Section III, two other alternatives exist to embedded FS (and thus CS): filtering and wrapped methods. But these have some limitations: the first relies only on univariate metrics [6] i.e., taking into account one feature at once and without considering feature interaction. Also, this approach is often unsupervised. The second is often a more accurate method, but it is very time-consuming since it requires training several ML models for different feature sets and choosing the best by cross-validation [39]. Also, it may come with the risk of overfitting, and it is weak in the case of high correlation among features [38]. For the embedded FS, since the feature pruning step is included in the training procedure of models, it requires no additional time. We conducted a comparative analysis, juxtaposing the efficacy of regularized Linear Models (embedded FS and CS) against filtering and wrapper methods. In particular, we considered a filtering procedure grounded in scrutinizing multicollinearity among features. Also, as wrapper methods, we used RFE with a Random Forest as a baseline estimator [40] These methods will be compared analyzing how much they can reduce multicollinearity in the dataset, and secondly, how well they perform in the downstream regression task (i.e., measuring several prediction metrics on a proper test set).

About the ML models, we compared several regularized linear models and their ability to extract relevant features for the regression task. Other types of models have been considered in previous studies [4], [8], [34], as will be discussed in Section VII.

A. Filter Methods

To remove multicollinearity in the dataset, we used a filter method that employed hierarchical clustering based on Ward's linkage [41] of the features' Spearman rank-order correlations [11]. In hierarchical clustering, the goal is to group similar items into clusters defined by some hierarchy, based on some measure of dissimilarity or distance. It begins with each data point as its own cluster and iteratively merges clusters based on a specified distance metric until all data points belong to a single cluster. Ward's method is a linkage criterion that determines how to merge clusters at each step of the hierarchical clustering process [41], based on minimizing the sum of squared differences within all clusters.

The Spearman correlation matrix is first converted to a distance matrix. This was done by computing its complementary to 1 (i.e. the lower the distance, the higher the correlation among features).

A dendrogram can be used to visualize the clustering outcome. Each leaf of the tree represents an individual data point, and branches represent clusters of data points that are more similar to each other. The dendrogram proves particularly beneficial in comprehending the relationships among clusters and determining the appropriate level at which to sever the tree to achieve the desired number of clusters.

By inspecting the corresponding dendrogram, we set a threshold t, to retain at least a SMON/feature from each cluster. The selected subset is then used to train ML models. We will call this technique *CorrT*.

B. Wrapper Methods

We used an RFE with an internal 5-fold cross-validation to determine the optimal number of features. As a baseline estimator, we used a Random Forest [13]. This particular choice has been successfully applied repeatedly in the literature [39], [40], and was used as the baseline in many previous research [1]–[4] about $F_{\max}$ screening. The benefits of using Random Forest are several: this model can deal with the non-linear relation between feature and target, removing the necessity of applying a prior non-linear transformation. Second, this model naturally provides feature importance scores, which can be used for ranking and selection in the RFE process. However, we will avoid using it as the final estimator for several reasons. The first is lack of interpretability: being an ensemble model composed of multiple decision trees, can be challenging to interpret compared to simpler models like linear regression. Secondly, the computational complexity: deploying a large number of decision trees in a Random Forest can be computationally expensive. It may be slower than simpler linear models for real-time applications. Third is memory

usage: Random Forests can consume a significant amount of memory, especially when dealing with many or deep trees. This can be a limitation in memory-constrained environments. Finally, this model tends to overfit data, and may not be suitable in the presence of a limited amount of labeled data (as in dataset *B*). The second and third points make this model unsuitable for eventual deployment on an MCU with limited computation capabilities, eventually enabling on-chip screening. Thus, in the presence of similar prediction accuracy, we prefer using Linear Models because of their superior simplicity.

VII. Experimental Setup

The proposed methodology has been validated on two different datasets coming from two MCUs with different characteristics in terms of ROs, maximum achievable frequency, number of cores, and memory equipment. Dataset *A* is composed of 3588 labeled devices while dataset B is composed of 439 labeled samples.

In product A, each MCU is equipped with 27 SMONs. Devices from product B, instead, presents a significantly larger number of SMONs: 579.

Ten labels are available for both datasets, measuring the $F_{\max}$ for different functional test patterns. The final performance of the devices and thus the target label of our model (the maximum operating frequency) is the artificial label $P_{\min}$ (the minimum among the available labels, for each MCU sample).

For model evaluation, we used 5 train-test splits with a proportion 80%-20%, generated by different random states. Thus, each statistical prediction metric is the mean of 5 values computed on 5 different training/test splits of the dataset, avoiding biased prediction error estimation. Results are presented in terms of normalized Root Mean Square Error (nRMSE), the coefficient of determination score (R^2), Learning Curves, the Area Under the nRMSE Learning Curve (AUC-nRMSE), Guardband (G), and Compression Rate (C). RMSE is popular regression performance index [4], but here normalized by the mean value of $F_{\max}$ in the test set, i.e. nRMSE = RMSE(y_{true}, y_{pred}) / average(y_{true}), to obtain a percentage of the error. R^2 quantifies the proportion of variance in the dependent variable (the target, y) that is explained by the independent variables (the features, x) in the model. An R^2 value of 1 (or 100%) indicates a perfect fit. A constant model that always predicts the mean of the dependent variable would yield an R^2 score of 0. The learning curve plots correlate the training set size with the generalization capabilities of a model. They log on the x-axis the number of samples used to train the model and on the y-axis a measure of prediction performance of the model on the test set (in our case, the nRMSE). The learning curves were created by extracting (for each point x-y) a random sample of the training set of increasing size. Overall, the AUC-nRMSE value indicates the generalization capability of a model: the lower this value, the better the ability of the model to generalize with less labeled samples. We computed AUC-nRMSE values for the first 20 points of the learning curves, focusing on how the models are good at generalizing when only a low number of samples is available (so, about 600 labeled samples for A and about 100 for B).

To account for potential errors and uncertainties in statistical predictions, a risk-based guardband (G) is required. In practice, G effectively raises the pass/fail threshold from the screening frequency f_{screen} to $f_{screen} + G$ to provide an additional margin of safety. By applying this error guardband, manufacturers can ensure that minor variations or uncertainties in the prediction do not affect the product's quality or reliability [4].To minimize the impact on production yield, G should be kept as small as possible. It can be calculated using a test set with true frequencies y, predicted frequencies $\hat{y}$, and errors $e = y - \hat{y}$, defined as:

$$G = \mu_e + k\sigma_e \tag{7}$$

where μ_e and σ_e represent the mean and standard deviation of the error distribution, respectively, and k is a parameter that determines the defect level in parts per million (ppm). For instance, $k = 5.2$ approximates 0.1 ppm, but a stricter value of $k = 6$ is used in this study. G is expressed as a percentage of the actual $F_{\max}$ specification outlined in the datasheet.

The Compression Rate (C) quantifies the reduction in data size achieved by a compression algorithm. It is defined as the ratio of the uncompressed size to the compressed size. In this study, C is calculated by dividing the total number of ROs (or the number of coefficients after Polynomial expansion, if applicable) by the number of non-zero coefficients in the linear model: $C = \frac{\text{totInput}}{\text{Coeff}\neq 0}$.

A higher value of C indicates a smaller number of coefficients, which reduces the model size and the volume of information that needs to be stored for logging purposes.

In previous studies [3], [4], [8], [35], extensive research on dataset A has already been conducted, demonstrating that advanced models such as Neural Networks, including transformer models pre-trained on large-scale unlabeled data, offer only marginal improvements in prediction error over simpler models like the Polynomial Ridge Regressor, with a difference of approximately 0.05% in normalized RMSE (nRMSE). Furthermore, the existing body of work primarily employs models based on Linear Regression or Tree-based methods, with the Polynomial Ridge Regressor consistently identified as providing the best fit for the available data. This evidence supports its selection as a strong baseline for the current study's comparative analysis. Thus, for each model train on *A*, we first transformed the SMONs value by computing a degree-2 polynomial combination of the features.

For dataset B, the larger number of available SMONs allows the relationship between the target and the features to be effectively approximated by a linear fit, eliminating the need for feature transformation without any significant advantage.

We can resume the preprocessing and modeling steps used to enhance the predictive accuracy and robustness of the regression models:

- **Feature Standardization:** All features are initially standardized using a standard scaler to ensure they are on a consistent scale. This step is crucial for the performance

of regularized linear models, as it ensures that all features are comparable, preventing any single feature from disproportionately influencing the model due to differing units or magnitudes [21].

- **Feature Selection (Optional):** For methods involving RFE or CorrT, an optional feature selection step is performed before model training. This step is used to compare the effectiveness of different feature selection techniques.
- **Feature Transformation (only for Dataset A):** For Dataset A, a polynomial feature transformation is applied to address non-linear relationships between SMONs frequency values and F_{max}, enabling the use of linear models even when the relationships between the features and the target variable are not inherently linear.
- **Regression Modeling:** The final step involves applying the predictive regression model, which is constructed using regularized linear models, to predict F_{max}.

Regularized linear models used are described in Sections III-B to III-E. The regularization strength for each model (the α parameter) is chosen using a 5-fold CV, among a grid of 100 values of alphas logarithmically distributed in the range $(0.01, 100)$, finding the hyperparams that maximize the R^2 score.

The experiments were conducted on a server configured with an Intel® Core™ i9-9900K processor, featuring 16 cores running at a base frequency of 3.60 GHz. The server was also equipped with 32 GiB of RAM, allowing for the effective execution of the proposed framework's methods while managing the associated computational loads. A complexity and time analysis was performed in Section VIII-D. Models were trained in *Python* using popular ML libraries; scikit-learn [42] was employed for plain linear regression and with convex regularization (LASSO, Elastic Net, Ridge, OMP). For non-convex penalty models, we utilized scikit-glm [28] package, an extension of scikit-learn for generalized linear models with a custom penalty term.

For the CorrT approach, the corresponding dendrograms for the two MCU products under evaluation can be seen in Fig. 3. To select at least one feature per cluster (see Section VI-A), we choose $t = 0.04$ for product A and $t = 0.3$ for product B. These choices lead to 5 features (over 27) for product A and 10 (over 579) for product B. The low number of features selected in B is due to the high degree of multicollinearity in the dataset (Fig. 2). We recall that this analysis is unsupervised because it relies only on the Spearman Correlation among features. The successive ML models (especially the ones based on L_1 penalty like LASSO) can further stream the features set, selecting the ones the most related to the supervised task. RFE procedure leads to 23 features for *A* and 359 for *B*.

VIII. Experimental Results

In the following section, we present the results of the aforementioned regularization linear models and the a-priori feature selection techniques. We first analyze the different feature sets extracted by the three feature selection strategies under analysis, considering how much these can deal with the multicollinearity in the dataset (Section VIII-A). We then present the empirical analysis in terms of tackling the underlying regression task for both datasets A (Section VIII-B) and B (Section VIII-C).

TABLE I
R^2 Score (%) on datasets *A* and *B* with RandomForest and different feature sets (5-splits)

Method	Dataset A: #Features	Dataset A: R^2 Score	Dataset B: #Features	Dataset B: R^2 Score
All	27	97.94 ± 0.10	579	93.53 ± 1.02
RFE	23	97.92 ± 0.11	359	93.48 ± 1.00
CorrT	5	97.32 ± 0.08	10	93.57 ± 0.85
OMP	11	97.87 ± 0.10	12	93.48 ± 1.19
LASSO	11	97.90 ± 0.12	50	93.98 ± 0.94
ElasticNet	12	97.87 ± 0.13	83	93.70 ± 0.89
MCP	9	97.62 ± 0.08	16	92.86 ± 0.75
LGP	10	97.81 ± 0.13	24	93.27 ± 0.72

A. Multicollinearity analysis

Features sets extracted by CorrT filtering, RFE procedure, and by the main regularized linear models (LASSO, ElasticNet, OMP and LGP and MCP as non-convex regularization), are compared with the whole feature set, considering to what extent each method can deal with multicollinearity if used as a feature selection method. We used a Random forest as a baseline estimator for this task. The model is trained on 5 random train-test splits (80-20%).The performances in terms of R^2 score for the feature sets are presented in Table I. Random Forest performs well on both datasets. The different feature selection methods selected a different number of features, but overall there is no significant difference in performance among the various methods. The scores of all the feature sets taken into account are similar. For both datasets, pruning SMONs does not impact dramatically the model accuracy. However, some methods may be more efficient in reducing the number of features while maintaining comparable performance. For example, the CorrT method selects only 5 features for Dataset *A* with a relatively high R^2 score. For dataset *B*, having a reduced feature set permits slightly decreasing the overfitting of the model, with a little increase in prediction power (see LASSO, ElasticNet).

Due to correlation in the dataset, not all the features are important. This can be highlighted by computing the permutation importance score: this metric serves as a valuable model inspection technique. It involves assessing how shuffling a single feature's values impacts the model accuracy [43]. The permutation importance for the features of the two datasets is presented in Figs. 4 and 5.

These were computed by considering 10 different permutations, for each feature. Only features that produce a drop in the R^2 score greater than 0.1% are shown).

For dataset *A*, apart from couples of SMON with higher impact, all the other SMONs have importance near to zero. Permutation importance analysis indicates that not all the features are deemed important, a result conflicting with the observed high test accuracy. For dataset *B*, both the high number of features and the high correlation among each other

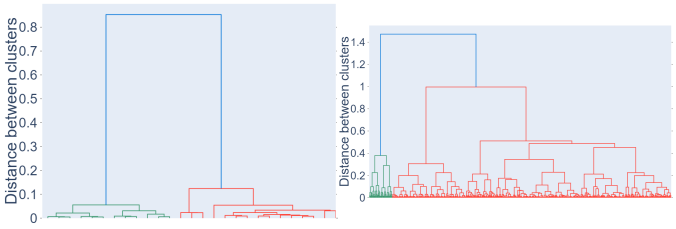

Fig. 3. The Dendograms, results for the hierarchical clustering procedure, for dataset A (upper) and B (lower). By visually inspecting the dendograms, it is possible to set a threshold, to cut it at a certain level, extracting a certain number of cluster. On the y-axis, the distances between clusters. On the x-axis, the SMONs form each (colored) cluster

amplify the effect of low importance for each feature, with an importance of a maximum of 1.5%. Permutation importance values in the negative range suggest that the model performs better on randomly shuffled (or noisy) data compared to the original dataset. This implies that the corresponding feature has minimal influence on predictions. The observed accuracy improvement on shuffled data is likely a result of random chance. This phenomenon is particularly prevalent in smaller datasets (as in B-dataset, where the RandomForest has been trained on about 400 samples).

CorrT technique is the most able to deal with multicollinearity in the dataset since we are forcing removing highly correlated features. The increase in the importance of each feature is evident. For RFE, most of the features have importance near zero: this technique seems not to be able to deal with multicollinearity, with feature importance practically identical to the ones in the original feature set.

Regularized linear models can, to a certain degree, identify variables with greater predictive power. The selected features exhibit slightly elevated permutation importance, indicating their enhanced contribution to predictive accuracy. As an example, in dataset A, the second SMON for importance (SMON 10) goes from 5-6% in the whole and RFE feature set to 10-11% with the regularized linear models. Its importance is practically doubled.

Also for dataset B the importance of each SMON is more than doubled. A particular case is the OMP, which tends to select very few variables, and almost all of them are important.

This analysis shows that

- Prior feature selection steps is not extremely important in increasing prediction accuracy nor in identifying crucial features. With regularized linear models, we can both achieve practically the same prediction accuracies while identifying important features.
- CorrT technique permits a drastic decrease in the number of features, at the cost of a slight decrease in prediction accuracy. But this is achieved also by OMP technique (that selects the feature the most correlated with the residuals) and can be in principle achievable also by other regularized linear models with a higher value for the α parameter.
- RFE technique fails both in both pruning a relevant number of coefficients or increasing accuracy concerning the baseline.

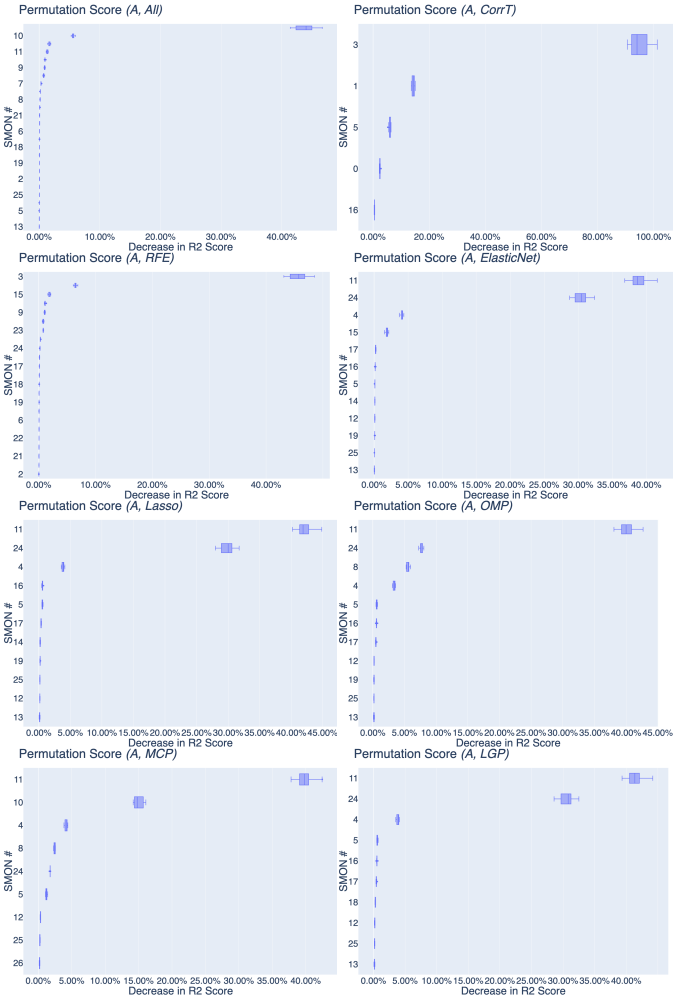

Fig. 4. Permutation Importance Index for dataset A

B. Model metrics: Dataset A

Degree-2 Polynomial transformation applied to each feature sets leads to

- 405 features, from 27 SMONs in the case of no prior FS.
- 299 features from 23 SMONs in the case of RFE.
- 20 features from 5 SMONs in the case of CorrT.

For dataset A, almost all the algorithms converge to the same prediction error (Fig. 6, Table II). What is changing is the number of coefficients of the models.

The regularization permits reducing the overfitting. As a claim of this, the upper plot in Fig. 6 shows the behavior of the Polynomial Linear Regression: as the number of training samples approaches the number of features (about 400), the model perfectly fits them, but is not able to generalize to new unseen samples, with nRMSE on the test sets of about 15%. As the number of samples increases, this behavior is reduced. This behavior does not happen with the regularized model.

OMP usually achieves the highest compression rate: starting from the whole feature set, we can compact information up to about 10:1. with a minimal loss in accuracy (0.01 of R^2 loss). Overall, it is a great save in memory needed to store the datasets and in computations. Among convex regularization,

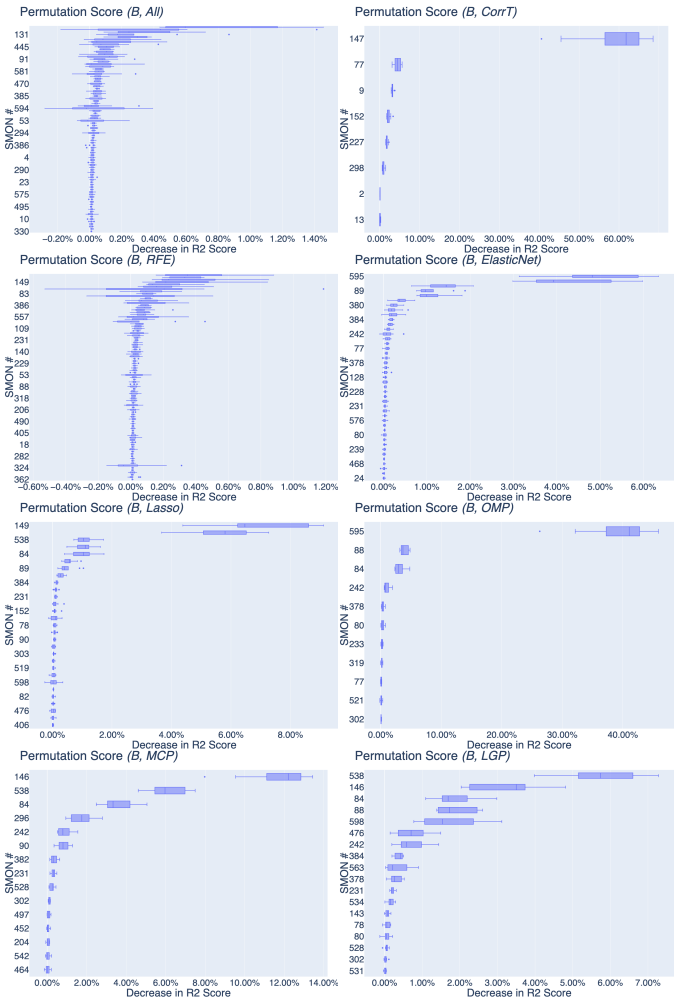

Fig. 5. Permutation Importance Index for dataset B

LASSO is the model with the highest balance in features retained and prediction accuracy.

Regarding non-convex regularization, LGP offers superior performance: the nRMSE is aligned with using all the features (1.66% nRMSE), but with a higher compression rate of about 13:1.

Features from CorrT technique are the ones with the highest prediction errors: removing multicollinearity, we probably removed also informative SMONs. If this accuracy drop would be acceptable, this would make CorrT a good alternative with the highest compression rate (up to about 29:1 for OMP).

For this dataset, RFE procedures can produce very slightly better performances concerning no prior pruning (Ridge with RFE have R^2 score of 97.85%, while LASSO without RFE have an R^2 score of 97.84%). But even in this case, having a further step of FS is beneficial (ElasticNet/LASSO in conjunction with RFE have R^2 of 97.86%, while Ridge 97.85%). However, this comes at the cost of a high computational cost to obtain the starting reduced features set (23 SMONs).

$L_{0.5}$ penalty does not perform well: the behavior heavily depends on the training set, meaning that it is probably overfitting (see the shark-tooth pattern in the learning curve, Fig. 6).

To correlate the degree of generalization capabilities with the training samples, we compute the nRMSE-AUC (Table II). Regularized models are the ones with the lowest AUC, meaning that they require fewer samples to be efficiently trained. In general, simpler models, in terms of parameters or complexity, can often generalize better with fewer samples compared to more complex models. This is a fundamental principle in machine learning known as the *bias-variance tradeoff* [44]. LASSO is the one with the lowest AUC-nRMSE, making it the best in terms of compromise between number of training samples and nRMSE.

TABLE II
ERROR METRICS ON A, 5-FOLDS AVERAGE, DIVIDED BY FEATURE SET (IN BOLD)

Model (per SMONs set)	Selected Coef (over Total)	Selected SMONs (over Total)	C	nRMSE%	R^2%	G%	AUC
All SMONs							
Lin. R.	405/405	27/27	1:1	1.75%	97.58%	11.16%	28.62
Ridge	405/405	27/27	1:1	1.66%	97.84%	10.53%	10.14
LASSO	55/405	26/27	7.36:1	1.66%	97.83%	10.55%	9.98
ElasticNet	76/405	26/27	5.32:1	1.66%	97.84%	10.54%	10.15
OMP	41/405	23/27	9.87:1	1.67%	97.81%	10.61%	10.45
All SMONs (Non convex)							
$\ell_{\frac{1}{2}}$	68/405	26/27	5.95:1	1.68%	97.76%	10.72%	16.88
LGP	31/405	21/27	13.06:1	1.66%	97.83%	10.57%	10.26
MCP	31/405	22/27	13.06:1	1.71%	97.69%	10.90%	10.50
SCAD	29/405	20/27	13.96:1	1.71%	97.67%	10.92%	10.57
CorrT							
Lin. R.	20/405	5/27	20.25:1	1.88%	97.22%	11.96%	11.53
Ridge	20/405	5/27	20.25:1	1.88%	97.22%	11.98%	11.31
LASSO	17/405	5/27	23.82:1	1.88%	97.22%	11.98%	11.32
ElasticNet	19/405	5/27	21.32:1	1.88%	97.22%	12.35%	11.29
OMP	14/405	5/27	28.93:1	1.88%	97.22%	12.33%	11.41
RFE.							
Lin. R.	299/405	23/27	1.35:1	1.71%	97.69%	10.90%	25.60
Ridge	299/405	23/27	1.35:1	1.65%	97.85%	10.50%	10.09
LASSO	67/405	23/27	6.04:1	1.65%	97.86%	10.49%	10.03
ElasticNet	78/405	23/27	5.19:1	1.65%	97.86%	10.49%	10.01
OMP	42/405	22/27	9.64:1	1.66%	97.83%	10.56%	10.44

About SMONs selection: since linear models are highly interpretable, by manually inspecting the coefficients of the models it is possible to identify which SMONs contribute to the selected polynomial coefficient. In other words, we can discard any of the original SMONs if they do not appear among the selected polynomial terms. Coefficients chosen by non-convex regularized linear models permit pruning a certain number of SMONs, without losing in accuracy (LGP, as an example, can prune 6 SMONs from the feature set).

C. Model metrics: Dataset B

For this dataset, no polynomial transformation is used. Thus, the number of coefficients reflects the actual number of SMONs selected. The best overall accuracy is obtained by Ridge Regression without pruning any features (*the more SMONs the higher the information about the device's speed)*. However, we aim to find a tradeoff between prediction accuracy and the number of SMONs selected. Starting from the whole feature set, it is worth noting how LASSO, again, is able to retain informative SMONs from the dataset, and with only a little decrease in the prediction accuracy (nRMSE from 0.86% to 0.89%) can achieve a compression rate of 9.65:1. About non-convex regularization, it is evident how $L_{0.5}$ presents

Fig. 6. Learning curves for different models on A dataset. The x-axes is the number of training samples (log scale) while the y-axes is the nRMSE computed on the test set. The upper plot shows the overfitting of Linear Regression and the Shark-tooth behavior of $\ell_{\frac{1}{2}}$. We disabled that traces in the lower plot. CorrT techniques collapse to 1.94% nRMSE. All the other models are aligned in terms of final error metrics, with slightly better results in terms of error/coefficient for the non-convex LGP model.

again a shark-tooth pattern: also in this case, this model did not behave well. LGP, instead, can further stream the feature set, at the cost of an increase in the prediction accuracy. For CorrT feature set, all the models collapse to about the same performance. R^2 is in the range 94.31% (the best) for OMP, to 94.22% for the non-regularized Linear Regression. Manually removing highly correlated features permits reaching the lowest number of coefficients (only 8 with corrT + OMP), but at the cost of the worst prediction accuracies (that, however, are still under the 1% of nRMSE). RFE method could not prune a relevant amount of features but maintained decent prediction accuracy. In terms of the tradeoff between accuracy and compression rate, LASSO seems overall the best model (about 60 ROs selected, 0.89% nRMSE, both in the case of no and RFE prior feature selection, see Table III). Also, this model permits achieving a good AUC value. Employing a prior feature selection is not beneficial regarding a gain in prediction accuracy. CorrT technique permits finding the lowest possible feature set while maintaining decent prediction. However, this can also be achieved with OMP and no prior feature selection, which also comes with superior prediction performance concerning CorrT. Non-convex regularization performed worse than convex one in terms of prediction accuracy but permitted achieving a higher compression rate.

D. Time Analysis

Generally, Filtering Methods are computationally efficient with a time complexity linear or close to linear concerning the number of features, since they evaluate each feature

TABLE III
ERROR METRICS ON B, 5-FOLDS AVERAGE, DIVIDED BY FEATURE SET (IN BOLD)

Model	Coef No. (over Input)	C	nRMSE%	R^2%	G%	AUC
All SMONs						
Lin. R.	579/579	1:1	1.15%	92.26%	9.29%	0.76
Ridge	579/579	1:1	0.86%	95.64%	6.94%	0.74
LASSO	60/579	9.65:1	0.89%	95.31%	7.17%	0.76
E.Net	82/579	7.06:1	0.92%	94.97%	7.42%	0.75
OMP	18/579	32.16:1	0.93%	94.78%	7.59%	0.89
All SMONs (Non-Convex)						
$\ell_{\frac{1}{2}}$	37/579	15.65:1	1.13%	92.52%	9.14%	1.17
LGP	25/579	23.16:1	0.93%	94.91%	7.51%	0.89
MCP	19/579	30.47:1	0.96%	94.57%	7.69%	0.86
SCAD	14/579	41.36:1	0.97%	94.39%	7.81%	0.87
CorrT						
Lin. R.	10/10	1:1	0.99%	94.22%	7.96%	1.24
Ridge	10/10	1:1	0.98%	94.25%	7.94%	0.80
LASSO	8/10	1.25:1	0.98%	94.29%	7.91%	0.78
Elastic	9/10	1.11:1	0.98%	94.27%	7.92%	0.77
OMP	8/10	1.25:1	0.98%	94.31%	7.90%	0.80
RFE.						
Lin. R.	358/358	1:1	3.72%	17.15%	30.90%	0.76
Ridge	358/358	1:1	0.87%	95.47%	7.05%	0.73
LASSO	75/358	4.77:1	0.89%	95.28%	7.20%	0.77
E.Net	71/358	5.04:1	0.92%	94.97%	7.43%	0.74
OMP	16/358	22.37:1	0.93%	94.83%	7.51%	0.88

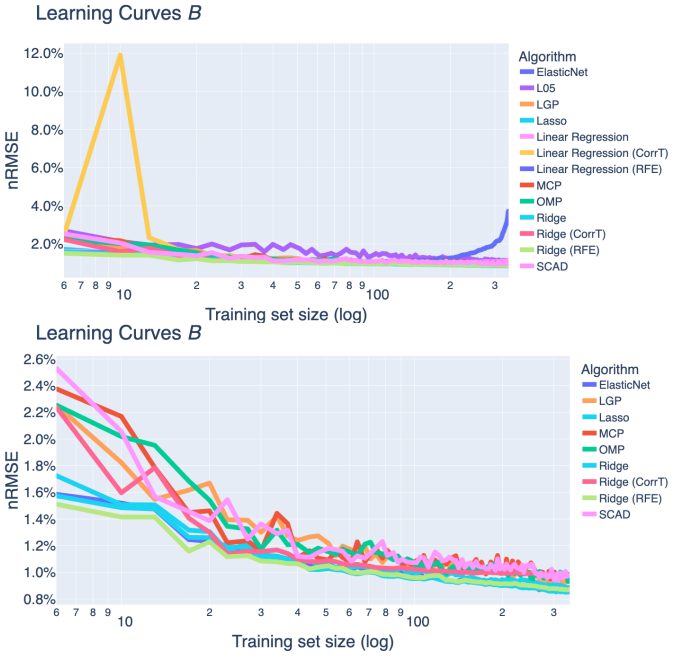

Fig. 7. Learning curves for different models on dataset B. The x-axes is the number of training samples (log scale) while the y-axes is the nRMSE computed on the test set. The upper plot shows, even for this dataset, the overfitting of Linear Regression and the Shark-tooth behavior of $\ell_{\frac{1}{2}}$. We disabled the traces in the lower plot. LASSO model is the most able to reach good balance between number of feature chosen (60) and prediction accuracy (0.89% nRMSE). Non-convex techniques tend to prefer sparser solutions, at the cost of an increase in the prediction error.

independently. In our framework, it took a few seconds to extract the CorrT feature sets.

The time complexity of RFE depends on the choice of the underlying model. At each iteration, it recursively fits the model and evaluates feature importance, resulting in a linear time complexity concerning the feature, that should be multiplied by the training complexity of the underlying model. In our frameworks, it required a few hours to extract RFE feature sets.

CS and regularized linear models integrated the feature selection into the model training process, and their time complexity is determined by the training complexity of the underlying model. These methods do not add additional time to the training procedure. However, training the LASSO or the ElasticNet models is computationally heavier than Ridge or Linear Regression (a few minutes vs a few seconds).

Non-convex models, instead, are slightly faster to train concerning convex ones because of a better underlying optimizer (a few seconds).

For each method, there is a tradeoff between the quality of the feature sets extracted and the time needed to mine it. However, experiments showed how regularized linear models were always able to find a good compromise between prediction accuracy and compression rate. These, in conjunction with the limited time needed to train it, make these models the preferred choice. Reducing the time required for feature selection and SMONs analysis accelerates the time to market by streamlining both development and deployment processes. This increased efficiency enables manufacturers to swiftly implement and capitalize on the framework, leading to faster product launches.

IX. Discussion

While all the regularized linear models presented achieve satisfactory predictive performance in terms of nRMSE, certain models may be more suitable for specific tasks. OMP generally provides the highest sparsification and compression capabilities, albeit with a slight reduction in predictive accuracy. Ridge regression, on the other hand, achieves the best predictive performance without any feature sparsification.

LASSO and LGP offer a balanced compromise between accuracy and feature retention. LASSO tends to retain more features while delivering performance comparable to Ridge regression. In contrast, LGP excels in feature pruning, albeit with a slight reduction in performance.

If the objective is to maximize feature pruning while preserving performance, LGP emerges as a compelling choice. Conversely, if the aim is to maintain predictive performance as close as possible to the Ridge baseline (without pruning), LASSO would be the preferred method.

X. Conclusions

This paper introduced the use of Compressed Sensing and regularized linear models as a novel unifying framework for MCU performance screening. The proposed framework is tailored for direct application during in real-world production, utilizing data derived from the characterization of real MCU products. By presenting a new mathematical approach to the ROs/$F_{\max}$ relationship, a simplified methodology for performance screening is proposed. This perspective, combined with a comparative analysis of three primary methods for feature selection, demonstrates that prior feature selection is unnecessary in this context. Instead, the optimal SMONs for predicting $F_{\max}$ can be directly identified through regularized linear models. This insight is crucial for determining which SMONs should be incorporated into the final product. The framework aims to enhance the efficiency and accuracy of MCU performance testing, particularly in the early production stages with limited labeled samples. The approach facilitates faster and more reliable quality control, reducing production costs and time-to-market. Implementation challenges include ensuring data quality and availability, as the main limitation of MCU performance screening is the limited availability of data in the models' design and training phases. Also, the framework assumes a linear relationship between SMONs frequency values and $F_{\max}$. This assumption holds for dataset B, while for dataset A, it is achieved through a polynomial transformation of the SMON values. Although this broadens the applicability to cases where the feature-target relationship is not strictly linear, extending the approach to other types of non-linear relationships may prove challenging and may require a preliminary feature analysis combined with domain expertise.

The papers shown that wrapper methods are associated with high computational costs, and they not significantly enhance prediction accuracy or reduce the number of features. Filter methods, including the proposed dendrogram technique, can eliminate a substantial number of features; however, due to their reliance solely on feature information, they may also discard relevant features necessary for the subsequent supervised regression, resulting in lower prediction accuracy compared to embedded feature selection methods.

Regularized linear models emerge as an effective balance between prediction accuracy and the retention of SMONs, making them the preferred choice for both datasets *A* and *B* (specifically, Lasso and LGP), regardless of whether the SMONs-$F_{\max}$ relationship is linear or polynomial. OMP achieves the highest compression rate while maintaining acceptable accuracy. Non-convex regularization techniques (particularly LGP) are positioned between LASSO and OMP in terms of feature retention and prediction accuracy.

Future Research Directions may include more diverse datasets representing various real-world scenarios, different types of MCUs, and varying production conditions. This would help validate the robustness and generalizability of the proposed framework. Research could also focus on optimizing the framework for real-time deployment in production environments. This could involve developing more efficient algorithms, reducing model complexity, or leveraging hardware-specific optimizations and edge computing capabilities to improve speed and performance. Also, research could explore the use of other types of non-linear models or more adaptive feature transformation techniques to handle a wider range of features relationships.

REFERENCES

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

[2] S.-P. Mu *et al.*, "Statistical framework and built-in self-speed- binning system for speed binning using on-chip ring oscillators," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2016.

[3] R. Cantoro *et al.*, "Machine Learning based Performance Prediction of Microcontrollers using Speed Monitors," in *IEEE International Test Conference (ITC)*, 2020.

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

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

[7] G. D. Natale *et al.*, *Cross-Layer Reliability of Computing Systems*. Jan. 2020.

[8] N. Bellarmino *et al.*, "Deep learning strategies for labeling and accuracy optimization in microcontroller performance screening," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2024.

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

[10] W. Jia *et al.*, "Feature dimensionality reduction: A review," *Complex & Intelligent Systems*, Jun. 2022.

[11] P. Schober *et al.*, "Correlation coefficients: Appropriate use and interpretation," *Anesthesia I& Analgesia*, 2018.

[12] P. Barbiero *et al.*, "Predictable features elimination: An unsupervised approach to feature selection," in *Machine Learning, Optimization, and Data Science*, G. Nicosia *et al.*, Eds., Cham: Springer International Publishing, 2022.

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

[14] M. Lustig *et al.*, "Sparse mri: The application of compressed sensing for rapid mr imaging," *Magnetic Resonance in Medicine*, 2007.

[15] E. J. Candès *et al.*, "Near-optimal signal recovery from random projections: Universal encoding strategies?" *IEEE Transactions on Information Theory*, 2006.

[16] S. M. Fosson *et al.*, "Sparse linear regression from perturbed data," *Automatica*, 2020.

[17] S. Foucart *et al.*, *A Mathematical Introduction to Compressive Sensing*. New York: Springer, 2013.

[18] J. A. Tropp *et al.*, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Transactions on Information Theory*, 2007.

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

[20] J. Yang *et al.*, "Compressive sensing-enhanced feature selection and its application in travel mode choice prediction," *Applied Soft Computing*, 2019.

[21] T. Hastie *et al.*, "The elements of statistical learning: Data mining, inference, and prediction," *Springer Science and Business Media*, 2009.

[22] G. Gasso *et al.*, "Recovering sparse signals with a certain family of nonconvex penalties and dc programming," *IEEE Transactions on Signal Processing*, 2009.

[23] A. Prater-Bennette *et al.*, *The proximity operator of the log-sum penalty*, 2021.

[24] B. Li *et al.*, *Minimax concave penalty regularized adaptive system identification*, 2023.

[25] H. Xie *et al.*, "SCAD-penalized regression in high-dimensional partially linear models," *The Annals of Statistics*, 2009.

[26] H. Zou *et al.*, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 2005.

[27] R. Rubinstein *et al.*, "Efficient implementation of the k-svd algorithm using batch orthogonal matching pursuit," 2008.

[28] Q. Bertrand *et al.*, "Beyond l1: Faster and better sparse models with skglm," in *NeurIPS*, 2022.

[29] V. Cerone *et al.*, "Fast sparse optimization via adaptive shrinkage," *IFAC-PapersOnLine*, 2023.

[30] S. M. Fosson, "A biconvex analysis for lasso ℓ_1 reweighting," *IEEE Signal Process. Lett.*, 2018.

[31] G. Fracastoro *et al.*, "Playing the lottery with concave regularizers for sparse trainable neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, 2024.

[32] L.-C. Wang, "Experience of data analytics in eda and test—principles, promises, and challenges," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2017.

[33] 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.

[34] N. Bellarmino *et al.*, "Semi-Supervised Deep Learning for Microcontroller Performance Screening," in *IEEE European Test Symposium (ETS)*, 2023.

[35] 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.

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

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

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

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

[40] L. Demarchi *et al.*, "Recursive feature elimination and random forest classification of natura 2000 grasslands in lowland river valleys of poland based on airborne hyperspectral and lidar data fusion," *Remote Sensing*, 2020.

[41] J. H. Ward, "Hierarchical grouping to optimize an objective function," *Journal of the American Statistical Association*, 1963.

[42] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of machine learning research*, 2011.

[43] A. Altmann *et al.*, "Permutation importance: a corrected feature importance measure," *Bioinformatics*, Apr. 2010.

[44] S. Geman *et al.*, "Neural Networks and the Bias/Variance Dilemma," *Neural Computation*, 1992.

Nicolò Bellarmino is a PhD Student in Computer and Control Engineering at Politecnico di Torino. He received his MS degree in Computer Engineering from Politecnico di Torino in 2021. His main research interest are Machine Learning, Data Analysis, and AI systems and their application to real-world cases. He worked in Machine Learning applied to device testing and reliability since 2020. He is part of IEEE-HKN.

Riccardo Cantoro received the MS degree and the PhD in computer engineering from Politecnico di Torino, Italy, in 2013 and 2017, respectively. He is currently a researcher with the Department of Computer Engineering of the same university. His research interests include software-based functional testing of SoCs and memories, and machine learning applied to test and diagnosis. He is a member of the IEEE.

Sophie M. Fosson (Member, IEEE) received the M.Sc. degree in applied mathematics from the Politecnico di Torino, Italy, in 2005 and the Ph.D. degree in mathematics for the industrial technologies from Scuola Normale Superiore di Pisa, Italy, in 2011. From 2012 to 2016, she was a Postdoctoral Associate with the Department of Electronics and Telecommunications, Politecnico di Torino. She visited the Centre Tecnològic de Telecomunicacions de Catalunya, Spain, in 2013, 2014 and 2016. She was researcher at Istituto Superiore Mario Boella, Turin, Italy, in 2017. She is currently an Assistant Professor with the Department of Control and Computer Engineering, Politecnico di Torino. She is Associate Editor for the IEEE Control Systems Letters. Her main research interests include sparse optimization, machine learning, system identification, control and cyber-physical systems.

Martin Huch received the Dipl.Ing. and Dr.Ing. degrees in electrical engineering from the Technical University of Darmstadt (TUD), Germany, in 1986 and 1991. He joined the TriCore design team of Siemens Corporation, Munich in 1997. This was later carved out to become part of Infineon. After some years of SOC design he transitioned to product engineering, where he "baby-sitted" all of Infineon's TriCore products from the first samples until volume production. Focus topics: general analysis methodology, power integrity, and performance validation.

Tobias Kilian received the B.Sc. and M.Sc. degree in electrical engineering and information technology from the Technical University of Munich (TUM), Munich, Germany, in 2017 and 2019, respectively. He is currently pursuing the Ph.D. degree as part of a collaborative project between Infineon Technologies A.G. and the Technical University of Munich. His research focus lies on performance monitors for automotive microcontrollers.

Ulf Schlichtmann (Senior Member, IEEE) received the Dipl-Ing and Dr-Ing degrees in electrical engineering and information technology from the Technical University of Munich (TUM), Munich, Germany, in 1990 and 1995, respectively. He is a professor and the head of the Chair of Electronic Design Automation, TUM. He joined TUM in 2003, following 10 years in industry. His current research interests include computer-aided design of electronic circuits and systems, with an emphasis on designing reliable and robust systems. Increasingly, he focuses on emerging technologies, such as lab-on-chip, and photonics.

Giovanni Squillero (Senior Member, IEEE) received a Ph.D. in Computer Engineering from Politecnico di Torino in 2002; his research mixed computational intelligence and machine learning, with industrial applications that range from electronic CAD to bio-informatics. Currently, Squillero is a professor of Computer Science at Politecnico di Torino, Department of Control and Computer Engineering; he is serving in the technical committee of the IEEE Computational Intelligence Society Games, and in the editorial board of Genetic Programming and Evolvable Machines.