A Multi-Label Active Learning Framework for Microcontroller Performance Screening

A Multi-Label Active Learning Framework for Microcontroller Performance Screening

Nicolò Bellarmino, Riccardo Cantoro, Martin Huch, Tobias Kilian,
Raffaele Martone, Ulf Schlichtmann and Giovanni Squillero

***Abstract*—In safety-critical applications, microcontrollers have to be tested to satisfy strict quality and performances constraints. It has been demonstrated that on-chip ring oscillators can be used as speed monitors to reliably predict the performances. However, any machine-learning model is likely to be inaccurate if trained on an inadequate dataset, and labeling data for training is quite a costly process. In this paper, we present a methodology based on *active learning* to select the best samples to be included in the training set, significantly reducing the time and cost required. Moreover, since different speed measurements are available, we designed a *multi-label* technique to take advantage of their correlations. Experimental results demonstrate that the approach halves the training-set size, with respect to a random-labelling, while it increases the predictive accuracy, with respect to standard single-label machine-learning models.**

***Index Terms*—Performance Screening, Fmax, Speed Monitors, Machine Learning, Active Learning, Device testing**

I. Introduction

Electronic devices in the automotive and aerospace industry require high dependability, and this is especially true for microcontrollers (MCUs) used in safety-critical components. To verify the performance of a device, traditional techniques resort to structural tests like transition delay testing and path delay testing; functional tests; or other indirect monitors. Differently, in performance screening, the maximum working clock frequency $F_{\max}$ of the device is precisely determined in different worst-case conditions; this latter technique is not intended to replace structural testing, yet it is a much more informative process compared to a simple speed binning, where devices are merely sorted into categorical bins (e.g., "fast" and "slow") [1]–[7].

It is common that the oscillation frequencies of appropriate *Ring Oscillators* (RO) can be measured via software during the productive test flow. Since their structures and positions are different, when used as Speed MONitors (SMONs) they are likely to capture the relevant variations in the physical parameters and can be used to predict the eventual performance of the MCUs. Scholars proposed successful models of various complexity, such as [3], [4], [8]–[10].

In our case, the Machine-Learning (ML) model that predicts the performances maps SMONs' frequencies (*features*) to the maximum operating frequencies in a few dozens different test cases (*labels*). For the learning, these labels are acquired by applying functional test patterns that replicate customer use-cases of the MCU. This characterization cannot be done directly in production, but necessitates a separate step during the the engineering phase, and therefore requires significant time and resources [11].

This work demonstrates how Active Learning (AL) techniques can identify the data points that are more interesting and from which a model can generalize better on new unseen data, achieve a 50% reduction in the number of samples required to train an ML model (and thus, a reduction in the time needed to acquire the training set). Furthermore, while all different test cases are required, they are likely to be somewhat related. This work also demonstrate how to increase the accuracy of standard single-target regression models by means of multi-label techniques that take full advantage of correlations between the different labels.

To summarize, the contribution of the paper are the following:

- We propose four new selection strategies that exploit active leaning to reduce the number of samples required to train the model.
- We propose multi-label regressors that take advantage of the unintended, yet inevitable correlations of the functional tests.
- We demonstrate that feature-space reductions and non-linear transformations can reduce the prediction error, allowing the use of simpler and more explainable models.
- We improve the prediction models we previously proposed in [10].

This paper is organized as follows: related works on Machine Learning applied on CAD are presented in Section II, general background information, in Section III. The active-learning metrics and the selection strategies are described in Section IV; the proposed approach is detailed in Section V. The data-collection method is summarized in section Section VI; the experimental evaluation is presented in Section VII. Conclusions are drawn in the last section of the paper.

II. Related Work

The term Machine Learning (ML) was first used by Arthur Samuel in 1959 in a very specific context [12], but since then the term described generic automatic processes whose performances improve through experience [13]. Since 2010s, techniques based on ML have been used in countless practical, industrial applications, and the record of successful stories is

N. Bellarmino, R. Cantoro, R. Martone, and G. Squillero are with Politecnico di Torino (Turin, Italy). M. Huch is with Infineon Technologies AG (Munich, Germany). U. Schlichtmann is with Technical University of Munich (Munich, Germany). T. Kilian is both with Infineon Technologies AG (Munich, Germany) and Technical University of Munich (Munich, Germany).
Authors are listed in alphabetical order.

constantly growing. It is established that it can be applied in the field of Integrated Circuit (IC) testing and, more in general, in Computer-Aided Design (CAD): as design of electronic circuits is a complex process broken down in many steps, ML can be used to achieve circuit's quality, both directly after manufacturing for testing purpose and during its lifetime to increase reliability. The increasing complexity of IC designs naturally motivates the use of these techniques for overcoming scalability issues in tests. Many works in recent years exploited ML techniques at several stages in the production chain. A summary of some works can be found in [14], in which three groups presented their research on interpretable ML in the field of IC test; deep-learning techniques to electrostatic analysis using physics-informed neural networks; modelling degradation phenomena to study the impact at chip level. In [15], some applications of ML in integrated circuit testing are presented. One application is in the so-called "Alternate Test", i.e., the application of appropriate low-cost test stimuli and the extraction of low-cost alternate measurements from which the performances can implicitly be inferred. If alternate measurements highly correlated to the performances are selected, then variations in the performances can be tracked implicitly through the variations in the alternate measurements through an input-output function in the form of a regression model. This paradigm has been proposed as a replacement of the specification tests, with the aim to largely simplify the test process and reduce the test cost [16].

The correlation between structural and functional $F_{\max}$ was investigated in several works. An approach using complex ML algorithms was first presented in [3], while previous works were only considering single parameters [1], [2]. The work identified, among various algorithms applied on a dataset of 60 devices, the *Gaussian Process* [17] as the most effective way to learn from different sets of features (100 flip-flops, 100 transition fault patterns, and 112 testable paths). The work also showed that outliers negatively influence the prediction, and should be identified and removed from the training set; a *conformity check* was implemented for this purpose.

In [4], two lots of the same product, composed respectively of 79 and 74 devices were compared. The difference was in the packages. According to authors, the models trained on one lot were not able to predict devices belonging to the other lot, and this might be caused by the use of different packages or by the limited number of samples in the dataset.

In a more recent work, a different research group correlated the functional $F_{\max}$ with low-cost embedded sensors to accurately measure the slack of a selected number of paths [9]. The method was tested on 300 OpenSPARCT2 SoC devices equipped with 25 sensors, using transition level simulations of two process variation scenarios with both features and labels coming from the wafer sort.

In [10], the values of 27 SMONs coming from wafer sort were correlated to functional $F_{\max}$ measured on more than 4,000 packaged devices extracted from 26 corner-lot wafers; training devices were randomly selected among the available ones; moreover, the models were developed taking into consideration the process variations of the various corner-lots, without considering further dataset shifts during the production.

In [11], the use of active learning techniques in ML applied to the MCU performance screening test was introduced, and three active learning metrics was deployed to build an optimal training set for MCU performance prediction based on SMONs.

In [18], a novelty detection procedure based on wafer-level information derived by SMON measurements on single die was exploited, in order to estimate possible shifts in the data distribution and to decide if re-train or not the underlying performance prediction model.

In [19], an outlier detection technique was developed to identify abnormal samples that lies away from labels' normal distribution, in order to build an higher quality dataset.

III. Background: Testing, MCU and Machine Learning

In this section background information will be given, to better understand the developed framework and the data we managed. In Section III-A, a general view on device testing approaches is given. In Section III-B, we described the technology of the MCU under test and the on-board ring oscillator used as features in the performances screening task. Section III-C briefly describes background information on ML in terms of popular ML models/tools that will be used in this work, including Multi-label approaches and, in particular, the Regression Chain.

A. Device Testing

Tests on chips are mandatory to accomplish datasheet characteristics. Test activities are generally carried out at different stages during the life cycle of an IC. First validation tests are performed during the design-phase of the product, making sure that this can meet the specifications. Characterization tests are performed on the first prototypes of the devices in order to identify failure and fault, possibly leading to a redesign of the product. Once the device has passed the characterization test, large volume production can start: chips are now tested directly on-wafer in the production testing phase before being diced and packaged. Once the chips have been packaged, other tests ensure that this process has been completed in the right way and the IC can be sold on the market. Then, in-the-field tests can be done to ensure that a certain application can be correctly performed. Functional tests are performed to verify that the IC meets the specification, but in the digital microcontroller era, since the functionality has become more complex, verifying the functionality of each part in an exhaustive way is infeasible [20]. Structural tests are based on finding out if there exist some faults in the circuits, rather than verify the functionality of each single part. This approach may need some external test equipment, often costly since this deals with different types of chips with a variable number of pins. To cut cost, test features can be incorporated into the chip. One strategy is to include in the circuit dedicated structures (Design-for-Testability) that make it easier to develop and apply test applications. This approach is a *structural paradigm* because it tries to make sure that the circuit has been manufactured correctly. However,

these additional testing structures needed to be designed and inserted in the circuit, and this may increase the design time and physical space on the die.. When dealing with MCUs or System-on-Chips with at least one microprocessor, suitable test programs can be executed by the processor to test the device itself. Programs are loaded into the processor, exciting one or more parts of the circuit, and the responses are collected for evaluation. This strategy is usually referred to as Software-Based Self-Test (SBST) [21]. The advantages of this solution are multiple: it is non intrusive, it does not require extra hardware and the tests can be performed at the processor operating frequency. This enables the screening of defects that are not detectable at lower speed, avoid overtesting, and can be applied in the field, during the product lifetime. These tests also do not require expensive test equipment.

B. Speed Monitors

The target automotive MCU of this study is manufactured in mature CMOS technology; 26 split-lot wafers are manufactured covering a wide range of technology behavior of the MCU product. Split lots are produced only for engineering purposes, where the tuning of the technology parameters can mimic slow, typical and fast dies. In some split lots, the n–MOSFET has a faster design than the p–MOSFET or vice versa. Nearly all potential technology behavior is covered with this wide variety of split lots, allowing a precise prediction.

The SMONs used in this work are ROs of different structures. The SMONs consist of library cells in series with an overall inverting behavior; thus, the structures oscillate. There are 27 SMONs on the die: the majority is placed in the die center as a compact block; eight identical monitors are distributed across the chip to cover the spatial variations, also known as *global mismatch*. The SMONs on the die can be further divided into five different groups based on their structure, denoted as INV, NAND, NOR, EXT, VM. The INV–RO consists of inverters from the standard library of the design. This symmetric cell type reveals the n– and p–MOSFET behavior in the same way. With the NAND SMON we can create a certain n-biased behavior of the die. The two daisy-chained n–MOSFETs dominate the NAND cell behavior. In the NOR–RO, a p-biased behavior is visible for these gates. The n- and p-biased behavior is further reinforced with the placement of dedicated transistors [22]. More sophisticated SMONs are called EXT; a netlist of the cell design is retrieved, and a particular path is extracted and replicated as RO. Thus, EXTs are replicated paths from the design converted as an RO. These EXT ROs behave similarly to the selected path from the design. The VM–RO is mainly designed and placed for physical parameter variation monitoring across the chip.

All five groups of SMONs are manufactured with two major types of transistors. The two transistor types can be distinguished regarding their threshold voltage: Regular threshold voltage (RegVT) and high threshold voltage (HVT). The RegVTs are fast transistors with high performance but contribute significantly to the leakage current due to the lower threshold voltage. The HVT transistors, on the other hand, have a slow switching behavior (low performance) and significantly reduce the leakage current. Depending on the MC specifications - high performance or low power - the designers and the EDA tools must favor HVT or RegVT transistors. The final design usually consists of a mixture of both transistor types. Some paths in the design consist of transistors with a homogeneous threshold voltage, and some paths have a certain proportion of HVT, and RegVT-this proportion is not fixed and varies widely. Therefore some SMONs are manufactured with homogeneous threshold voltage transistors and some with a mixture of both.

The diverse SMON types, in combination with the various split-lot wafer, give a solid foundation for the investigation of the chip behavior in every corner and enormous benefit for the data processing.

C. Multi-label regression

Multi-Label, also known as Multi-Output or Multi-Target, Regressions are ML problems that involve the prediction of more than one numeric values (labels) for each input data: given an input space $D \in R^n$ and a sample $x = (x_1, x_2, x_3 \dots x_n) \in D$, we want to predict target $y = (y_1, y_2 \dots y_m)$ in the output space $Y \in R^m$, so a multi-label model can be considered as a function that maps R^n to R^m.

The multi-label regression problems can be faced in several ways [23]: one is to transform the multi-label problem into different single-target problems, where only one label is predicted. This resolution method considers each target variable as independent from the others, training m independent models where each model $i \in (1 \dots m)$ takes as input the tuple (x, y_i). However, the targets in many applications are not independent, and independent models do not take advantage of their mutual correlation.

In [24], two approaches have been developed: the Regressor Chain (RC) and the Multi-Target Regressor Stacking (MTRS). The Regressor Chain is a problem-transformation method based on the idea of chaining single-target models, creating a list of regressors called $h = (h_1, h_2 ... h_m)$, in which each of them takes advantage of the predictions made by the predecessors in the list. Chosen an order C in the prediction of the label (assuming $C = (y_1, y_2, ... y_m)$), given the input features X, the first model of the chain is trained on $D_1 = (X, Y_1)$. The successive models are trained on an augmented dataset, including the predicted label ($\hat{Y}$) at the previous step in the feature space: h_2 is trained on $D2 = (X, \hat{Y_1})$, h_3 is trained on $D3 = (X, \hat{Y_1}, \hat{Y_2})$ and so on. Going on in the chain, the models can exploit relationships in the output space. We refer to the corrected version of the Regressor Chain, discussed in [24], that use the predictions of the regression models earlier in the chain instead of the true values of the target variables for fitting the models. These predictions are obtained using an internal k-fold cross-validation. In Fig. 1, it is possible to see the regression chain approach.

IV. Active Learning and Selection Strategies

In many domains annotated data are hard and expensive to obtain due to the cost of the process or the time needed to label the samples. The choice of what samples to label

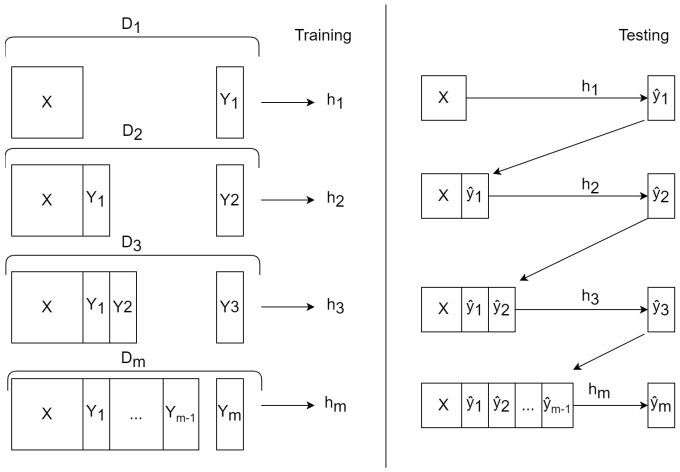

Fig. 1. A visual representation of the Regression Chain.

and insert in the training set of a ML algorithm is often done randomly, since with no assumption or further investigation on the domain of the data there is no reason to choose one sample over another. However, a precise strategy in choosing the data to be inserted in the training set could lead to better models, able to generalize well on new unseen samples, making the training procedure faster and more efficient.

Active learning (AL) aims at easing the data collection process. An AL model automatically decides which instances an annotator should label to train a supervised algorithm in the quickest and most efficient manner. The core idea is to let the algorithm pick examples to be trained on, rather than choosing random points to label and then train the model on those. AL is based on the fact that an ML model trained on a small amount of carefully-chosen data can perform as well as the same model trained with a large random-labeled dataset, if not better [25]. Also, labeling randomly the data may lead to noise or redundancy in the training set and this may slow down the learning process. A model based on Active Learning techniques can possibly reach an higher level of accuracy while using a smaller number of training examples: curious algorithms are better learners in terms of efficiency and knowledge extracted by the data. Also, large part of time of any ML project is spent on getting the correct training dataset. AL techniques can help in the sense that the annotator (that can be a human or a machine) only annotates the most important examples related to the task to be solved, making the whole process more efficient.

It is possible to compare Active Learning to famous sampling techniques as Importance Sampling: also importance sampling can be used to speed-up the convergence of an ML model (such as Deep Neural Network [26]). However, this would require the computation of the Loss Functions, that is the measure of the error made by the ML model on the training samples, and thus, it require the presence of the labels. In this context, instead, we consider only unlabeled sample at the beginning of the procedure, building the training set step-by-step by sampling the most informative samples on the basis of some unsupervised metrics.

There exist different AL framework, divided into in three main categories: Stream-based selective sampling [27], Pool-based sampling [28], and Membership query synthesis [29]–[31]. Pool-based sampling is the most well-known scenario for active learning. This methods enables the algorithm to evaluate the entire dataset before selecting the best samples. The active learner algorithm is often initially trained on a labeled part of the data which is then used to determine which instances would be most beneficial to insert into the training set for the next active learning loop.

Applications such as remote sensing image classification [32], text classification [33], optical networks [34], and certainly ours too, need a representative dataset to learn from. Active Learning performs as well as Domain Adaptation Transfer Learning approaches [34], and can be an alternative to this technique if obtaining samples from other sources is expensive.

Depending on the approach used, a re-training of the actual ML model may be required at each insertion-step: this is the main difference between passive sampling and active sampling approach. In passive sampling, the training set is built from a pool of unlabeled data without relying on learning a function to estimate the "score" value of data, but considering geometric properties of the data itself, selecting a sample only on the basis of its location in the feature space; this approach is computationally efficient because it does not require to re-train a model at each step and it is agnostic on the task that we are facing (classification/regression) or on by the underlying model. Active sampling, instead, relies on a regressor model that is updated at every step, and uses this model to compute a score function to rank the unlabeled data, in order to choose the best sample to label at each step.

The framework chosen in this work is a pool-based sampling strategy with different metrics to rank the samples of the unlabeled dataset: in particular, seven unsupervised metrics are deployed in order to find the distance between the training set and the unlabeled samples. The points for which these metrics present the highest values or score are chosen as points to be labeled. Both active and passive sampling techniques are used. In the following subsections, we will discuss the selection strategies taken into account for our ML framework.

A. Local Outlier factor

Local Outlier Factor (LOF) algorithm inspects the "local" property of the data, taking into account a restricted number of neighbors to indicate the degrees of outlierness for each point in the dataset. The LOF model is trained on "normal data" (i.e., the training set) and tested on new samples. This approach can be used as AL metrics: if the LOF of a point is high, this should be inserted in the training set since it is identified as an outlier.

Local outlier factor is based on "local density", where locality is given by the k-neighbors of a point. The basic concepts that define the LOF of a points are:

- The *K-distance* of a point ($dist_K(x_i)$), so the distance between the point and its K-th nearest neighbor
- The *K-neighbor* of a point x_i, so the cardinality of the set of points that lie inside the circle of radius *K-distance*

- The *Reachability Distance* between two points ($RD(x_i, x_j)$), so the maximum between the *K-distance* of the point (x_j) and the distance between these two points:

$$RD(x_i, x_j) = \max(dist_K(x_j),\ dist(x_i, x_j)) \quad (1)$$

- The Local Reachability Density (LRD), so the inverse of the average reachability distance of a point x_i from its neighbors. Intuitively, the higher is the average *RD* (i.e., the neighbors of a point are far from this), the less is the density of points present around a particular point. This gives information on how far a point is from the nearest cluster. Low values of *LRD* imply that the closest cluster is far from the point.

LOF is the ratio between the average *LRD* of the K neighbors of a point and the LRD of the point itself. If a point x_i is not an outlier, the average LRD of its neighborhood is similar to the LRD of the points x_i and the *LOF* is nearly equal to one. If the point is an outlier, it has a substantially lower density than its neighbor and the *LOF* is greater than one.

The number of neighbors to consider, K, is a parameter to be tuned in the model, in addition to the distances used in density evaluation. In practice, K from 10 to 20 appears to work well in general. This is an active learning approach because it requires to keep the LOF model updated (with a re-training) at each step. In our settings, K is equal to 10 and the distance used is the classical Euclidean distance (i.e., Minkosky distance with $p = 2$).

B. Query by committee

The Query-by-Committee (QBC) method is an effective active learning approach based on having a set of models (called committees) and on a measure of disagreement among the predictions made by the ensemble of models involved. A committee of learners is trained on the available labeled data. Given a pool of unlabeled samples, at each iteration we select one point to label. This point, considered the most informative one, is inserted into the training set. The criterion used to choose the point to be labeled is based on the principle of max-disagreement among the committees: for each unlabeled point, the committees produce a set of predictions and the points on which the committee members have maximal disagreement it is chosen as the one to be labeled. In our setting, the measure of disagreement is the standard deviation of the predictions made by the committee members.

To be an effective approach, the committees have to be heterogeneous: a possible choice is to consider different types of models trained on the whole training set. We chosen models of different natures (linear, non-linear and tree-based models, and in particular Ridge Regressor [35], Support Vector Machine for Regression SVR [36], Random Forest [37], Gaussian Process Regressor [38]) in order to catch different properties of the data. This is an active sampling approach, because the committee have to be retrained at each step.

C. Euclidean Distance

The euclidean distance (EDD) measures how far are the points in the pool of unlabeled samples from the current training set. This is computed in terms of the euclidean distance (L2-norm) between all the unlabeled samples x_i, $i = 1 \dots N$ and the centroid of the actual training set, that is the mean position of all the points in all of the coordinate directions, or the center of mass of the points that form the dataset. In formula, given a training set T and its samples $\tilde{x}$ and a pool of unlabeled samples X, the centroid of TS is

$$\bar{x} = \frac{1}{N} \sum_{\tilde{x} \in T} \tilde{x} \quad (2)$$

and the distance between the training set and the unlabeled samples is:

$$EDD(X, x_n) = ||\bar{x} - x_n||_2 \quad (3)$$

The point to label is the one with the highest EDD. This is a passive sampling approach because there is no underlying model to be trained.

D. Greedy sampling

Following [39], three novel approaches have been used: "Greedy sampling on the input" (GSx), "Greedy Sampling on the output" (GSy) and "improved Greedy Sampling both on input and output" (iGS). In the *GSx* approach, the distance between the training set and an unlabeled sample is computed on the features. Given a starting training set, at each iteration we select a new sample located furthest away from all previously selected samples in the input space to achieve the diversity among the samples. For each of the unlabeled samples, we compute the minimum distance between its features and each of the labeled samples, calling it d_n^x and we choose the one with the maximum d_n^x, so the criterion used is the maximum minimum distance between unlabeled and labeled samples. Assuming that we have already sampled k items, for each of the remaining $N - k$ unlabeled samples x_n, $n = k + 1...N$, we compute the distance between the sample and each of the k labeled samples:

$$d_{nm}^x = ||x_n - x_m||, \quad m = 1...k,\ n = k + 1 \dots N \quad (4)$$

then d_n^x, the shortest distance from x_n to all k labeled samples is computed:

$$d_n^x = \min_m d_{nm}^x, \quad n = k + 1 \dots N \quad (5)$$

and finally we select the sample with the maximum d_n^x to label.

The *GSy* is an active sampling approach that aims to measure the diversity in the output space. Given an initial training set,in each iteration we select a new sample located furthest away from all previously selected samples in the output space. This approach requests to construct a regression model $f(x)$, that also must be tuned (as an hyper-parameter). For each of the unlabeled samples x_i, we compute the minimum distance between the prediction made by the regressor on it, $f(x_i)$, and each of the labels of the training samples, calling it d_n^y and choose the one with the maximum d_n^y, so the criterion used is the maximum minimum distance between prediction of the label of the new sample and the real label of the samples in the current training set. Assuming that we already sampled

and labeled k samples with outputs $y_m,\ m = 1...k$, and a regression model $f(x)$ has been constructed, for each of the remaining $N-k$ unlabeled samples, $x_n,\ n = k+1 \dots N$, we compute first its distance to each of the k outputs:

$$d^y_{nm} = ||f(x_n) - y_m||, \quad m = 1 \dots k, \quad n = k+1 \dots N \quad (6)$$

and d^y_n that is the shortest distance from $f(x_n)$ to $y_m,\ m = 1 \dots k$:

$$d^y_n = \min_m d^y_{nm}, n = k+1 \dots N \quad (7)$$

and then we select the sample with the maximum $d^y_n$to label.

The iGS aims to measure the distance between training and unlabeled samples both in the input and output space, being a mix of GSx and GSy. For each unlabeled sample, we compute the distance on the input and and on the output between it and all the training samples, as indicated above, choosing as score for the sample the minimum between all the product of distance in input and output space. In formula: assuming that k samples have already been labeled with labels $y_n,\ n = 1 \dots k$, for each of the remaining $N-k$ unlabeled samples $x_n,\ n = k+1 \dots N$, iGS computes first d^x_{nm} and d^y_{nm} and then d^{xy}_n as:

$$d^{xy}_n = \min_m d^x_{nm} \cdot d^y_{nm}, \quad n = k+1 \dots N \quad (8)$$

and then selects and label the sample with the maximum $d^x y_n$.

More on GSx, GSy and iGS can be found in [39] While GSx is a passive sampling approach, GSy and iGS are active sampling approach since require the update of a model at each step.

E. Hausdorff distance

The Hausdorff distance (HD) measures how far two subsets of a metric space are from each other. Two sets are close in the Hausdorff distance if every point of either set is close to some point of the other set. In other words, it is the greatest of all the distances from a point in one set to the closest point in the other set. The HD distance has been used in computer vision works to measure the distance between images for generative models [40] or face recognition [41]. In this work, the HD is used as an off-the-shelf distance measure between the training set and the unlabeled samples. This is a passive sampling approach.

V. Proposed Performance Screening Flow

Firsts devices that come out from production lines are grouped by their own production wafer (corner-lots), which have dimension of hundreds of devices. Each new unlabelled wafer is considered as a single point of a dataset, that can be represented by its *centroid*, computed as the mean of all the features of the devices in the wafer. We assume to start with at least one already labelled wafer, otherwise an analysis on how choosing the first wafer to be inserted in the training set is needed. In [39], a methodology for that problem is presented. However, the methodology for the successive step is the same independently of how the first wafer was chosen. (Experiments will be performed by doing a mean on all the possible choices for the initial wafer). For all the remaining unlabelled wafers, an active learning measure is computed to measure the "distance" from the training set at that time. Then, the wafer with the highest utility score is chosen and the labels are acquired. This procedure is repeated until a stop criterion is reached, based on the value of the AL strategy chosen. Since the AL value can be correlated to the prediction error, low values for the AL strategy indicate low prediction error, as shown in Section VII-E. The labelled samples represent the training set for successive machine learning models, that can be trained using one of the multi-label regression techniques presented above. Our experiments concerned the Regression Chain as the main multi-label approach. During the test phase, SMONs values of new wafers are collected and passed to the Machine Learning models in order to predict the maximum operating frequency. For each device, if the predicted operating frequency is below a certain threshold, the device is considered bad and discarded. Eventually, we can compute the AL metrics also in wafers that comes from production lines, and if the score of a particular wafer is above a certain threshold [18] (discussed in Section VII-F), one can decide to label that wafer, insert it in the training set and update the ML model. This approach permits us to:

- Select only the best samples, that contribute the most to fit robust machine learning models. This reduces the size of the training set for our models (i.e. number of devices to be labelled), decreasing the time needed for the characterization process (tens of days of saving).
- Fit an effective machine learning model, that take advantage of the correlation between labels (patterns), improving the performances of the model (in terms of prediction error and production yield).
- Decide to update the model by labelling critical wafers that comes from production.

VI. Data collection

Data acquisition from devices can be separated into two fundamental types: labeled and unlabeled devices. An unlabeled device has only the SMON readouts, while a labeled device has the SMON readout and also goes through a separate characterization with functional patterns. It is always tried to keep the number of labeled devices as small as possible because the characterization process is very complex. Thus there are possible parts in the data collection:(i) The SMON readout of the chip, that lead the *features* and (ii) the measurement of the functional patterns, that provide the *reference labels* for the ML training.

The SMONs are measured in an early production phase when the dies are still on the wafer. The wafer is mounted on a chuck which has high-temperature stability during the measurement process. An internal counter measures the oscillation frequencies of the SMONs. Previous to measurement, each SMON is calibrated to ensure high quality. This calibration is implemented in the software for the measurement process. The stable temperature and calibrated measurement generates high-precision SMON data. The average error is less than 0.15% at the minimum voltage $V_{\min}$. All 27 SMON data are logged for further processing. The SMON readout can

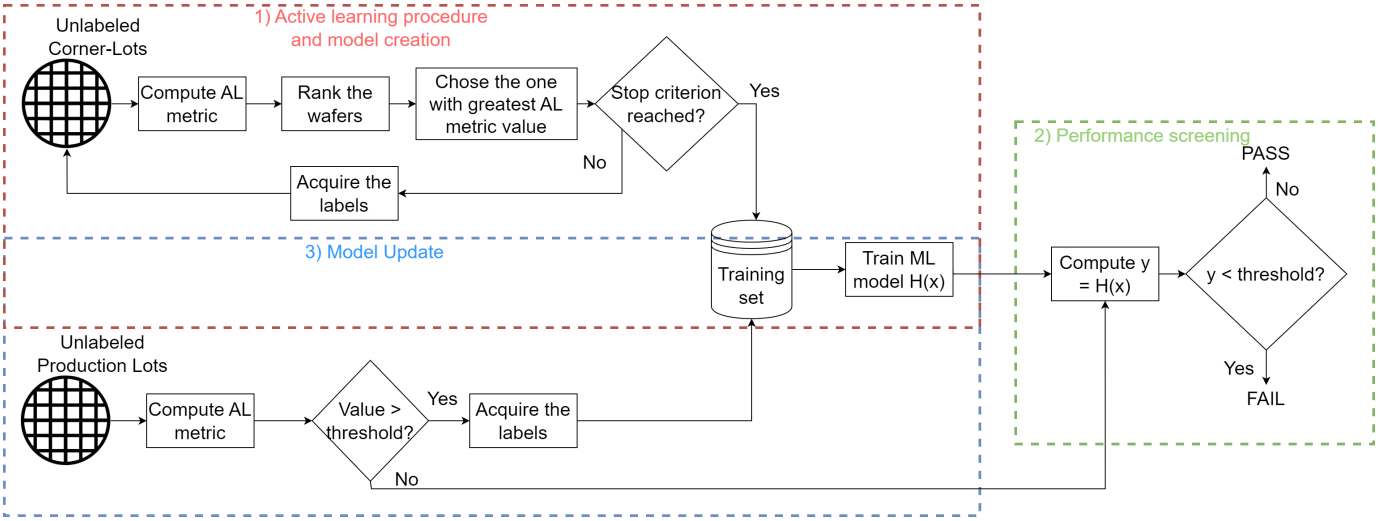

Fig. 2. Proposed performance screening flow

be integrated in the test program of the production test flow. Thus, the SMON data is stored for every produced MCU, possibly leading to millions of unlabeled samples. The smons data are acquired both before and after packaging the dies, to analyze the effect of burn-in on the features. In Fig. 3, the manufacturing procedure and data acquisition steps are represented. Measuring the reference labels is a more elaborate process. The reference labels are measured with packaged die from the wafer. The dies are mounted on a customer application-oriented board. This pseudo-in-system test board has a high accuracy radio frequency (RF) socket with low resistance and the least possible parasitics. Furthermore, the test board has high power integrity and on-board energy-storage capacitance to prevent voltage droop during the testing. During the test, the MCU's temperature and voltage can be precisely controlled and are continuously sampled by on-chip voltage and temperature sensors.

When the voltage and temperature conditions are on the defined steady state values, the MCU is flashed with the firmware. Then a functional pattern is uploaded and started in an endless loop on the MCU. The MCU starts to execute a functional pattern with a low frequency. The frequency is slowly stepped-up in each loop until the functional pattern fails. The process is executed several times (five) to ensure that the failing frequency of the MCU remains at the same value. This failing frequency of the MCU is considered as the critical frequency $F_{\max}$ of the functional pattern and is stored in the database.

The applied functional patterns have a high diversity in stressing various design units of the MCU. The functional patterns are designed that no customer application fails earlier than one out of the functional pattern. Therefore, some patterns are designed to exhaust the maximal load of the MCU. However, there is some $F_{\max}$ uncertainty remaining caused by the diversity of potential customer applications. This uncertainty is why a frequency *guardband* is added to the measured $F_{\max}$. One way to compute this guardband is discussed in Section VII.

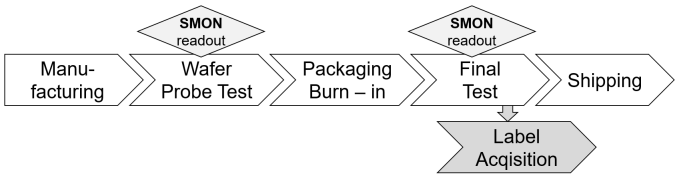

Fig. 3. Data collections steps.

For each MCU, the most critical patterns is the one with the lowest $F_{\max}$ (the one we call $P_{\min}$).

While the measurement of the features is highly accurate, the measurement of the label is more inaccurate:it is performed under worst-case voltage (V_{crit}) and temperature (T_{crit}) conditions at the limits of the specified operating window. To make matters worse, even minor deviations in individual parameters have a negative effect on the measurement result. Even small mechanical vibrations, a change in the base resistance, or statistical noise change the result. Thus, the label measurement is influenced by many unknowns in the acquisition.

To cope with uncertainty in the labelling process, those devices which are more than four standard deviations outlying from the wafer median in a functional test are classified as *outliers*. This outlier detection is applied for every functional test pattern. If an outlier is detected in one functional test, the device is generally marked as an extrinsic device even if all other functional test cases are within the acceptable values. This outlier detection is a very pessimistic approach but necessary for high-quality training data.

VII. Experimental Evaluation

In this section we will discuss about the experiments carried on. First, the so called *advanced model* will be presented. These use feature reduction techniques and polynomial transformation to adapt linear model to our scope, making them able to better fit to our data. Then, the multi-label approach will be discussed: the ten different patterns available $(P1, \ldots, P10)$ will be used at once to predict an

artificial target, $P_{\min}$. Since we want to have a single final performance prediction, we can use the pattern with minimum maximum frequency to be more conservative on the devices performances. Finally, we will demonstrate how the use of active learning can be useful to optimally train the model with a great save in number of samples (and time) needed for building the training set.

Results are presented in terms of famous regression performance indices (mean absolute error MAE, root mean squared error RMSE) but normalized by the mean of $F_{\max}$ in the test set, i.e. $nMAE = MAE(y_{true}, y_{pred})/mean(y_{true})$ and $nRMSE = RMSE(y_{true}, y_{pred})/mean(y_{true})$. We also computed the *Guardband* (G). G [42] is defined as how much increase the amount of minimum product specification to ensure that, even with prediction error, the product will meet the minimum specification. In other words, the guardband is the amount of Hz by which the operating frequency defined in the datasheet has to be increased to ensure that no more than a fixed parts per million (ppm) of false positives will be in production. To make an example, let considering that a machine learning model has been trained and G has been calculated. Supposing that the screening frequency is f_{screen}, the effect of G is to increase the threshold for the pass/fail screening from f_{screen} to $f_{screen} + G$. The goal to achieve is to have G as small as possible, as it affects the production yield. We can compute G on a test set with true frequencies y, predicted frequencies $\hat{y}$ and errors $e = y - \hat{y}$ as:

$$G = \mu_e + k\sigma_e \tag{9}$$

in which μ_e and σ_e are the mean and the standard deviation of the error distribution and k is a parameter that permits to chose the defects level in ppm. $k = 5.2$ is an approximation for 0.1 ppm. In [10], it was discussed how to compute the guardband for production wafers using bootstrapping, rather than for a single test set.

A. Experimental Setup

We used a dataset composed by about 4000 sample with 27 features (the SMONs) and 10 patterns (the labels). A large number of measured samples have missing values in the labels, which is due to measurement errors. These samples were dropped. At the end we have about 2400 full-measurements samples. The training and test sets were created by splitting the data into training and test sets (if not otherwise specified), with proportion 75%-25%, respectively. ML models come from a state-of-the-art data-analysis Python package [43]. Experiments was run into a standard workstation equipped with an Intel® Core™ i9-9900K CPU @ 3.60GHz × 16 and 32 GB of RAM.

B. Single-Target Regression

A first analysis on predicting the operating frequency by deploying machine learning models was done in [10]. There, several base regression algorithms were tested. In this work, we want to improve results already achieved. In particular, a novel method based on applying polynomial transformation and feature reduction is presented. We used both Linear Regression with L1-L2 regularization (Ridge [35], Lasso [44], ElasticNet [45]), and Non-Linear method (kernel-SVR [36], tree-based models [37], [46]) to widely cover the type of models a user can find in common ML tools. Model were trained by means of 5-folds Cross Validation (5-f CV), and Grid-Search was used to find the best hyper-parameters of the models out a a grid of possible ones, that lead to the lowest generalization error.

Previous results show that linear models perform worse than the others, although the difference is not so significant [10]. This suggests that the relation between input and output is not very complex, even though a linear relationship does not perfectly fit for this task. Nonlinear feature transformation can be used to enable the use of linear regression models, by projecting the input feature space into a higher-order space where – with more likelihood – the input-output relationship is linear. The transformation used in this context is the polynomial transformation, that is based on generating polynomial combination of the input features with a certain degree. For example, the degree-2 polynomial features of a 2-dimensional input sample of the form (a,b) is:

$$\Phi(a, b) = (1, a, b, a^2, ab, b^2) \tag{10}$$

In our case, 27 SMONs will generate 406 features with a 2-degree transformation, 4,060 features with a 3-degree transformation and so on. The 2-degree transformation has been chosen. It resulted more effective in terms of prediction error achieved, avoidance of overfitting and number of resulting features. For the feature space reduction analysis, we first computed the Pearson Correlation [47] heatmap. According to this, some SMONs turned out to be strongly correlated. Each single SMON has an almost perfect correlation with 12 to 14 other ones. This suggested to reduce the dimensionality of the feature space to reduce overfitting as much as possible.

We evaluated two approaches for dimensionality reduction: Principal Components Analysis (PCA) and Univariate Feature Selection (UFS). In the first approach, our models extract the first k Principal Components (PCs) and apply the regression on the new feature space, while in the second approach the best k features are selected based on univariate statistical tests. The difference between the two approaches is that PCA creates new features, while UFS simply eliminates them. Results of these analyses are reported in Table I. In the first two columns, the table reports the number of features (k) used and the explained variance (EV) ratio [48], obtained using that number of features with the PCA approach. More than 99% of the variance can be explained using only 3 PCs. The remaining columns report the mean nRMSE value on 10 patterns computed on the test set. We selected 3 well-representative algorithms (i.e., Ridge, SVR, and Extra Trees) for this step.

We added the polynomial transformation of the features extracted by the PCA in the Linear models. Results highlighted that PCA-based approach outperforms UFS using the same k. Moreover, UFS error is inversely proportional to y, while in some cases, using fewer PCs produces better results than the all initial features (e.g., SVR with RBF gave the best results with only 4 PCs, while it starts overfitting when increasing

k). Tree-based models resulted more robust concerning this behaviour. Polynomial regressions using linear models are now aligned with the other results: a comparison is visible in Table II, in which we compare the most recent results with the ones from [10]: for the advanced Ridge model we used the PCA with 14 components and polynomial transformation (obtaining 120 features), while for the SVR we used the PCA with 4 PC and RBF kernel. In the advanced Extra Trees, we used the PCA with 14 PCs. We also tried to include both the Feature Space reduction and the Feature Space transformation in a single step by using the Kernel PCA (KPCA) [49], [50]. KPCA is an extension of PCA which achieves non-linear dimensionality reduction through the use of kernels, or functions that map data into an higher dimensional space. Kernels should enable dealing with more complex data patterns, which would not be visible with only linear transformations. KPCA firsts construct the kernel matrix K from the training dataset by mean of a kernel function k, $K_{i,j} = k(x_i, x_j)$ (for each couple of sample (x_i, x_j)) and then compute the kernel principal components (14, in our case), as happen in PCA. This approaches resulted in higher prediction error with respect to the double-step of feature reduction and feature-transformation (Table IV), but with a final feature-space of dimension 14, rather than 120 (obtained with the application of PCA, extracting the 14 PCs, and then the polynomial transformation of degree 2).

TABLE I
COMPARISON BETWEEN PCA AND UFS (MEAN NRMSE ON 10 PATTERNS, ADVANCED INDEPENDENT MODELS)

#Features	EV ratio	PCA (nRMSE)			UFS (nRMSE)		
		Rigde[1] (poly)	SVR (RBF)	E.Trees	Rigde (poly)	SVR (RBF)	E.Trees[1]
1	87.61%	2.84%	2.57%	2.53%	3.15%	2.93%	2.92%
2	98.90%	1.81%	1.71%	1.70%	3.06%	2.74%	2.75%
3	99.24%	1.62%	1.54%	1.60%	3.02%	2.66%	2.70%
4	99.47%	1.61%	1.54%	1.59%	1.86%	1.84%	1.83%
5	99.52%	1.60%	1.54%	1.59%	1.81%	1.80%	1.81%
6	99.57%	1.60%	1.54%	1.59%	1.79%	1.79%	1.80%
7	99.62%	1.60%	1.54%	1.60%	1.79%	1.78%	1.80%
8	99.67%	1.60%	1.55%	1.60%	1.79%	1.78%	1.79%
9	99.71%	1.60%	1.55%	1.60%	1.75%	1.72%	1.76%
10	99.74%	1.59%	1.54%	1.60%	1.74%	1.71%	1.75%
11	99.77%	1.59%	1.54%	1.59%	1.73%	1.70%	1.74%
12	99.79%	1.58%	1.55%	1.60%	1.71%	1.68%	1.72%
13	99.81%	1.58%	1.55%	1.59%	1.68%	1.65%	1.68%
14	99.83%	1.58%	1.55%	1.59%	1.68%	1.64%	1.67%
15	99.85%	1.58%	1.55%	1.60%	1.68%	1.64%	1.66%
16	99.86%	1.58%	1.55%	1.60%	1.63%	1.58%	1.63%
17	99.88%	1.58%	1.56%	1.60%	1.62%	1.57%	1.60%
18	99.89%	1.58%	1.56%	1.59%	1.62%	1.57%	1.60%
19	99.90%	1.58%	1.56%	1.60%	1.62%	1.57%	1.59%
20	99.92%	1.58%	1.56%	1.60%	1.61%	1.56%	1.59%
21	99.93%	1.58%	1.56%	1.59%	1.60%	1.56%	1.57%
22	99.94%	1.58%	1.56%	1.61%	1.60%	1.56%	1.57%
23	99.96%	1.58%	1.56%	1.61%	1.60%	1.56%	1.58%
24	99.97%	1.58%	1.56%	1.60%	1.60%	1.56%	1.57%
25	99.98%	1.58%	1.56%	1.60%	1.60%	1.56%	1.56%
26	99.99%	1.59%	1.56%	1.61%	1.59%	1.57%	1.56%
27	100.00%	1.59%	1.56%	1.61%	1.59%	1.56%	1.55%

TABLE II
ADVANCED MODELS VS BASE MODELS SCORES (MEAN VALUES ON 10 PATTERNS, INDEPENDENT TARGET)

Algorithm	Base Models			Advanced Models		
	nMAE	nRMSE	G	nMAE	nRMSE	G
Ridge[1]	1.8%	2.4%	15.9%	1.19%	1.58%	10.30%
SVR	1.1%	1.6%	10.4%	1.12%	1.54%	10.22%
Extra Trees[1]	1.2%	1.6%	10.2%	1.15%	1.55%	10.15%

TABLE III
PEARSON CORRELATION BETWEEN LABELS

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
P1	1	0.99	0.99	0.99	0.98	0.99	0.99	0.99	0.99	0.99
P2	0.99	1	0.99	0.99	0.99	0.98	0.99	0.99	0.99	0.99
P3	0.99	0.99	1	0.99	0.99	0.98	0.99	0.99	0.99	0.99
P4	0.99	0.99	0.99	1	0.99	0.98	0.99	0.99	0.99	0.99
P5	0.99	0.99	0.99	0.99	1	0.98	0.99	0.99	0.99	0.99
P6	0.98	0.98	0.98	0.98	0.98	1	0.98	0.98	0.98	0.98
P7	0.99	0.99	0.99	0.99	0.99	0.98	1	0.99	0.99	0.99
P8	0.99	0.99	0.99	0.99	0.99	0.99	0.99	1	1	1
P9	0.99	0.99	0.99	0.99	0.99	0.98	0.99	1	1	1
P10	0.99	0.99	0.99	0.99	0.99	0.99	0.99	1	1	1

C. Multi-Label Regression

The Pearson correlation matrix was computed on the labels (Table III). Its easy to see that all the label are extremely high correlated and thus they cannot be independently considered, justifying the use of multi-label learning strategies. The prediction of one label can be useful to predict the others. This idea is the base of multi-label regression. This approach is based on exploiting the correlation in the label space to predict the minimum maximum operating frequency (i.e. the artificial target $P_{\min}$. The Regressor Chain was applied to several common regression models, in addition to the ones mentioned above. The chained and non-chained algorithms are compared. Each regressor was trained using a Grid Search 5-f CV algorithm to tune the hyper-parameters, and this was done in 20 training-test split in a nested CV fashion, to obtain an unbiased estimation of the generalization error [51]. The inner loop of CV is responsible for model selection and hyperparameter tuning, while the outer loop is used for the generalization error estimation. We performed a paired t-Student test on the 20 results obtained by the outer loop, to reject the hypothesis that one model performed better than another only by chance. Each model was inserted in a pipeline, and pre-processing steps were applied as follow:

- Ridge, Lasso, Elasticnet, OMP: Standard Scaler, PCA with 14 components and polynomial transformation with degree 2 (according to the results already achieved)
- SVR: Standard Scaler, PCA both with 14 and 4 components. RBF kernel was used (according to the results already achieved).
- Decision Tree, Extra Trees, Random Forest: Standard Scaler, PCA with 14 components
- Ridge, Lasso, Elasticnet with KPCA, 14 components and both RBF and Polynomial (degree 2) kernels (namely RK, PK).

- Kernel Ridge with RBF kernel, a Ridge Regression that make use of kernel trick [52]
- Multivariate Adaptive Regression Splines algorithm (MARS) algorithm, both with all the features and only with 14 PCs [53]

The order chosen for the RC was $C = (P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P_{min})$

Results are listed in Table IV: regressor chain performs slightly better than the non-chained models in Ridge, Lasso, OMP, Elasticnet (with a reduction of all the error metrics computed and on G). With Linear model, we can now achieve a G below 10% and nRMSE near to 1.5%. Tree-based regressors can obtain better results in the non-chained version. This can be related to the fact that the ratio $N_{\text{samples}}/N_{\text{features}}$ decrease going on in the chain (at each step we increase by one the features), and when this happen tree-models tends to overfit, loosing in generalization performances. For SVR-4, it seems we do not have real improvement in the chained version. For SVR-14 we cannot say anything: with a t-test on the 20 split used for the estimation of generalization error we obtained a p-value greater than 5% (that is, the control-value we have chosen) and so we are not able to reject the hypothesis that the slight differences in the performances are due to the choice of training and test sets and so happen by chances. Linear Regression models with polynomial-PCA work slightly betters in chain, and this hold also for MARS models.

D. Discussion on ML models

Experiments showed that satisfying performances (about 1.5% of nRMSE) can be achieved with pretty different types of models. However, in this scenario, Linear Models are preferable since they are easily interpretable, because they are based on coefficient assigned to each feature (and thus, they can give diagnostics information on features importance). Even if the kernel methods work good, and the performances are aligned to other type of method, they require to compute the kernel matrix K and store it, as it is needed to transform new data points: this is a quite memory-demanding method, and does not scale well in the case of great-size training set. The use of Multi-Labels regression turned out to be useful to reduce the prediction error on the critical pattern $P_{\min}$ One can choose to use or not Regression Chain or multi-labels approach for performance prediction task on the basis of the final goal: if we are interested in predict a single label (using a single pattern among the ones available), the use of Multi-Labels regression may be unnecessary. But, if highly correlated labels are available, and we aim to have information on all of them plus the critical patterns at once, exploiting the existing correlation, the use of multi-labels regression is useful since it permits to have more accurate models. This mean that we can achieve a lower guardband G, and so increase the production yield, since we discard less working devices.

[1]Similar results can be obtained with other type of linear and tree-based models. Linear models benefits of input transformation while tree based model seems to not

TABLE IV
COMPARISON BETWEEN ADVANCED MODELS IN-CHAIN OR SINGLE TARGET ($P_{\min}$ TARGET)

Algorithm	Single models			Multi-Label (RC) models		
	nMAE	nRMSE	G	nMAE	nRMSE	G
Ridge	1.15%	1.55%	10.20%	**1.12%**	**1.53%**	**10.00%**
SVR (4 PCs)	1.08%	1.51%	9.99%	1.08%	1.51%	9.99%
ExtraTrees	**1.11%**	**1.51%**	**9.91%**	1.12%	1.52%	9.97%
Lasso	1.15%	1.55%	10.20%	**1.11%**	**1.51%**	**9.92%**
ElasticNet	1.15%	1.56%	10.20%	**1.11%**	**1.51%**	**9.93%**
OMP	1.18%	1.59%	10.41%	**1.15%**	**1.55%**	**10.19%**
SVR (14 PCs)	1.08%	1.50%	9.97%	1.08%	1.50%	**9.95%**
Random Forest	**1.12%**	**1.53%**	**10.04%**	1.13%	1.54%	10.07%
Decision Tree	**1.57%**	**2.15%**	**14.09%**	1.60%	2.18%	14.29%
Ridge RK	**1.24%**	**1.67%**	**10.65%**	1.25%	1.68%	10.72%
Lasso RK	**1.24%**	**1.67%**	**10.67%**	1.25%	1.69%	10.77%
ElasticNet RK	**1.24%**	**1.67%**	**10.67%**	1.25%	1.69%	10.77%
Ridge PK	1.27%	1.70%	10.86%	**1.22%**	**1.66%**	**10.58%**
Lasso PK	1.27%	1.71%	10.88%	**1.22%**	**1.66%**	**10.61%**
ElasticNet PK	1.27%	1.71%	10.88%	**1.22%**	**1.66%**	**10.61%**
Kernel Ridge	1.15%	1.56%	9.90%	**1.12%**	**1.52%**	**9.70%**
MARS	1.33%	1.78%	11.34%	**1.30%**	**1.73%**	**11.05%**
MARS (14 PCs)	1.43%	1.89%	12.07%	**1.40%**	**1.85%**	**11.78%**

E. Active Learning

We compared three regression algorithms, namely Ridge regression with polynomial features (PFR), Support Vector Regression (SVR), and Random Forest (RF)) to study the generalization problem in our domain. These models are well representative of the vastness of the models available in general ML tools (linear, non-linear and tree-based models) The AL selection strategies exploited and compared are seven: *LOF*, *QBC*, *EDD*, *GSx*, *GSy*, *iGS*, *HD*. We also used a random sampling and labelling approach where models were trained using a random subset of the available wafers. At each step, the models are tested on a proper test-set. According to the results in Table VI, for the Polynomial Ridge we can obtain good accuracy (below 2% of nRMSE) with 13 wafers using a random approach, 11 using ED, 8 using QBC and GSx, 7 using GSy and iGS and only 6 wafers using LOF and HD. These last two metrics are also the ones with the lowest nRMSE variance (even if also for GSx and iGS the variance is low after 9 wafers). We can obtain similar results with the other algorithms taken into account, summarized in Fig. 4, in which we can clearly see the advantages of the use of AL techniques over a random choice. In the first steps the error is high, but adding wafers it decreases. For all the nRMSE curves, we can calculate the Area Under the Curve. The lower this value is, the better the model is, being able to generalize well in few steps. In the random approach the area under the nRMSE curve value is quite high, but this not hold if a precise strategy is chosen. For all the metrics taken into account, the nRMSE curve goes down faster than a random approach. The nRMSE is halved just after a few steps if a precise strategy is used, and this holds for all the algorithms. Table V shows that LOF and HD are probably the best approach for this kind of problem, resulting in the lowest area under the nRMSE curve.

The error scores obtained at each step of the AL procedure were correlated to the unsupervised metrics and reported

TABLE V
AREA UNDER THE NRMSE CURVE FOR DIFFERENT ALGORITHM (POLYNOMIAL RIDGE, RANDOM FOREST, SVR)

Alg	Rnd	EDD	LOF	QBC	GSx	GSy	iGS	HD
PFR	86.8	57.8	54.2	55.0	54.7	56.1	54.8	**54.1**
RF	93.2	77.8	**62.8**	64.9.0	64.2	66.1	63.1	64.1
SVR	89.3	53.9	52.2	57.0	53.2	55.4	52.8	**52.1**

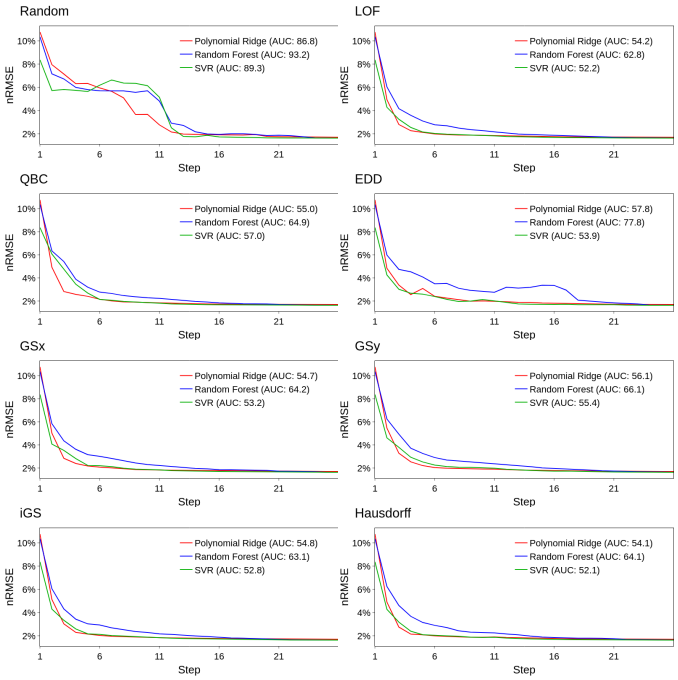

Fig. 4. nRMSE obtained with different regression algorithms, using AL strategies. The curves represent the mean value of the errors depending on all possible initial wafer choices.

in Table VII. Four metrics were used: mutual information, Pearson, Spearman, and Kendall [54] correlation. All of them show a very strong correlation between the nRMSE and the selection strategies score; also in this case, LOF and HD seem to be the best metrics to be used for the AL. A graphical view of the correlation can be observed in Fig. 5 where mean values are reported.

F. Discussion on Active Learning

We can quantify the impact of the AL procedure by measuring the time needed to create the ML training set. We consider the case in which no labelled devices are available at the beginning of the procedure. The labels acquisition for each device require at least 30 min. Each wafer is composed on average by 200 devices. If a random sampling and labelling approach is chosen, to reach an acceptable error (below the 2% of nRMSE) we need 13 wafers, while only 6 wafers are needed with AL (LOF strategy), as indicated above (Fig. 4). This mean that with a random approach the creation of the dataset would last at least $30\text{min} \cdot 13\text{wafers} \cdot 200\text{devices} = 78,000\text{min} \approx 55\text{days}$. If AL is used, the dataset creation requires less than half of the time (≈ 25 days). Also, in a practical case we cannot test the performances of the models at each step of the AL procedure (as done in Fig. 4) since no test set would be available. This hold for random sampling and labelling, but if AL is used, we can have an idea of the performances of the models by looking at the value of the selection strategies on the unlabelled data, since these are highly correlated with the prediction error (Fig. 5, Table VI). Low values for AL strategies should be correlated to low prediction error. The concept of 'low value' depend on the strategy used and it is dataset-dependent. To compute thresholds, it would be necessary to have collected enough data to build a proper test set. In any case, models created with AL procedure are more likely to be of a better quality at each step with respect to random sampling and labelling. AL metrics can also be applied to wafers that comes from production lines. If a particular wafer present a value for an AL metric above a certain threshold (computed on the basis of the value measured during the training-set building), it can be labeled and included in the training set, and the model can be updated. This avoid that the ML model becomes obsolete (and this may happen for many reason, for example a shift in the production processes). A step toward this direction was studied in [18], with focus on LOF metric.

VIII. CONCLUSIONS

This work presented a methodology for predicting MCUs maximum operating frequency. We developed a framework that uses famous existing ML techniques to be deployed in devices performance screening during the production, using active learning techniques and multi-label regression.

The achieved contributions are 1) to have improved existing performance prediction methods by mean of features-transformation approaches and multi-label learning that permitted us to reach lower prediction error and lower guardband, increasing the yield of the process and 2) to have developed a method to use as low as possible labeled device to build the ML models, saving tens of days in the dataset creation

We improved existing performance prediction models [10] with dimensionality-reduction techniques, useful to decrease the number of features that the predictive ML models have to analyze. New features are created, that can compact the information to simplify the models.

The polynomial transformation permits to create non-linear features, that are more suitable for the underlying problem with a reduction in prediction error with respect to baseline methods. These two approaches enable the use of Linear Model in this task, and they are preferable due to their simplicity and explainable nature. Multi-label learning approach turned out to be useful to increase regression performance by analyzing relationship and correlation among target variables in the labels space. We were able to reduce the prediction error of the critical pattern $P_{\min}$. Experiments demonstrated that almost every regression algorithm can benefit of the Regression Chain technique, while for tree-based estimator the performances in chain are not better than the non chained version due to overfitting problems.

The use of AL techniques demonstrated to be suitable for the purpose of reducing the number of labeled point to reach a certain amount of error, because they are able to find an optimal training set. LOF and HD seem to be the best selection

TABLE VI
POLYNOMIAL RIDGE REGRESSION MEAN AND VARIANCE NRMSE DURING ACTIVE LEARNING USING ALL POSSIBLE INITIAL WAFER CHOICES

Step	nRMSE mean [%]								nRMSE variance [%]							
	Rnd	ED	LOF	QBC	GSx	GSy	iGS	HD	Rnd	ED	LOF	QBC	GSx	GSy	iGS	HD
1	10.76	10.76	10.76	10.76	10.35	10.35	10.35	10.35	3.29	3.29	3.29	3.29	4.02	4.02	4.02	4.02
2	7.95	4.84	4.95	4.92	5.83	6.25	6.05	6.25	2.96	2.68	2.71	2.54	1.23	1.63	1.61	1.36
3	7.15	3.39	2.80	2.83	4.36	4.93	4.31	4.61	2.98	0.70	0.57	0.63	1.17	1.65	0.85	0.99
4	6.31	2.56	2.26	2.58	3.62	3.71	3.43	3.68	2.62	0.54	0.23	0.46	0.54	0.9	0.6	0.9
5	6.33	3.09	2.12	2.41	3.16	3.26	3.03	3.15	2.58	0.97	0.16	0.53	0.26	0.74	0.26	0.31
6	5.95	2.41	1.99	2.15	3.01	2.91	2.93	2.9	2.45	0.47	0.16	0.21	0.28	0.31	0.31	0.22
7	5.65	2.26	1.93	2.02	2.83	2.7	2.69	2.71	2.25	0.28	0.09	0.13	0.31	0.27	0.22	0.24
8	5.09	2.11	1.89	1.92	2.63	2.62	2.52	2.43	1.85	0.27	0.06	0.08	0.29	0.41	0.22	0.14
9	3.66	1.99	1.89	1.91	2.44	2.53	2.37	2.32	1.26	0.16	0.07	0.09	0.16	0.44	0.17	0.11
10	3.68	2.01	1.87	1.87	2.3	2.44	2.28	2.27	1.31	0.11	0.06	0.08	0.19	0.35	0.17	0.12
11	2.77	1.99	1.83	1.84	2.22	2.36	2.17	2.25	0.48	0.09	0.05	0.06	0.13	0.32	0.15	0.13
12	2.16	1.94	1.82	1.82	2.15	2.27	2.12	2.16	0.11	0.07	0.06	0.06	0.16	0.28	0.15	0.12
13	1.98	1.89	1.80	1.78	2.07	2.2	2.05	2.09	0.06	0.05	0.05	0.03	0.13	0.21	0.16	0.09
...																
26	1.70	1.70	1.70	1.70	1.70	1.70	1.70	1.70	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

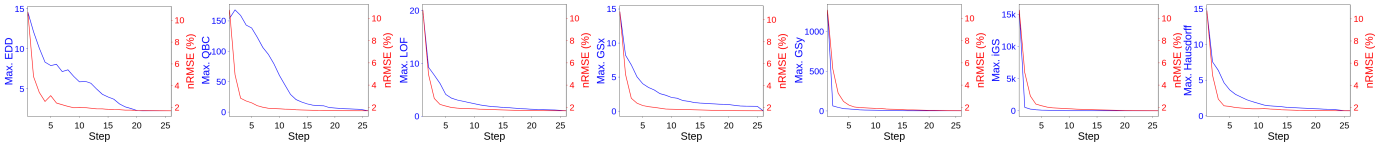

Fig. 5. Correlation between active learning unsupervised scores (EDD, LOF, QBC, GSx, GSy, iGS, HD) and nRMSE using polynomial Ridge regression.

TABLE VII
CORRELATION BETWEEN SELECTION STRATEGIES AND NRMSE ON THE $f_{\max}$ RIDGE REGRESSION MODELS WITH POLYNOMIAL FEATURES

Selection strategy	Mutual information	Pearson correlation	Spearman correlation	Kendall correlation
EDD	0.928	0.708	0.927	0.782
LOF	0.889	0.828	0.967	0.849
QBC	0.898	0.526	0.973	0.854
GSx	0.802	0.839	0.967	0.857
GSy	0.944	0.643	0.822	0.609
iGS	0.935	0.611	0.953	0.817
HD	0.859	0.843	0.968	0.857

strategies to be deployed in production in order to choose the wafer to be labeled, able to halving the training set size with respect to random sampling and labelling approach. This permits to save time in the dataset creation, going from 55 days of continuous labelling in the case of a random approach to 25 days with LOF/HD AL approach. Also, with AL we can have an indication of the generalization error of our models. To conclude, the combined use of AL and multi-label regression permits to build robust ML models faster and better, achieving lower prediction error, reducing the guardband and increasing the yield of the process. In general, the methods presented in this paper can be applied to any situation in which we aim to correlate alternative measure to devices performances by execute testing program on a processor (and thus, create a relation between features and labels by mean of ML algorithms). Multi-label techniques are more suitable with respect to single-label model in every similar scenarios, when several extremely highly-correlated labels are available and the goal is to predict all of them, or, likewise, we aim to predict a final target that derive directly from the available labels (in our case, $P_{\min}$).

REFERENCES

[1] B. D. Cory *et al.*, "Speed binning with path delay test in 150-nm technology," *IEEE Design Test of Computers*, 2003.

[2] J. Zeng *et al.*, "On correlating structural tests with functional tests for speed binning of high performance design," in *2004 ITC*, 2004.

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

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

[5] S. Mu *et al.*, "Statistical Framework and Built-In Self-Speed-Binning System for Speed Binning Using On-Chip Ring Oscillators," *IEEE VLSI*, 2016.

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

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

[8] T. Chan *et al.*, "Synthesis and Analysis of Design-Dependent Ring Oscillator (DDRO) Performance Monitors," *IEEE VLSI*, 2014.

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

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

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

[12] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, 1959.

[13] M. I. Jordan *et al.*, "Machine learning: Trends, perspectives, and prospects," en, *Science*, Jul. 2015.

[14] H. Amrouch *et al.*, "Special session: Machine learning for semiconductor test and reliability," in *2021 IEEE 39th VLSI Test Symposium (VTS)*, 2021.

[15] H.-G. Stratigopoulos, "Machine learning applications in ic testing," in *2018 IEEE 23rd European Test Symposium (ETS)*, 2018.

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

[17] C. E. Rasmussen *et al.*, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[18] F. Angione *et al.*, "Test, reliability and functional safety trends for automotive system-on-chip," in *2022 IEEE European Test Symposium (ETS)*, 2022.

[19] 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.

[20] T. Ruokonen *et al.*, *Fault Detection, Supervision, and Safety for Technical Processes*, ser. SAFEPROCESS'94 : IFAC Symposium, Helsinki University of Technology. International Federation of Automatic Control, 1994.

[21] M. Psarakis *et al.*, "Microprocessor software-based self-testing," *IEEE Design Test of Computers*, 2010.

[22] Y.-J. An *et al.*, "All-digital on-chip process sensor using ratioed inverter-based ring oscillator," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2016.

[23] H. Borchani *et al.*, "A survey on multi-output regression," *WIREs Data Mining and Knowledge Discovery*, 2015.

[24] E. Spyromitros-Xioufis *et al.*, "Multi-label classification methods for multi-target regression," *arXiv*, Nov. 2012.

[25] S. Wang *et al.*, "Pool-based active learning based on incremental decision tree," in *2010 ICMLC*, 2010.

[26] A. Katharopoulos *et al.*, *Not all samples are created equal: Deep learning with importance sampling*, 2018.

[27] D. Cohn *et al.*, "Improving generalization with active learning," *Mach. Learn.*, May 1994.

[28] D. D. Lewis *et al.*, "A sequential algorithm for training text classifiers," *CoRR*, 1994.

[29] J. Zarecki *et al.*, "Textual membership queries," *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, Jul. 2020.

[30] R. Schumann *et al.*, "Active learning via membership query synthesis for semi-supervised sentence classification," in *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, Hong Kong, China: Association for Computational Linguistics, Nov. 2019.

[31] D. Angluin, "Queries and concept learning," *Mach. Learn.*, Apr. 1988.

[32] D. Tuia *et al.*, "A Survey of Active Learning Algorithms for Supervised Remote Sensing Image Classification," *IEEE JSTSP*, 2011.

[33] S. Tong *et al.*, "Support Vector Machine Active Learning With Applications To Text Classification," *JMLR*, Dec. 2001.

[34] D. Azzimonti *et al.*, "Comparison of domain adaptation and active learning techniques for quality of transmission estimation with small-sized training datasets [invited]," *IEEE OSA*, 2021.

[35] R. Rifkin *et al.*, "Notes on regularized least squares," May 2007.

[36] M. Awad *et al.*, "Support vector regression," in *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*. Berkeley, CA: Apress, 2015.

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

[38] C. K. I. Williams *et al.*, "Gaussian processes for regression," in *NIPS*, 1995.

[39] D. Wu *et al.*, *Active learning for regression using greedy sampling*, 2018.

[40] W. Li *et al.*, "Hausdorff gan: Improving gan generation quality with hausdorff metric," *IEEE Transactions on Cybernetics*, 2021.

[41] J. Kim *et al.*, "Recognition of face orientation by divided hausdorff distance," in *2015 38th International Conference on Telecommunications and Signal Processing (TSP)*, 2015.

[42] R. Williams *et al.*, "The effect of guardbands on errors in production testing," in *Proceedings ETC 93 Third European Test Conference*, 1993.

[43] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, 2011.

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

[45] J. Friedman *et al.*, "Regularization paths for generalized linear models via coordinate descent," en, *J. Stat. Softw.*, 2010.

[46] L. Wu *et al.*, *Comparison of neuron-based, kernel-based, tree-based and curve-based machine learning models for predicting daily reference evapotranspiration*, en, P. Pławiak, Ed., May 2019.

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

[48] 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.

[49] Q. Wang, *Kernel principal component analysis and its applications in face recognition and active shape models*, 2012.

[50] B. Schölkopf *et al.*, "Kernel principal component analysis," in *Artificial Neural Networks — ICANN'97*, W. Gerstner *et al.*, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1997.

[51] S. Raschka, *Model evaluation, model selection, and algorithm selection in machine learning*, 2020.

[52] K. Vu *et al.*, *Understanding kernel ridge regression: Common behaviors from simple functions to density functionals*, 2015.

[53] J. H. Friedman, "Multivariate adaptive regression splines," *The Annals of Statistics*, 1991.

[54] A. G. Sefidmazgi *et al.*, "Correlation analysis as a dependency measures for inferring of time-lagged gene regulatory network," in *IKT*, 2016.

Nicolò Bellarmino is a PhD Student in Computer and Control Engineering at Politecnico di Torino. He received the MS degree in Computer Engineering from Politecnico di Torino in 2021. 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.

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. After 5 years of digital circuit design with Bosch, Reutlingen he joined the TriCore design team of Siemens Corporation, Munich in 1997, which 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 very first samples until volume production. Focus topics are general analysis methodology, power integrity, 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.

Raffaele Martone received the MS Degree in Computer Engineering at Politecnico di Torino in 2020. He worked on data science topics, his main activities included data analysis and development of machine learning models.

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 an associate 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.