Experience of Data Analytics in EDA and Test—Principles, Promises, and Challenges

Li-C. Wang

***Abstract*—Applying modern data mining in electronic design automation and test has become an area of growing interest in recent years. This paper reviews some of the recent developments in the area. It begins by introducing several key concepts in machine learning and data mining, followed by a review of different learning approaches. Then, the experience of developing a practical data mining application is described, including promises demonstrated through positive results based on industrial settings and challenges explained in the respective application contexts. Future research directions are summarized at the end.**

***Index Terms*—Data mining, design automation, machine learning, test, verification.**

I. Introduction

MODERN data mining approaches have found many applications in electronic design automation (EDA) and test in recent years. In design and test processes, tremendous amounts of simulation and measurement data are generated and collected. These data present opportunities for applying data mining [1].

Generally speaking, data mining refers to the process of extracting *patterns* from data. Fig. 1 illustrates this simplistic view. The term *pattern* commonly refers to something that has statistical significance or appears frequently. With this view, three key questions can be asked.

1) How to feed the data to a data mining tool?
2) Which data mining tool to use?
3) How the results (patterns) can be utilized?

Today, many software packages are available, providing a comprehensive set of modern data mining tools. An example is the Scikit-learn Python library [2].

Typically, a learning tool expects to see a dataset formatted in a matrix form, as illustrated in Fig. 2. In such a dataset, each *sample* is represented by a vector of values based on a set of *features*. To format a dataset as shown in Fig. 2, one needs to make two decisions: 1) to define what a sample is and 2) to define the set of features. Given raw data, there can be multiple choices for the sample definition, and there can be many choices for features as well. Therefore, many datasets may be constructed to explore those choices.

Given a dataset, the selection of the tool depends on the characteristics of the dataset. For example, when $\vec{y}$ is present

Fig. 1. Three key questions with data mining.

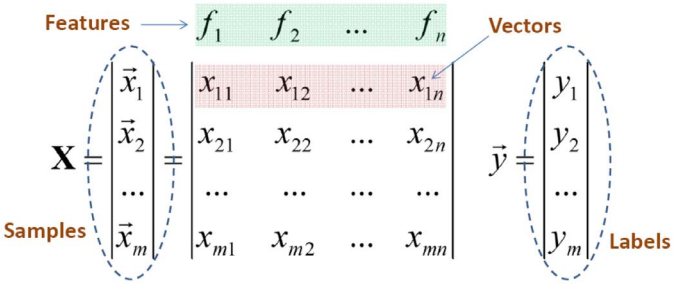

Fig. 2. Dataset seen by a learning tool.

and there is a label for every sample, the dataset is analyzed with a *supervised* learning tool (e.g., a *classification* tool if y is a category label, or a *regression* tool if y is a real number value). When $\vec{y}$ is not present and only $\mathbf{X}$ is, the dataset is analyzed with an *unsupervised* learning tool. When some (usually many fewer) samples have labels and others do not, the analysis can be based on *semisupervised* learning.

In some cases, y can be multivariate. Instead of $\vec{y}$, the right-hand side can be a matrix $\mathbf{Y}$. For example, a partial least square regression tool is designed for regression between two matrices. A canonical correlation tool performs multivariate correlation analysis on a dataset of $\mathbf{X}$ and $\mathbf{Y}$ (see [3]).

The selection of the tool also depends on the utilization of the learning results. For example, if the results are to be interpreted by a person, a tool that produces a simple model is preferred. If the results are used by another tool, then accuracy of the prediction may become the key criterion to select a tool.

In practice, the effectiveness of data mining is not just about the tool. To illustrate this point, Fig. 3 depicts a more realistic view for data mining. In this view, data mining can be seen as an iterative knowledge discovery (KD) process [4].

A KD process begins by a person taking a *perspective*, which can be thought of as a particular way to construct the datasets that are fed into a tool. The tool outputs results (or models) described with some statistical properties. The person examines these properties to determine their meaningfulness, which can mean that the results are interpretable and actionable, or the models are applicable.

The process is iterative because often the person determines that the results are not meaningful. Then, in the next iteration, the person alters the perspective, i.e., changing the way to construct the datasets, and starts a new run.

Manuscript received April 18, 2016; revised September 18, 2016; accepted October 4, 2016. Date of publication October 26, 2016; date of current version May 18, 2017. This paper was recommended by Associate Editor A. E. Gattiker.

The author is with the Department of Electrical and Computer Engineering, University of California at Santa Barbara, Santa Barbara, CA 93106-9560 USA (e-mail: licwang@ece.ucsb.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2016.2621883

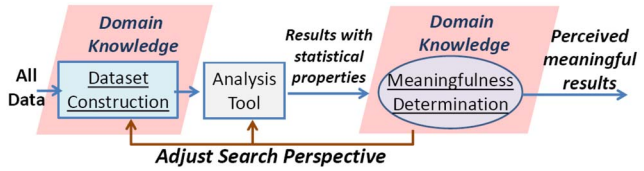

Fig. 3. Iterative KD process.

A KD process essentially is a search process, either searching for interpretable and actionable results or searching for applicable models. In such a process, domain knowledge is involved in the dataset construction and meaningfulness determination steps. And because of this, one can perceive that *learning from data* is a process of using the data to enhance one's domain knowledge. In other words

$$\text{Knowledge} + \text{Data} \Rightarrow \text{Learning}.$$

In theory, learning from data would not be possible without any knowledge [5] to begin with. While this view is not new, it is especially important to emphasize the need for domain knowledge in a KD process because its effectiveness depends on the dataset construction and meaningfulness determination steps, both of which are domain knowledge driven.

Based on the KD view, the objective of this paper is to share our experience from developing various data mining applications in the past. Six application areas are included for discussion.

1) Functional verification.
2) Physical verification.
3) Design-silicon timing correlation.
4) Yield improvement.
5) Customer return modeling.
6) Test cost reduction.

In each of these areas, results based on an actual industrial setting are summarized and discussed.

Applications discussed in this paper are examples, and they are not intended to be a representative set of important applications in EDA and test. Also, it is not author's intent to provide a survey or tutorial of the relevant works. The goal is to utilize these examples to illustrate the principles, promises, and challenges of data mining in EDA and test.

The rest of this paper is organized as follows. Section II discusses several key concepts in machine learning. These concepts serve as a foundation to understand the essence of learning from data. Section III provides a short review of learning approaches commonly employed in an application. Section IV presents the methodologies and results in three presilicon areas. Then, Section V presents the methodologies and results in three post-silicon areas. Section VI discusses a fundamental issue concerning the robustness of data mining. Section VII concludes this paper.

II. Concepts in Machine Learning

This section explains several key machine learning concepts. While the discussion is based on supervised learning, the concepts can also be applicable in unsupervised learning. To begin the discussion, Fig. 4 shows two simple examples, one in a discrete space and the other in a continuous space.

The discrete space is a 3-D space based on binary features x, y, and z. Samples in this space have eight possible feature vectors, 000, 001, . . . , 111. Suppose the data contains seven samples. They are labeled as "green" and "red" as shown in

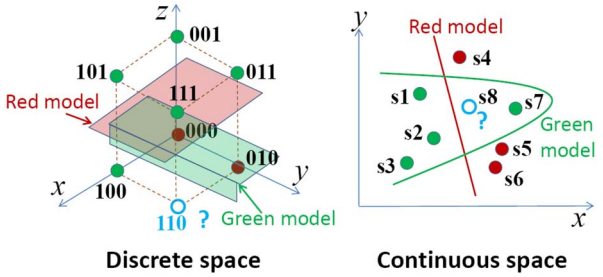

Fig. 4. Examples to illustrate model complexity and noise.

the plot. The one sample whose label is unknown is the feature vector 110. The learning task is to learn from the seven samples and predict the label of the sample 110.

The example shows two possible models, a red model labeling sample 110 as red, and a green model labeling the sample as green. It is important to note that the true label of the sample can be either red or green. Hence, one cannot say which model is better with respect to the truth. What one can say is why one model is chosen over the other.

For example, if one desires to choose a simpler model, the red model will be chosen. One can say that the red model is chosen because it has a lower *model complexity* than the green model. The red model basically classifies the upper four samples as green and the lower four as red. For example, one can specify the model as "If $z > 0.5$, then label green; otherwise label red." To specify the green model, two features are required. For example, the green model can be specified as "If $z > 0.5$ and $x > 0.5$, then label green; otherwise label red." The green model is more complex because it requires one more feature to describe the model.

While the red model is simpler, it ignores that sample 100 is labeled green in the data. This can be viewed as that one believes the data contains *noise*. Hence, the label with 100 is not entirely trustable. Consequently, the simpler red model is chosen by treating the sample 100 as noise. This belief may be altered if multiple samples of feature vector 100 are drawn and they are all labeled green. In that case, one may be forced to abandon the red model and adopt the green one.

The example shown in the continuous space is similar. There are seven samples, $s1, \ldots, s7$ with color labels. Sample $s8$ is to be predicted. The red model predicts its label as red by treating the green label of $s7$ as noise. The green model predicts $s8$'s label as green. The red model has a lower complexity than the green model because the red model is a linear model while the green one is a nonlinear model. Again, the true label of $s8$ is unknown and can be either red or green. One cannot say which model is better with respect to the truth.

The simple examples point out two important considerations for learning a model: 1) *model complexity* and 2) *noise*. It is important to recognize that these considerations are only based on one's assumptions, and in the worst case, a chosen model can still be wrong with respect to a particular prediction regardless how much effort is put into the learning.

A. Setting of the Supervised Learning Problem

More formally, Fig. 5 depicts the setting of the supervised learning problem. The learning problem comprises three components.

1) A generator G that draws samples x from an unknown probability distribution $p(x)$.

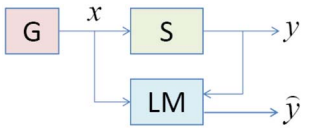

Fig. 5. Setting of the supervised learning problem.

2) A supervisor S that takes x as input and outputs a value y according to a function $F(x)$.
3) A learning machine (LM) capable of implementing a set Q of functions $q_\alpha(x) = \hat{y}$ based on a set of parameters $\alpha \in \Lambda$.

The goal is to learn a function $q()$ based on a set T_m of m *training samples* (x_i, y_i), $1 \leq i \leq m$ such that $q()$ provides as good as possible an approximation to $F(x)$. The approximation is evaluated based on samples not in T_m, i.e., samples with unseen feature vectors.

Without any assumption on G and on $F(x)$, there is no learning (i.e., not *learnable*). For example, one needs to make an assumption stating how the training samples (past) is related to the unseen samples (future). This concerns the generator G. One also needs to make an assumption (or restriction) for the underlying phenomena to be learned. This concerns the function $F()$. In fact, the *no free lunch theorem* for machine learning [6] states that without either assumption, the learning is not possible, or more specifically that there is no one learning algorithm that is better than another in terms of their average performance in predicting for the unseen samples.

Note that the question of *learnable* was studied in the early years within the general scope of inductive inference [7], [8]. For example, *learnability* is characterized *in the limit* where the inductive inference is viewed as an infinite process [7]. One aspect of the research focused on characterizing what families of formal language are learnable under a particular inference scheme, e.g., whether two classes or just one class of samples are provided; whether the learner is allowed to query the supervisor. In contrast, Fig. 5 uses a narrower view of machine learning simply to facilitate the discussion.

B. Computational Learning—Limited Discrete Feature Values

Suppose the learning is based on n features where each feature can have a value of 0 or 1. Suppose the supervised function $F()$ to be learned is a Boolean function. This formulation is called Boolean learning, and was studied with the computational learning theory (CLT) [9], [10].

The CLT captures the notion of *learnable* based on the so-called probable approximately correct (PAC) model. Under PAC, the learning result is *assured* by two parameters: 1) $0 < \delta < (1/2)$: that the learning algorithm with $1 - \delta$ probability will output a desired model (Probably) and 2) $0 < \epsilon < (1/2)$: that the desired model has an error bounded by ϵ (Approximately Correct). Note that we need two parameters because the training samples are randomly drawn and, consequently, the performance of a learning algorithm is randomized.

The PAC framework is commonly used to study if a function is *efficiently learnable* or not. For example, suppose the function structure is $F(x) = P_1 + P_2 + P_3$ where each P is a product term. Then, it can be shown that this structure is not *efficiently PAC-learnable*. More precisely, one can show a polynomial-time reduction to translate an NP-complete language (e.g., the graph three-coloring problem) to the PAC learning three-term disjunctive normal form (DNF) formulas. In other words, learning three-term DNF is computationally hard.

For a function structure that is not efficiently PAC-learnable, learning the function may require to draw an exponential number (with respect to n—the number of features) of samples. Many function structures are not efficiently PAC-learnable. In other words, the power of learning is rather limited. However, this view is based on the accuracy requirement guaranteed by the two parameters δ and ϵ. For practical use, for example, the work in [11] shows that if one only seeks for "good" results without the particular guarantee, learning a Boolean function with a high *percentage* of accuracy may still be achievable.

In the cases that one desires to learn a function that requires an exponential number [i.e., $O(2^n)$] of training samples, one has to limit the n. This means that one has to preselect a small set of important features to enable the learning. In Fig. 3, this preselection would take place in the dataset construction step.

C. Statistical Learning—Continuous Feature Values

When the features take values in a continuous domain, it is a subject studied in statistical learning theory [12]. In statistical learning, it is also often assumed that the outputs are corrupted with some random *noise* ε.

1) Learning Complexity: Instead of capturing learning complexity in terms of the function structure as that in CLT, in statistical learning theory, learning complexity is captured in the concept called Vapnik–Chervonenkis (VC) entropy [12]. Again, suppose the LM is capable of implementing a set Q of functions based on some parameters $\alpha \in \Lambda$. Informally, for a set of m random samples, let $H(Q, m)$ be the subset of functions that *fit* the sample set. VC entropy measures the complexity of $H(Q, n)$, i.e., the entropy of the set of functions based on the given samples. In other words, the VC entropy can be thought of as the complexity seen by the LM on the data of size m.

Because the parameters α can take on any values in a continuous space, the set $H(Q, m)$ is not countable. To measure entropy, one way is to "discretize" the continuous space using a concept call ϵ-net [12]. A ϵ-net basically is a small region of parameter values where the corresponding functions are deemed indistinguishable. With this construct, complexity measurement becomes feasible [12].

The VC entropy is used to study the concept called *learning consistency*. Suppose the LM always picks a learning model that has the smallest possible error rate on the training samples (this is called empirical risk minimization principle). Consistency means that the error rate given by the LM on the training samples represents the true error rate for future samples. Vapnik [12] shows that the necessary and sufficient condition to achieve consistency is $\lim_{m\to\infty}(\text{VCEntropy}(H(Q, m))/m) = 0$. In other words, the complexity seen by the LM on the data grows slower than the data size.

D. Capacity and Falsifiability

The VC Entropy, as $m \to \infty$, can be thought of as a measure of the *capacity* of an LM. One important conclusion from the statistical learning theory is that an LM should not have unlimited capacity, i.e., being able to learn a function of any complexity, which means that the LM can always generate a model to fit the data regardless how big and complex the

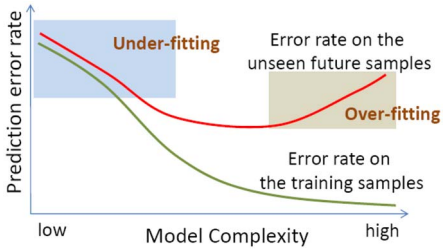

Fig. 6. Under- and over-fitting in view of model complexity.

data is. If this is indeed the case, then the learning becomes *inconsistent*.

Vapnik [12] argues in his book that such an LM is not scientific by applying the principle of falsifiability proposed by Popper [13]. Under this principle, a necessary condition to justify a model to be scientific is the feasibility of its falsification, i.e., the existence of an instance that *cannot* be explained by the model even though the instance falls into its modeling domain. In other words, a scientific LM should not be able to fit all samples all the time.

The consistency concept provides a theoretical view to explain the fundamental ideas in modern LM design. The following explains this view by first explaining the concept of *over-fitting*.

E. Over-Fitting and Cross-Validation

As discussed above, consistency is to ensure that the error rate seen on the training samples (*empirical risk*) is at the same level as the error rate on future unseen samples (*true risk*). To achieve consistency in learning, one cannot have an LM with unlimited capacity. In other words, one has to control the capacity of an LM.

When this control is too tight, an LM is unable to produce a model whose complexity matches the complexity of the data. The result is high training error rate. However, this high error rate is more representative of the true error rate. This situation is called *under-fitting*. Note that under-fitting achieves consistency. However, it is not desirable because of the high error rate.

On the contrary, when the control is too loose, the LM can always produce a model whose complexity matches the complexity of the data. The result is very low training error rate, say almost 0. However, this low error rate is unlikely representative of the true error rate. This situation is called *over-fitting*. Over-fitting represents inconsistency in learning. Fig. 6 illustrates the under- and over-fitting situations in view of the model complexity.

The most common way to check for over-fitting is *cross-validation*. In k-fold cross-validation, the training samples are divided into two subsets, one with $(k-1/k)\%$ of training samples and the other with $(1/k)\%$ of validation samples. This division is repeated k times. The true risk is estimated as the average error rate on the k sets of the validation samples.

While cross-validation is such a common practice, the no free lunch theorem [6] warns about its misuse in practice. Unless one can ensure that the data is somewhat a complete representation for the future unseen samples, cross-validation may provide no practical meaning in an actual application.

F. Learning Algorithm Design—Neural Network Versus SVM

In view of Fig. 6, one desires to implement an LM whose capacity is limited by the model complexity between under- and over-fitting. To estimate under- and over-fitting, cross-validation can be used. With cross-validation, one can then select a learn machine whose capacity is at an optimal tradeoff between under- and over-fitting. Design of a neural network follows this idea.

For example, the capacity of a neural network is controlled by its architecture. A single-layer network (e.g., perceptron) is limited to linear models. A two-layer network can model nonlinear functions. A multilayer deep learning network [14] can model extremely complex nonlinear functions.

An alternative idea is to allow the LM to essentially have unbounded capacity. Then in training, the *simplest* model that fits the dataset based on some noise assumption is selected. In other words, the model complexity is controlled "on-the-fly" rather than in advance with a fixed capacity. Choosing the simplest model is supported by the *Occam's Razor* principle. However, it is also important to note that the fitness of a model depends on the noise assumption. With this principle, the larger the noise is, the lower complexity of the selected model would be.

The support vector machine (SVM) [15] is a popular approach employing the Occam's Razor principle. An SVM learning model is of the form: $M(\vec{x}) = [\sum_{i=1}^{m} \alpha_i k(\vec{x}, \vec{x}_i)] + b$, where $\vec{x}_1, \ldots, \vec{x}_m$ are the training samples. Each $k(\vec{x}, \vec{x}_i)$ measures the similarity between the new input $\vec{x}$ (to be predicted) and the training sample $\vec{x}_i$. Each $\alpha_i \geq 0$ characterizes the importance of the training sample $\vec{x}_i$. In SVM theory, model complexity can be measured as $C = \sum_{i=1}^{m} \alpha_i$. Notice that because the model is based on a collection of samples, its capacity (or complexity) is not fixed in advance.

Let E denote the training error rate. An SVM algorithm tries to minimize the objective of the form $E + \lambda C$, i.e., finding the lowest-complexity model with a minimal training error rate. This approach is called *regularization* and λ is a *regularization* constant [15]. Regularization is not specific to SVM. In many modern learning algorithms, the regularization is applied to alleviate over-fitting [15].

In an SVM model, training samples with a nonzero α value are called support vectors (SVs) because they are the ones actually used by the model. Let l be the number of SVs in an SVM model based on n training samples. The quantity (l/n) can also be seen as a measure of the resulting model complexity. The smaller the quantity is, the more consistent the learning is. For example, this idea of using the quantity (l/n) in practice for building a delay test classifier was studied in [16] before.

III. Learning Approaches

The discussion above focuses on the two contrasting approaches, neural network and SVM, to illustrate the importance of capacity control. In general, a learning algorithm can also be seen as employing one or a combination of four basic ideas or their variations.

1) Nearest neighbor.
2) Model estimation.
3) Density estimation.
4) Bayesian inference.

For example, Fig. 7 depicts a simple classification problem in a 2-D space. The idea for nearest neighbor is that the category of a point (red or blue) can be inferred by the "majority" of data points surrounding it. Then, the trick is in defining the majority (see [17]).

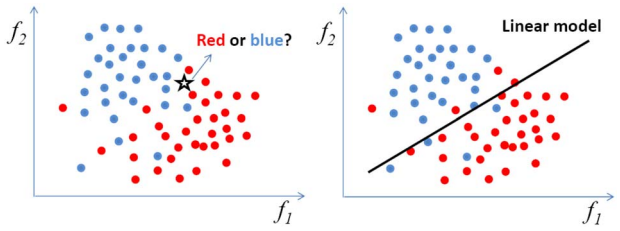

Fig. 7. Nearest neighbor versus model estimation.

In a model-based approach, one begins by assuming a model structure. For example, in binary classification one can assume a linear hyperplane to separate the two classes. A linear hyperplane in an n-dimensional space can be modeled as a linear equation with $n+1$ parameters, for example, $M(f_1, f_2) = w_1 f_1 + w_2 f_2 + b$, where w_1, w_2, and b are parameters to be estimated. An assumed model structure does not have to be equation-based. The model can be a tree [18], a collection of trees [19], or a collection of rules [20].

The third basic idea is to estimate the probability distribution of each class. For example, for each class of data points shown in Fig. 7, one can estimate its probability distribution by assuming it is a 2-D normal distribution, i.e., the red samples with mean μ_1 and covariance Σ_1 as $\mathcal{N}(\mu_1, \Sigma_1)$ and the blue samples as $\mathcal{N}(\mu_2, \Sigma_2)$. Then, for a new sample, the two probability functions are applied and the sample is classified as the class with the higher probability. The popular discriminant analysis [17] follows this idea. Of course, the probability density estimation can be more general than assuming a normal distribution (see [15]).

The fourth idea is following the Bayes' theorem. The Bayes' rule states that

$$\text{Prob}(\text{class}|\vec{x}) = \frac{\text{Prob}(\text{class})\text{Prob}(\vec{x}|\text{class})}{\text{Prob}(\vec{x})} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}.$$

Let $\vec{x} = (x_1, \ldots, x_n)$ be a sample to be predicted. Assume that sample occurrence is uniformly distributed. Then, $\text{Prob}(\vec{x})$ is a constant. In naive Bayes classifiers, it is assumed that all features are mutually independent. Hence, $\text{Prob}(\vec{x}|\text{class}) = \text{Prob}(x_1|\text{class}) \cdots \text{Prob}(x_n|\text{class})$. Each $\text{Prob}(x_i|\text{class})$ can then be estimated using a frequency count on the f_i column of the dataset in Fig. 2.

In practical application, the mutual independence assumption rarely holds. Hence, more sophisticated algorithms are designed to explore the mutual dependence (see [21]).

A. Feature Space and Kernel Method

As mentioned before, a learning tool takes a dataset like Fig. 2 where the space of learning, or *feature space*, is defined by the features. In *kernel*-based learning (see [15], [22]), a learning algorithm (for the most part of its operation) no longer directly accesses the data matrix **X** shown in Fig. 2. This is illustrated in Fig. 8.

In kernel-based learning, a kernel function $k()$ is used to measure the *similarity* between any pair of samples $\vec{x}, \vec{x}'$ as $k(\vec{x}, \vec{x}')$. A kernel function *implicitly* defines the feature space. A kernel-based learning algorithm relies on the *relative information* provided by the kernel function to compute its learning model. Hence, the samples do not have to be represented as vectors as shown in Fig. 2 anymore, as long as a proper kernel function $k()$ can be defined.

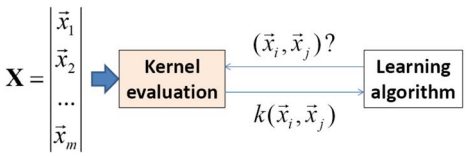

Fig. 8. Kernel function versus learning algorithm.

B. Learning the Feature Space

The definition of the feature space affects the performance of learning. Learning a feature space, either by learning a suitable kernel or by learning a set of features, is not a trivial task. For example, Gaussian process (GP) [23] is an approach that is capable of automatically scaling the input feature values, i.e., weighting their importance. More recently, deep learning network [14] is capable of automatically learning a feature space using the intermediate layers of the network, before building a learning model using the output layer. Such a strategy has shown substantial improvements over previous learning approaches on many applications [14].

C. Types of Learning Algorithms

SVM [15], tree-based algorithms [18], [19], and neural network [14], [17] are popular choices for classification problems. For an unbalanced dataset where there are much more samples from one class than from another, techniques were proposed to rebalance a dataset [24]. However, if the imbalance is quite extreme, rebalancing will not solve the problem and the problem becomes more like a feature search problem [25]–[27] than a traditional classification problem.

Popular regression algorithms include the straightforward nearest neighbor algorithm [17], the least square fit (LSF) [17], the regularized LSF [17], SVM regression [15] and GP [23]. As an example, the work in [28] studied these five types of regression algorithms in the context of learning a model to predict the maximum operating frequency of a commercial processor.

Clustering is among the most-widely used unsupervised learning methods in data mining. Popular algorithms for clustering include K-means, affinity propagation, mean-shift, spectral clustering, hierarchical clustering, DBSCAN, etc. (see [2]). Novelty detection is another widely applied unsupervised learning method. Novelty detection looks for "novel" samples in a dataset. The one-class SVM is a popular choice for novelty detection [15]. The result of clustering or novelty detection can be quite sensitive to the definition of the feature space in which the samples are analyzed.

Principal component analysis (PCA) [29] and independent component analysis (ICA) [30] are popular data transformation methods. For example, PCA can be useful for reducing the dimensionality of a dataset by transforming a high-dimensional **X** matrix into a low-dimensional **X'** matrix. PCA explores correlations among the input features to extract *uncorrelated* new features called *principal components*. ICA is similar to PCA except that instead of looking for uncorrelated components, ICA looks for (statistically) independent components. Both PCA and ICA have found applications in test data analytics [31], [32].

Rule learning is another family of algorithms applicable to a classification problem setting. A rule learning algorithm such as CN2-SD [20] uncovers rules where each rule tries to model a subset of the samples in a given class. For example, the work

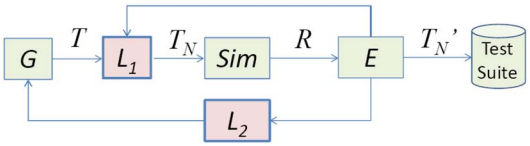

Fig. 9. Two learning components L_1, L_2 that can be added to a simulation-based functional verification flow.

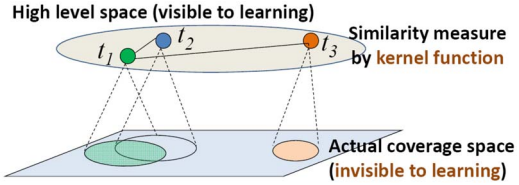

Fig. 10. Main challenge of using a kernel function.

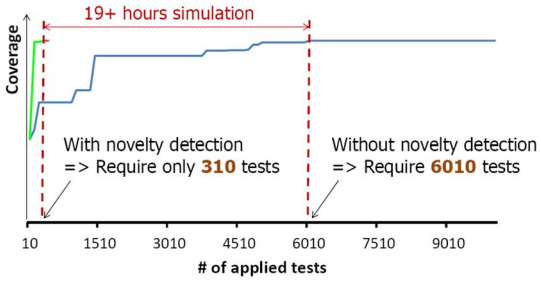

Fig. 11. Example result based on a commercial processor.

in [33] applies classification rule learning to identify features that potentially cause design-silicon timing mismatch.

Association rule mining [34] can be thought of as rule learning in an unsupervised context. In those applications, an algorithm tries to uncover *frequent* patterns (represented as rules) in the dataset. As an example, the work [35] employs the frequency concepts from association rule mining to analyze functional simulation traces.

IV. Presilicon Application Examples

This section includes the discussion of example applications in three areas: 1) functional verification; 2) physical verification; and 3) design-silicon timing correlation. The discussion focuses mostly on functional verification to echo some key concepts discussed before.

A. Functional Verification

Functional verification starts with a verification plan, specifying the aspects of the design to verify [36]. In addition to manually-written direct tests, tests can be generated by constrained random test generation, which is guided by constraints and biases specified in a test bench (or test template). Verification quality is measured by coverage metrics.

Design is an evolutionary process. From one revision to the next, new features are added and/or bugs are fixed. Functional verification evolves accordingly. Assets accumulated through the verification process can include the important tests and test templates. They can be collected into a regression test suite.

Fig. 9 provides an abstract view of functional verification flow with two data mining components, L_1 and L_2 that can be added [37]. The G can be thought of as a constrained random test generator that produces the test set T. The simulator Sim simulates the tests and produces the result R that is evaluated in the step E. The evaluation can be based on coverage or bug excitation. The set of the important tests (or test templates), denoted as T_N', is selected into the test suite.

The first component L_1 can be called a *filtering component*. Its goal is to filter out unimportant tests *before* the simulation and in turn, to save simulation cost.

The second component L_2 is called a *feedback component*. Its goal is to feedback knowledge learned from the result R for improving the generator, for example to adjust the constraints to better target a coverage point. While the focus of L_1 is on the efficiency, the focus of L_2 is on the quality of the tests.

1) Test Filtering Component: The early work in [38] proposed using novelty detection to implement the filtering component. A novelty detector can be learned using the unsupervised learning version of the SVM, the one-class algorithm [15]. Let $T_{\text{data}} = \{t_1, \ldots, t_n\}$ be the tests that have been simulated. The learning is based on T_{data} to build a novelty detector D_n. Then, in application, the importance of a new test t is predicted by $D_n(t)$. If t is predicted non-novel, then the test is filtered out, i.e., not important.

In functional verification of a processor or SoC, a test is a sequence of instructions, i.e., an assembly program. As mentioned before with Fig. 2, a learning tool takes a dataset in matrix form. Hence, one has to bridge this gap, for example, by using the kernel method discussed in Section III-A.

For the learning to be meaningful, a kernel $k()$ has to reflect what is really happening in the actual coverage space of interest. For example, in Fig. 10 suppose we have a kernel $k()$ calculating a similarity between two programs as a value in $[0, 1]$ where 0 means most dissimilar. Then, in the example, one should have $k(t_1, t_2) = z$ for some $z \in [0, 1]$ while have $k(t_1, t_3) = 0$ and $k(t_2, t_3) = 0$. This is because t_1 and t_2 overlaps in the coverage space and they do not overlap with t_3. In other words, the kernel somehow has to predict how two tests overlap in the coverage space of interest, before they are simulated and their actual coverages are known.

For the filtering component to be effective in application, however, the kernel does not have to be close to 100% accurate [39]. In fact, requiring a 100% accurate kernel would not be feasible, since learning a kernel accurately capturing the similarity relationships between tests in terms of their actual coverages can be quite challenging [41].

A recent work in [40] proposes implementing a kernel function based on a fast estimation of the coverage. The idea is to prebuild a coverage database using single-instruction and short multi-instruction samples. The coverage of an actual instruction sequence is then estimated by accessing the information from the database (without simulation). The kernel calculates the similarity between two programs using their estimated coverage on a set of coverage points. This set, however, is dynamically adjusted to remove those points that have been frequently covered during the actual simulation.

The approach was applied to a dual-thread low-power processor [40]. Fig. 11 shows a typical result. The application flow is iterative. For example, a large set of N tests are generated in total. Initially, ten random tests are selected to be simulated and a novelty detector is learned based on these ten tests. The detector is used to select the next 30 novel tests to simulate. Then, a novelty detector is learned based on the total 40 tests. This process repeats until a desired coverage is achieved or it runs out of tests.

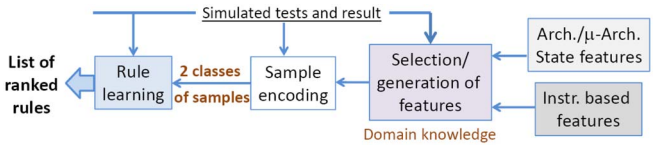

Fig. 12. Implementing the feedback component.

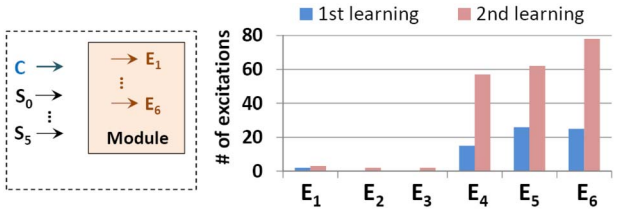

Fig. 13. Example of applying the feedback component.

In Fig. 11, each test is a 50-instruction program generated for verifying a complex fixed-point unit. With the novelty detection, only 310 tests are required to achieve the desired coverage, compared to 6160 tests without novelty detection. Hence, the saving is about 95% which translates into 19 h of simulation time using the server farm (or one week if using a single server). Overall, the approach can achieve 80%–96% saving on various scenarios for verifying the processor [40].

2) Test Feedback Component: Fig. 12 depicts the basic ideas for implementing a feedback component discussed in [42] and [43]. To learn from simulation results, i.e., traces, one needs to have the results divided into two classes of samples. For example, one class comprises those traces hitting a coverage point or exercising a bug event (positive) and the other comprises the rest (negative). Then, the next challenge is how to encode those traces.

Fig. 12 shows that a trace is encoded with *features* which can be, for example, state variables from architecture and/or micro-architecture specifications. The work in [42] also discusses features based on the instructions. With each encoding scheme, a trace is represented as a sequence of vectors where each vector tells whether the features are activated in the respective cycle or not.

The challenge is that if we treat each trace as a sample, each sample is represented with a matrix whose dimensions are defined by the number of features and number of cycles. The dataset is 3-D (because of the time aspect) rather than 2-D as shown in Fig. 2. In association rule mining, one way to consider the time aspect in mining is *sequence mining* and a key idea is to employ *sliding windows* [44], [45] to restrict the time aspect.

Sliding windows convert a 3-D dataset into 2-D [42]. Then, *rule learning* can be applied to extract rules that best differentiate the positive samples from the negative samples. As mentioned before in Section III-C, rule learning can be supervised [20] or unsupervised [34].

Because we have two classes of samples, the problem is supervised. However, one issue is that we usually have many fewer positive samples than negative samples, because in most cases, we are interested in understanding how to activate a rare event. Conceptually, this issue can be resolved in two steps [43]. The first is to construct a set of ranked hypotheses (candidate rules) using the positive samples. Then, the negative samples are used to filter those candidates. The work in [43] discusses that the quality of learning can very much depend on the number of positive samples. Hence, even though the simulation data is abundant, the quality of the data is reflected mostly in terms of the positive samples. In other words, more data does not always imply easier learning.

The feedback component has one major limitation. If one is interested in hitting an event that is never covered, then there is no data to learn from. Fig. 13 shows an example to explain how the learning could be applied when the initial simulation had zero coverage on the events of interest [43]. In this example, the events to cover were E_1–E_6. Simulation of more than 30K tests generated from a given test template could not cover any of them.

For this example, domain knowledge was applied to recognize that signal values on C and S_0–S_5 were highly related to the activation of the six E's events. Hence, learning was used to learn about the properties related to C and S_0–S_5. Fig. 13 shows that after one iteration of learning, the modified test template could cover four out of the six events (with 1K new tests). By learning on the new positive tests, the modified test template could cover all six events with 100 tests.

3) Discussion: The filtering and feedback components illustrate the two typical application scenarios, one where the results are used by another tool (simulation) and the other where the results are interpreted by a person (for modifying the test template). In both scenarios, domain knowledge is essential for the learning. In the filtering component, domain knowledge is applied in the kernel design. In the feedback component, domain knowledge is reflected in the initial set of features and also in the selection of relevant signals when the events of interest have zero coverage to begin with. During the learning, the important signals used in the kernel and the important features captured in the rules become part of the learned knowledge in the KD process.

4) Other Pointers: The filtering and feedback components are examples of applying data mining in functional verification. There are other important applications in verification and validation. For example, the team at IBM Haifa pioneers many works for improving functional test generation (see [36], [46]). Another important area is assertion generation. Recent results can be found in [47] and [48]. The work in [49] studies of the question of inferring design specifications from design behaviors with many theoretical insights. Another important area is debug. For example, the work in [50] pioneers the use of machine learning techniques to overcome challenges in post-silicon debug. System performance modeling is another important application area. Recent results can be found in [51] and [52], where the work in [53] also emphasizes the use of domain knowledge. Modeling thermal behavior and analog behavior are also important applications (see [54], [55]).

B. Physical Verification—Layout Hot Spot Detection

Layout hot-spot detection is another area where machine learning finds applications [56]. Traditionally, hot spots are detected through lithography simulation which is very slow or a pattern matching based tool which is fast but hard to maintain accuracy. The idea of applying machine learning is to achieve extra-fast hot-spot detection with improved accuracy over the pattern matching tool [57].

Fig. 14 depicts the setup proposed in an early work [58] where lithography simulation is used as the golden reference to learn from. The training data comprises good layout samples

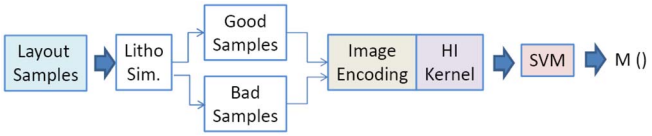

Fig. 14. Learning in lithography simulation context [58].

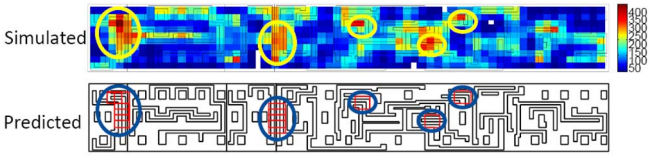

Fig. 15. Fast prediction of variability (hot spots).

and bad layout samples where good and bad are determined by the lithography simulation. These samples are extracted from a layout by applying Raster scanning, i.e., moving window with overlapping boundary [58]. The learning model $M()$ is a classifier that when applied to a future layout through Raster scanning, can identify the windows that are bad.

SVM was used in [58] for learning the binary classifier. However, similar to the filtering component [40], the main challenge for learning is in defining the feature space. The work in [58] used the histogram intersection kernel and experimentally showed its effectiveness. Another issue is the selection of the window size in Raster scan, which in [58] the size was determined experimentally with cross validation.

To illustrate the application of the classifier $M()$, Fig. 15 shows a result comparing the prediction accuracy of a model $M()$ to the lithography simulation. Most of the high variability areas (hot spots circled) identified by the simulation were correctly identified by the model M.

1) Other Pointers: A more recent work in [57] proposed a prelearning layout analysis approach to extract relevant features and showed great improvement in prediction accuracy. A benchmark suite was constructed later to facilitate research in the area [59]. A recent good review of latest research in the area can be found in [56].

C. Design-Silicon Timing Correlation

In design-silicon timing correlation, the idea is to learn from timing test data to uncover issues in timing analysis [60]. Test data is represented as a set of paths and their measured timing on silicon chips. The mismatch is characterized as the differences between silicon-observed path timing and predicted timing by timing analysis. The problem can be approached with binary classification [60] or with regression [61].

In this type of analytics, each path is encoded with a set of features. These features are our hypothesis basis for the reasoning. In the early works [60], [61], the goal was to rank features, i.e., identifying which features are important for explaining the mismatch. In later works [33], [62], rule learning methods were applied to discover rules (combinations of features) for explaining the mismatch. The methodology is similar to the feedback component discussed above.

In [62], a rule learning methodology is applied to a high-performance 4-core processor. The core question is to answer why a large number of timing analysis predicted critical paths are not critical on silicon, while a number of silicon critical paths are not predicted as critical. After some effort of pruning the test data, in the final analysis, the first set contains more

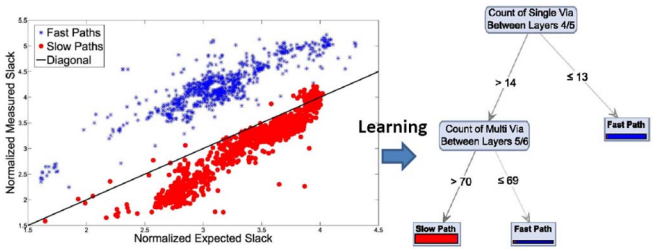

Fig. 16. Diagnose unexpected timing paths.

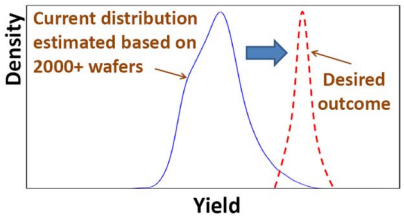

Fig. 17. Yield distribution plot based on 2000+ wafers.

than 12K paths while the second set has 158 paths. There are 96 features defined in the methodology, including features related to basic path characteristics, information from timing report, usage of various Vt devices, usage of cell types, RC information, usage of MUX cells, and location of the path.

Applying the methodology discovered several rules to explain 115 silicon critical paths. Analysis of the rules pointed to usage issues of several custom cell types. Hence, actions could be taken to improve those 115 paths [62].

Fig. 16 shows another example applying a similar methodology to another commercial processor [1]. The left plot shows two clusters of paths: 1) those whose silicon timing is faster than the predicted timing and 2) those whose silicon timing is slower. These paths belong to the same design block and the mismatch result was totally unexpected. The right shows one learning rule uncovered by the methodology. This rule basically says that if a path contains a large number of layers-4-5 vias and layers-5-6 vias it would be a slow path. Later it was confirmed that the issue causing the slow paths occurred on metal layer 5.

1) Other Pointers: Timing analysis is not the only area where test data can be utilized for improvement. Another common approach is to identify systematic defects and use that information to improve design, for example to diagnose layout issues. For example, the works in [63]–[65] are recent results reported along that line of research.

V. Post-Silicon Application Examples

Three example application areas are discussed in this section: yield improvement, customer return modeling, and test cost reduction. The discussion focuses more on the yield improvement to echo some key concepts discussed before.

A. Yield Improvement

Fig. 17 shows a yield issue analyzed in [66]. The data is collected from an automotive product line which is a tire pressure monitor sensor. Fig. 17 illustrates that the yield spreads widely across 2000+ wafers. Note that this result seen in Fig. 17 was already after one design revision and multiple test revisions attempted to improve the yield [66].

The goal is to determine if the yield can be improved by adjusting some process parameters. More than 130 process

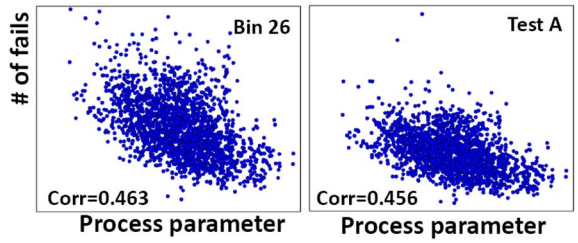

Fig. 18. Examples of the best results in the initial analysis.

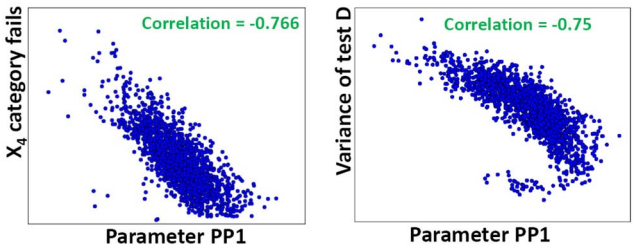

Fig. 19. Examples of improved results with deep analysis.

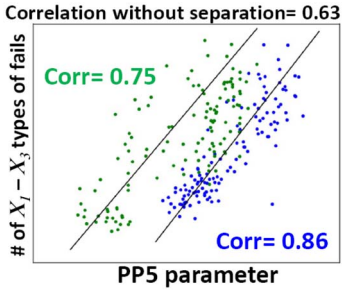

Fig. 20. Example of uncovering temporal effect.

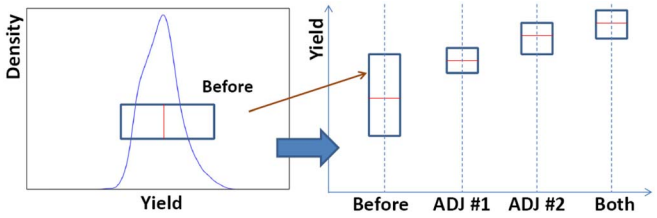

Fig. 21. Silicon validation of recommended process adjustments.

parameters are measured on five sites on a wafer [66]. They are usually called *E-tests* (or *class probes*). Failing dies on each wafer are organized into *test bins*. The essence of the problem is to search for strong *correlations* between *E*-tests and the failures seen in testing.

A common practice in yield analysis is to check the correlation between *E*-tests and the *number of fails* in a test bin or due to a test (a test bin collects fails from related tests). For example, suppose for an *E*-test its *average* measured value over five sites on a wafer, across all n wafers, are $\vec{e} = \{e_1, \ldots, e_n\}$. On the other hand, the numbers of fails, based on a test or test bin, across the wafers are $\vec{t} = \{j_1, \ldots, j_n\}$. One calculates the correlation Corr$(\vec{e}, \vec{t})$ between these two vectors.

Fig. 18 shows examples of the best results with the common correlation analysis practice. Every dot represents a wafer. Bin 26 is the bin with the largest number of fails while test A is a test in the bin. The correlations are not strong enough to convince the foundry to implement a process change.

The work in [66] proposes various ways to perform "deep" correlation analysis beyond the common practice. The deep analysis is based on two key observations.

First, correlations between *E*-tests and the failures can exist beyond the number of fails and the average value over five sites. These two are aggregate statistics. Other more detailed statistics can be used. For example, on the *E*-test side, a correlation can be to the value on an individual site or average over a subset, but not necessarily to the average over all sites.

On the failure side, a correlation can be to how a test decides the failures. For example, a test can have multiple test values. A correlation can be to one of the particular test value. Moreover, a correlation can be to the test value distribution seen on a wafer, for example to some statistics derived from the distribution such as its mean, variance, skew, or kurtosis.

On the failure side, there can be two additional aspects to consider. A correlation can be to a subset of dies on a particular region of a wafer. This is a *spatial* consideration. Further, a correlation can be to a subset of wafers produced in a particular time window. This is a *temporal* consideration.

The second observation is that, the word "correlation" commonly referred in practice does not necessarily mean "statistical correlation." Recall that the ultimate objective is to discover process parameter changes that can improve the yield. Consider an extreme case where a test t has only two values $\{v_1, v_2\}$ where v_1 means pass and v_2 means fail. Suppose there is a process parameter p such that $p > u_1$ implies $t = v_1$ and $p < u_2$ implies $t = v_2$ for some $u_1 > u_2$. If one runs a statistical correlation check between t and p, there might not be a high correlation. However, if one adjust the parameter to the u_1 corner, more dies would pass. This simple case illustrates that the search should also consider searching for a strong *association* between an *E*-test and a failure type. We use the term *association* because the concept can be thought of as similar to that being searched in association rule mining [34].

Fig. 19 shows two examples of improved results based on the deep analysis methodology proposed in [66]. In the left plot, the number of X_4 type of fails is correlated to a parameter PP1. Note that X_4 denotes a particular test value of test A shown in Fig. 18. In the right plot, the variance of measured values on another test D is correlated to PP1.

Fig. 20 shows an example of considering the temporal aspect. In this example, parameter PP5 is correlated to the number of X_1–X_3 types of fails. The wafers are selected into two subsets, colored as green and blue dots. The separation is based on the time of their production. Additionally, not all wafers are included.

With the deep analysis methodology [66], five parameter (PP1–PP5) changes were recommended to and accepted by the foundry. These changes are implemented as two process adjustments, ADI #1 and ADJ #2 whose effects are validated through manufacturing lots as shown in Fig. 21. The box plot shows the mean and the 25%–75% quantile spread of the yield. In the figure, we observe that individually each adjustment improved the yield while combined they achieved an even better improvement. These results were confirmed on actual silicon and hence, both adjustments were applied in mass production afterward.

1) Discussion: The yield example discussed above re-emphasizes the point that analytics is not a one-shot run, but an iterative search process. Refer to the KD process in Fig. 3 again. In each iteration, the analyst prepare the datasets to determine if a particular type of correlation or

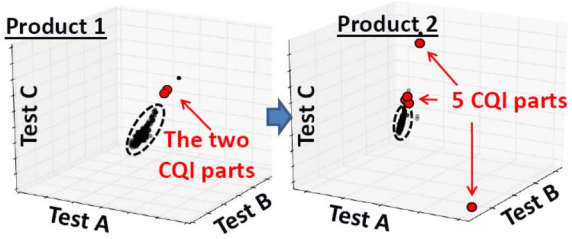

Fig. 22. Predicting CQIs via an outlier model.

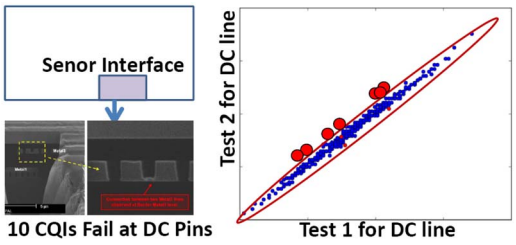

Fig. 23. Validating a CQI model via FA report.

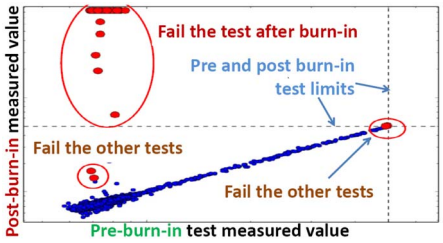

Fig. 24. Challenge in burn-in cost reduction.

association exists. Then, the results from an analysis tool are evaluated by the analyst to make such determination.

A key reason why the product team failed to discover the adjustment and the work [66] did, was due to the fact that the approach in [66] examined the data from the perspectives (ways to prepare the datasets) that were never considered by the product team. This shows that the effectiveness of data mining is not just about the analysis tool, and can largely depend on the methodology to prepare the datasets.

B. Customer Return Modeling

Customer returns are parts that pass all the tests before shipment but are determined as failing parts by the customer. Customer returns can be called customer quality incidents (CQIs) because returning by the customer does not necessarily mean that the parts are defective at their shipping time. For CQI analysis, one way is to search for test models that can screen those parts using test data prior to the shipment.

In a sense, CQI analysis involves: 1) searching for abnormalities in the data and 2) associating the abnormalities with the returns. There can be two types of abnormalities: 1) a CQI is an outlier in a *test space* defined by one or more tests [25] and 2) a CQI is located in a region of wafer showing an abnormal wafer pattern [67].

For example, if the goal is to project a CQI as an outlier, the search can be to find a test space in which the CQI is classified as an outlying part. If there are k tests, the naive search space comprises all possible $2^k - 1$ test spaces. Therefore, reducing this search space is a key consideration in practice [68].

As it will be discussed later, the definition of an "outlier" can be quite subjective. Hence, one of the major challenges in CQI analysis is to validate an outlier model [25], [67].

In practice, there are two ways to validate a model. For example, Fig. 22 shows that a 3-D outlier space is learned based on one CQI, which also projects another CQI as an outlier. The second CQI provides a validation of the model. When the model is applied to a sister product, it can capture five more CQIs as outliers, hence providing further validation. These products are SoCs sold to the automotive market [67].

Fig. 23 shows another example where the outlier model is validated through design knowledge and failure analysis (FA) report [68]. In this example, ten CQIs are analyzed. We knew that they all failed due to some defective dc pins on the sensor interface. Seven out of ten CQIs can be projected as (marginal) outliers in a test space comprising two dc line tests.

1) Discussion: In practice, implementing an outlier analysis approach to model and screen fails is not as simple as showing the results in Figs. 22 and 23. First, preparing the data for the analysis can consume much of the engineering efforts. For example, there can be missing data, corrupted data, or unavailable entries in the data. Ensuring data integrity can be a rather tedious task, taking much more time than the analytics. Second, while models like Figs. 22 and 23 provide evidences that they can be useful to screen the parts, whether they can be implemented and applied in an actual test flow can still be in question. For example, such a model involves yield loss, i.e., to screen a bad part, some good parts are also screened out. One needs to carefully estimate the tradeoff between yield loss and quality improvement before a model can be adopted. Moreover, impacts to the existing test flow need to be considered. As discussed in [69], it could take years of extensive study to evaluate the impacts on a variety of production flows to deploy a test data analytics methodology in practice.

C. Test Cost Reduction

Another important area of test data analytics is test cost reduction. Test cost reduction can take place within a test stage [70] or across multiple stages [71]. In test cost reduction, one desires to remove a test [70], remove a test insertion, or reduce the parts going through an expensive test stage like burn-in [71]. This means that, for all those parts that are supposed to be captured by the removed tests or removed test stage, now they need to be screened by other tests or in an earlier test stage. In other words, the fundamental problem can be viewed as to predict a future fail in the test flow by constructing a test model such as an outlier model [69].

The earlier paper [1] explains that test cost reduction can be quite challenging if one desires a very small defective parts per million (DPPM) impact. For example, if the target DPPM impact is 10, this means that at most there can be ten defective parts (per million) being miss-predicted. Defective parts often occur at the tail of a test distribution and the requirement demands a model to capture the behavior of the tail in high accuracy, which can be difficult [1].

Fig. 24 illustrates another issue which can make test cost reduction challenging. This is in the context of burn-in reduction. Each dot represents a part. The x, y of a part represent their measured value based on the same test, before and after the burn-in. The plot shows that there are many parts (within the big red circle) whose test values before the burn-in are very normal, but those values after the burn-in become abnormal. This shows that the burn-in does alter the characteristics

on those parts. If one does not have an earlier test (e.g., a wafer-level stress test) that can alter the characteristics in a similar way, it become questionable why a test model built on other tests can even predict those fails.

VI. On the Robustness of Data Analytics

In the applications discussed above, there are two types of analyses involved: 1) correlation analysis and 2) outlier analysis. In this section, we discuss the robustness issue of a data mining tool using these two types of analyses as examples.

An analysis tool typically computes some statistics as its outputs, for example in correlation analysis a correlation number and in outlier analysis an *outlier score*. In a KD process, it is up to the user to determine how meaningful these statistics are. Because this determination can be subjective, the KD process is not robust. This determination can further be complicated by the enormous search space. Consequently, it is difficult to know if the best models and statistics seen in hand are in fact the best that can be provided by the data.

To alleviate the robustness issue, one would desire a tool that by its own analysis can report "no model found." For example, in correlation analysis, the tool can report no correlation and in outlier analysis, the tool can report that there is no outlier.

Take correlation analysis as an example. Before adjusting a process parameter, it is important to ensure that there is no correlation between the parameter and other test results not intended to be affected. This is to ensure that the adjustment would not cause an unintended yield loss on another test [66].

A. Establishing No Correlation

In theory, showing no correlation can be approached by showing *statistical independence* [72]. Rényi [73] proposed an equation to capture statistical dependence: $\mathcal{Q}(P(x, y)) = \sup_{f,g} \mathrm{Corr}(f(x), g(y))$, where x and y are two random variables and f and g are Borel measurable and bounded functions (The "sup" denotes the least upper bound and "Corr" denotes the correlation calculation function). Rényi [73] showed that the quantity $\mathcal{Q}(P(x, y)) = 0$ implies statistical independence. Notice that independence is established based on considering all possible functions f and g.

If we modify Rényi's equation with matrices X, Y, we get: $CC(X, Y) = \max_{f,g} \mathrm{Corr}(f(X), g(Y))$, where f and g are some functions transforming X, Y into two vectors. For example, in canonical correlation analysis (CCA) [3] the two matrices X, Y are transformed with two weight vectors w_x and w_y into Xw_x and Yw_y, respectively, before their correlation is calculated with Corr(). Hence, maximizing based on f, g becomes maximizing based on the weight vectors. Note that $\mathrm{CCA}(X, Y) = 0$ implies there is no linear correlation, but nonlinear correlation can still exist.

To extend f, g to be nonlinear functions, one approach is kernel CCA (KCCA) [22]. However, it turns out that KCCA is not practical for showing statistical independence. Steinwart [75] show that with a universal kernel such as a Gaussian kernel [22], KCCA result is always 1, independent of the dataset. Then, Gretton *et al.* [76] show that with *regularization* (the concept is discussed in Section II-F) and universal kernels, $\mathrm{KCCA}(X, Y) = 0$ iff X, Y are independent. Regularized KCCA requires user to choose a regularization parameter γ. Experimentally, we find that this subjectivity makes it hard to interpret the results in practice [66].

In the yield analysis work [66], a different approach is taken following the idea proposed in [77]. The idea is to approximate KCCA by: 1) running kernel PCA [78] to extract the first p principal components in the feature space and 2) running CCA directly in the feature space with the p principal components. The value p is then used as the *model complexity measure*.

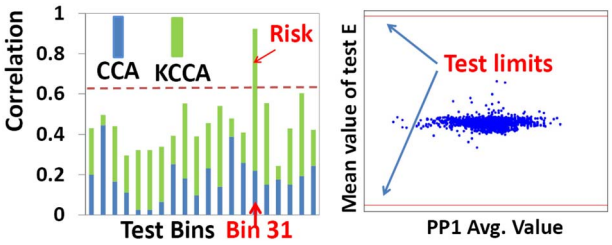

Fig. 25. Example of checking for no correlation.

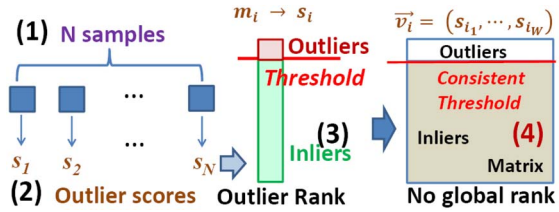

Fig. 26. Traditional outlier analysis versus consistent outlier analysis.

In practice, the correlation is checked up to a certain p value (i.e., one cannot determine no correlation in an absolute sense). Fig. 25 shows an example ($p = 17$) of *risk evaluation* in the yield application discussed above, employing the kernel correlation check. The plot shows both CCA and KCCA results on various test bins, as how each number of fails correlates to process parameter PP1 (which was adjusted).

Bin 31 shows a substantially higher risk than others. On the right plot (each dot is a wafer), the average test value of test E (the only test in bin 31) are plotted against the average PP1 measured value. Risk is contained by seeing that most of the test E values are far from the test limits and moving PP1 to the right corner (e.g., as suggested by Fig. 19 above) would not move the test E values up or down much.

B. Establishing No Outlier

Traditionally, outlier analysis is done by assuming that the data follows a known distribution, such as a normal or exponential distribution. In practice, this approach is not robust because the data almost never exactly matches an assumed distribution. A small error in modeling can introduce a large error in modeling the tails of a distribution, and the tails are most important with respect to outlier analysis.

Take parametric outlier screening [79]–[81] as an example, which is commonly applied for automotive products due to their extremely high quality requirement [82]. Fig. 26 illustrates three major components in outlier modeling.

The modeling begins with N samples such as parts from the same wafer. An *outlier method* is applied to calculate an *outlier score* for every sample. This results in an outlier rank. Then, a *threshold* is selected to separate samples into outliers and inliers. The process is inherently subjective. The choices of the sample set, the outlier score calculation method, and the threshold can all impact the outlier analysis outcome. Even with a modern novelty detection algorithm such as one-class SVM [83] where the threshold is preset at 0, another parameter ν, which represents an upper bound on the fraction of outliers, still requires user input.

To reduce the subjectivity, the work in [84] proposes a concept called *consistency check* which is labeled in Fig. 26 as the fourth component.

In the original outlier score calculation, each die m_i receives a score s_i based on the population of dies on the wafer. In consistency check, each die m_i receives a vector of scores $s_{i_1}, \ldots, s_{i_W}$ where each s_{i_j} is calculated based on the population of wafer j (assuming in total W wafers). In this way, a *consistent threshold* is defined as a threshold such that an outlier has every score greater than every corresponding score of an inlier. A minimum consistency threshold is the consistent threshold that results in the maximum number of outliers.

The work in [84] shows that in practice, data from many tests show no consistent outlier, i.e., cannot find a consistent threshold. Hence, it provides a way to determine that there is no outlier based on a test, which can be useful to facilitate the search process in CQI analysis discussed above.

The intuition behind consistency check is that the "gap" between an outlier and an inlier has to be larger than the *noise* seen in the data [84]. In test, outlier analysis is often wafer based. Hence, the noise can be measured based on wafer-to-wafer variations. However, this means that in consistency check, one has to first identify systematic shifts of a test value across wafers before applying the consistency check. Otherwise, a systematic shift would be treated as noise in the consistency check, which can be undesirable. For this reason, the work in [84] also implements a clustering-based approach to detect systematic shifts across wafers, using the ideas suggested in [85] and [86].

C. Other Pointers

Test data analytics is a rich field with some applications starting much earlier (see [87]) than those discussed above. For example, outlier analysis and adaptive test have been hot topics in test for many years [69], [88]–[90]. Applications to analog/RF devices [91]–[94] are popular. Optimizing cost and quality based on wafer modeling is another rich research topic [95]–[100]. Yield learning and test chip optimization are also important applications with analytics [101], [102]. Analyzing system-level test data is another important topic emerging in recent years [103]. Again, this list is incomplete. They are simply pointers to some of the research teams in the field.

VII. Conclusion

This paper emphasizes on the view that learning from data is a process of enhancing one's domain knowledge using the data. Because of the involvement of domain knowledge, learning from data is subjective to the knowledge. And because of this subjectivity, one would not be 100% sure that the conclusion drawn from the findings is always "correct." Hence, the goal of data mining is not to guarantee 100% a conclusion, but to discover evidences strong enough to support an action with minimal subjectivity in the discovery process.

The question is not about whether one needs domain knowledge, but about how much knowledge is required. This view is not only based on experience, but also consistent with the no free lunch principle fundamental to machine learning [6].

In practice, domain expertise plays a critical role in developing or using an analytics tool. A domain expert may develop a new tool or customize an existing tool to facilitate the KD process for an application context. Developing the methodologies to facilitate the dataset preparation and meaningfulness determination steps can also be largely empirical based on the domain expert's experience.

The KD view and the search perspective are based on the specific applications from our experience. They should not be taken as universally applicable in all applications in EDA and test. Further, while in this paper we discuss several applications somewhat reaching production level successes, it should not be taken directly that they are ready for a commercial implementation or production deployment. Additional research might be required to reach that stage. Based on the experience, there can be four directions for future research.

1) Section VI discusses some initial works to address the robustness issue in data mining. Much research is required to develop robust tools to support a robust KD process.
2) While one expert following a KD methodology with data mining tools can deliver promising results, it does not mean that the methodology can be easily implemented or can be used effectively by other people in a company. Again, one reason for this scalability difficulty is due to the subjectivity in the KD search process—different people may involve different subjectivities. Hence, minimizing subjectivity in KD is crucial to enable its usage scalability.
3) As mentioned before, design is an evolutionary process. Data mining should take advantage of this evolution. This means that knowledge should be learned, accumulated, and reused. However, this requires the development of a formal representation of knowledge in each specific application domain.
4) In a sense, deep learning [14] reduces subjectivity by asking an LM to figure out what the important features are. This is different from the KD view where the search for important features is carried out iteratively, outside the learning tool box. How to employ the deep learning concept to provide more capability to the learning tools in a KD process to minimize the number of iterations in the search can be another interesting research direction. However, because deep learning usually requires a rather large amount of data, how effective it can be with limited data (such as the applications discussed in this paper) remains to be studied.

References

[1] L.-C. Wang and M. S. Abadir, "Data mining in EDA—Basic principles, promises, and constraints," in *Proc. Design Autom. Conf.*, San Francisco, CA, USA, 2014, pp. 1–6.
[2] F. Pedregosa *et al.*, Scikit-learn: Machine learning in Python, *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.
[3] D. R. Hardoon, S. Szedmak, and J. R. Shawe-Taylor, "Canonical correlation analysis; an overview with application to learning methods," *Neural Comput.*, vol. 16, no. 12, pp. 2639–2664, 2004.
[4] K. J. Cios *et al.*, *Data Mining—A Knowledge Discovery Approach*. New York, NY, USA: Springer, 2007.
[5] O. Bousquet, S. Boucheron, and G. Lugosi, *Introduction to Statistical Learning Theory* (LNCS 3176). Heidelberg, Germany: Springer, 2004, pp. 169–207.
[6] D. H. Wolpert, "The lack of a priori distinctions between learning algorithms," *Neural Comput.*, vol. 8, no. 7, pp. 1341–1390, 1996.
[7] E. M. Gold, "Language identification in the limit," *Inf. Control*, vol. 10, no. 5, pp. 447–474, 1967.
[8] D. Angluin and C. H. Smith, "Inductive inference: Theory and methods," *ACM Comput. Surveys*, vol. 15, no. 3, pp. 237–269, 1983.
[9] L. G. Valiant, "A theory of the learnable," *Commun. ACM*, vol. 27, no. 11, pp. 1134–1142, 1984.
[10] M. J. Kearns and U. V. Vazirani, *An Introduction to Computational Learning Theory*. Cambridge, MA, USA: MIT Press, 1994.

[11] O. Guzey *et al.*, *Extracting a Simplified View of Design Functionality Based on Vector Simulation* (LNCS 4383). Heidelberg, Germany: Springer, 2007, pp. 34–49.

[12] V. N. Vapnik, *The Nature of Statistical Learning Theory*, 2nd ed. New York, NY, USA: Springer, 1999.

[13] K. R. Popper, *The Logic of Scientific Discovery*, 2nd ed. New York, NY, USA: Harper & Row, 1968.

[14] Y. Bengio, I. Goodfellow, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org/

[15] B. Schölkopf and A. J. Smola, *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001.

[16] B. N. Lee, L.-C. Wang, and M. S. Abadir, "Issues on test optimization with known good dies and known defective dies—A statistical perspective," in *Proc. IEEE Int. Test Conf. (ITC)*, Santa Clara, CA, USA, 2006, pp. 1–10.

[17] T. J. Hastie, R. J. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (Springer Series in Statistics). New York, NY, USA: Springer, 2001

[18] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*. Belmont, CA, USA: Wadsworth, 1984.

[19] L. Breiman, "Random forests," *Mach. Learn. J.*, vol. 45, no. 1, pp. 5–32, 2001.

[20] N. Lavrač, B. Kavšek, P. Flach, and L. Todorovski, "Subgroup discovery with CN2-SD," *J. Mach. Learn. Res.*, vol. 5, pp. 153–188, Dec. 2004.

[21] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2003.

[22] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[23] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.

[24] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A Study of the behavior of several methods for balancing machine learning training data," *Sigkdd Explor.*, vol. 6, no. 1, pp. 20–29, 2004.

[25] N. Sumikawa, J. Tikkanen, L.-C. Wang, L. Winemberg, and M. S. Abadir, "Screening customer returns with multivariate test analysis," in *Proc. Int. Test Conf.*, Anaheim, CA, USA, 2012, pp. 1–10.

[26] N. Sumikawa, D. Drmanac, L.-C. Wang, L. Winemberg, and M. S. Abadir, "Important test selection for screening potential customer returns," in *Proc. VLSI Design Autom. Test Symp.*, Hsinchu, Taiwan, 2011, pp. 1–4.

[27] Z. Zheng, X. Wu, and R. Srihari, "Feature selection for text categorization on imbalanced data," *Sigkdd Explor.*, vol. 6, no. 1, pp. 80–89, 2004.

[28] J. Chen *et al.*, "Data learning techniques and methodology for Fmax prediction," in *Proc. Int. Test Conf.*, Austin, TX, USA, 2009, pp. 1–10.

[29] I. T. Jolliffe, *Principal Component Analysis*. New York, NY, USA: Springer, 1986.

[30] A. Hyvärinen *et al.*, *Independent Component Analysis* (Wiley Series on Adaptive and Learning Systems). New York, NY, USA: Wiley, 2001.

[31] P. M. O'Neill, "Production multivariate outlier detection using principal components," in *Proc. Int. Test Conf.*, Santa Clara, CA, USA, 2008, pp. 1–10.

[32] R. Turakhia, B. Benware, R. Madge, T. Shannon, and R. Daasch, "Defect screening using independent component analysis on I_{DDQ}," in *Proc. IEEE VLSI Test Symp.*, Palm Springs, CA, USA, 2005, pp. 427–432.

[33] N. Callegari, D. Drmanac, L.-C. Wang, and M. S. Abadir, "Classification rule learning using subgroup discovery of cross-domain attributes responsible for design-silicon mismatch," in *Proc. Design Autom. Conf.*, Anaheim, CA, USA, 2010, pp. 374–379.

[34] C. Zhang and S. Zhang, *Association Rule Mining: Models and Algorithms* (LNCS 2307). New York, NY, USA: Springer, 2002.

[35] O. Guzey and L. C. Wang, "Coverage-directed test generation through automatic constraint extraction," in *Proc. IEEE Int. Workshop High Level Design Validation Test (HLDVT)*, Irvine, CA, USA, 2007, pp. 151–158.

[36] Y. Katz, M. Rimon, A. Ziv, and G. Shaked, "Learning microarchitectural behaviors to improve stimuli generation quality," in *Proc. DAC*, 2011, pp. 848–853.

[37] L.-C. Wang, "Data mining in functional test content optimization," in *Proc. Asia South Pac. Design Autom. Conf.*, 2015, pp. 308–315.

[38] O. Guzey, L.-C. Wang, J. Levitt, and H. Foster, "Functional test selection based on unsupervised support vector analysis," in *Proc. DAC*, Anaheim, CA, USA, 2008, pp. 262–267.

[39] P.-H. Chang, L.-C. Wang, and J. Bhadra, "A kernel-based approach for functional test program generation," in *Proc. Int. Test Conf.*, Austin, TX, USA, 2010, pp. 1–10.

[40] W. Chen *et al.*, "Novel test detection to improve simulation efficiency—A commercial experiment," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, 2012, pp. 101–108.

[41] L.-C. Wang and M. Marek-Sadowska, "Machine learning in simulation-based analysis," in *Proc. Int. Symp. Phys. Design*, Monterey, CA, USA, 2015, pp. 57–64.

[42] W. Chen, L.-C. Wang, J. Bhadra, and M. Abadir, "Simulation knowledge extraction and reuse in constrained random processor verification," in *Proc. DAC*, Austin, TX, USA, 2013, pp. 1–6.

[43] K.-K. Hsieh, W. Chen, L.-C. Wang, and J. Bhadra, "On application of data mining in functional debug," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, 2014, pp. 670–675.

[44] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," in *Proc. Int. Conf. Extending Database Technol.*, Avignon, France, 1996, pp. 3–17.

[45] H. Mannila, H. Toivonen, and A. I. Verkamo, "Discovery of frequent episodes in event sequences," *Data Min. Knowl. Disc.*, vol. 1, no. 3, pp. 259–289, 1997.

[46] S. Fine and A. Ziv, "Coverage directed test generation for functional verification using Bayesian networks," in *Proc. Design Autom. Conf.*, Anaheim, CA, USA, 2003, pp. 286–291.

[47] S. Vasudevan *et al.*, "GoldMine: Automatic assertion generation using data mining and static analysis," in *Proc. Design Autom. Test Europe*, Dresden, Germany, 2010, pp. 626–629.

[48] S. Hertz, D. Sheridan, and S. Vasudevan, "Mining hardware assertions with guidance from static analysis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 6, pp. 952–965, 2013.

[49] W. Li, "Specification mining: New formalisms, algorithms and applications," Ph.D. dissertation, Elect. Eng. Comput. Sci., Univ. California at Berkeley, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2014-20, 2013.

[50] A. DeOrio, Q. Li, M. Burgess, and V. Bertacco, "Machine learning-based anomaly detection for post-silicon bug diagnosis," in *Proc. Design Autom. Test Europe (DATE)*, 2013, pp. 491–496.

[51] D. Lee *et al.*, "Learning-based power modeling of system-level black-box IPs," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, Austin, TX, USA, 2015, pp. 847–853.

[52] X. Zheng, P. Ravikumar, L. K. John, and A. Gerstlauer, "Learning-based analytical cross-platform performance prediction," in *Proc. Int. Conf. Embedded Comput. Syst. Archit. Model. Simulat. (SAMOS)*, Samos, Greece, 2015, pp. 52–59.

[53] D.-C. Juan, "A learning-based framework incorporating domain knowledge for performance modeling," Ph.D. dissertation, Elect. Comput. Eng., Carnegie Mellon Univ., Pittsburgh, PA, USA, 2014.

[54] H. Zhao *et al.*, "Learning based compact thermal modeling for energy-efficient smart building management," in *Proc. Int. Conf. Comput.-Aided Design*, Austin, TX, USA, 2015, pp. 450–456.

[55] F. Wang, M. Zaheer, X. Li, J.-O. Plouchart, and A. Valdes-Garcia, "Co-learning Bayesian model fusion: Efficient performance modeling of analog and mixed-signal circuits using side information," in *Proc. ICCAD*, Austin, TX, USA, 2015, pp. 575–582.

[56] B. Yu, D. Z. Pan, T. Matsunawa, and X. Zeng, "Machine learning and pattern matching in physical design," in *Proc. Asia South Pac. Design Autom. Conf. (ASPDAC)*, 2015, pp. 286–293.

[57] D. Ding, J. A. Torres, and D. Z. Pan, "High performance lithography hotspot detection with successively refined pattern identifications and machine learning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 11, pp. 1621–1634, Nov. 2011.

[58] D. G. Drmanac, F. Liu, and L.-C. Wang, "Predicting variability in nanoscale lithography processes," in *Proc. Design Autom. Conf.*, San Francisco, CA, USA, 2009, pp. 545–550.

[59] J. A. Torres, "ICCAD-2012 CAD contest in fuzzy pattern matching for physical verification and benchmark suite," in *Proc. IEEE Int. Conf. Comput.-Aided Design*, San Jose, CA, USA, 2012, pp. 349–350.

[60] L.-C. Wang, P. Bastani, and M. S. Abadir, "Design-silicon timing correlation—A data mining perspective," in *Proc. Design Autom. Conf.*, San Diego, CA, USA, 2007, pp. 384–389.

[61] P. Bastani, N. Callegari, L. Wang, and M. Abadir, "Statistical diagnosis of unmodeled systematic timing effects," in *Proc. Design Autom. Conf.*, Anaheim, CA, USA, 2008, pp. 355–360.

[62] J. Chen *et al.*, "Mining AC delay measurements for understanding speed-limiting paths," in *Proc. Int. Test Conf.*, Austin, TX, USA, 2010, pp. 1–10.

[63] W. C. J. Tam and R. D. S. Blanton, "LASIC: Layout analysis for systematic IC-defect identification using clustering," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 8, pp. 1278–1290, Aug. 2015.

[64] W. C. Tam and R. D. Blanton, "Design-for-manufacturability assessment for integrated circuits using RADAR," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 10, pp. 1559–1572, Oct. 2014.

[65] W. C. Tam and R. D. Blanton, "Physically-aware analysis of systematic defects in integrated circuits," *IEEE Des. Test Comput.*, vol. 29, no. 5, pp. 81–93, Oct. 2012.

[66] J. Tikkanen, S. Siatkowski, N. Sumikawa, L.-C. Wang, and M. S. Abadir, "Yield optimization using advanced statistical correlation methods," in *Proc. Int. Test Conf.*, Seattle, WA, USA, 2014, pp. 1–10.

[67] N. Sumikawa, L.-C. Wang, and M. S. Abadir, "A pattern mining framework for inter-wafer abnormality analysis," in *Proc. Int. Test Conf.*, Anaheim, CA, USA, 2013, pp. 1–10.

[68] J. Tikkanen, N. Sumikawa, L.-C. Wang, and M. S. Abadir, "Multivariate outlier modeling for capturing customer returns—How simple it can be," in *Proc. IEEE On-Line Test Symp.*, 2014, pp. 164–169.

[69] K. M. Butler, A. Nahar, and R. Daasch, "What we know after twelve years developing and deploying test data analytics solutions," in *Proc. Int. Test Conf.*, 2016, pp. 1–8.

[70] D. Drmanac and M. Laisne, "Wafer probe test cost reduction of an RF/A device by automatic testset minimization—A case study," in *Proc. Int. Test Conf.*, Anaheim, CA, USA, 2011, pp. 1–10.

[71] N. Sumikawa, L.-C. Wang, and M. S. Abadir, "An experiment of burn-in time reduction based on parametric test analysis," in *Proc. Int. Test Conf.*, Anaheim, CA, USA, 2012, pp. 1–10.

[72] J. Jacod and P. E. Protter, *Probability Essentials*. New York, NY, USA: Springer, 2000.

[73] A. Rényi, "On measures of dependence," *Acta Mathematica Academiae Scientiarum Hungarica*, vol. 10, nos. 3–4, pp. 441–451, 1959.

[74] F. R. Bach and M. I. Jordan, "Kernel independent component analysis," *J. Mach. Learn. Res.*, vol. 3, pp. 1–48, Jul. 2002.

[75] I. Steinwart, "On the influence of the kernel on the consistency of support vector machines," *J. Mach. Learn. Res.*, vol. 2, pp. 67–93, Nov. 2001.

[76] A. Gretton, R. Herbrich, A. Smola, O. Bousquet, and B. Schölkopf, "Kernel methods for measuring independence," *J. Mach. Learn. Res.*, vol. 6, pp. 2075–2129, Dec. 2005.

[77] M. Kuss and T. Graepel, "The geometry of kernel canonical correlation analysis," Max Planck Inst. Biol. Cybern., Tübingen, Germany, Tech. Rep. 108, May 2003.

[78] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, no. 5, pp. 1299–1319, 1998.

[79] W. R. Daasch, C. G. Shirley, and A. Nahar, "Statistics in semiconductor test: Going beyond yield," *IEEE Des. Test Comput.*, vol. 26, no. 5, pp. 64–73, Sep./Oct. 2009.

[80] W. R. Daasch, K. Cota, and J. McNames, "Neighbor selection for variance reduction in I_{DDQ} and other parametric data," in *Proc. Int. Test Conf.*, Baltimore, MD, USA, 2001, pp. 92–100.

[81] K. M. Butler *et al.*, "Successful development and implementation of statistical outlier techniques on 90nm and 65nm process driver devices," in *Proc. IEEE IRPS*, 2006, pp. 552–559.

[82] M. J. Moreno-Lizaranzu and F. Cuesta, "Improving Electronic Sensor Reliability by Robust Outlier Screening," *Sensors*, vol. 13, no. 10, pp. 13521–13542, 2013.

[83] B. Schölkopf, R. C. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt, "Support vector method for novelty detection," in *Proc. Neural Inf. Proc. Syst.*, 2000, pp. 582–588.

[84] S. Siatkowski *et al.*, "Consistency in wafer based outlier screening," in *Proc. IEEE VLSI Test Symp.*, Las Vegas, NV, USA, 2016, pp. 1–6.

[85] J. M. Buhmann, "Information theoretic model validation for clustering," in *Proc. Int. Symp. Inf. Theory*, Austin, TX, USA, Jun. 2010, p. 9.

[86] S. Maneewongvatana and D. M. Mount, *On the Efficiency of Nearest Neighbor Searching with Data Clustered in Lower Dimensions* (LNCS 2073). Heidelberg, Germany: Springer, Jul. 2001, pp. 842–851.

[87] P. Nigh and A. Gattiker, "Test method evaluation experiments and data," in *Proc. Int. Test Conf.*, Atlantic City, NJ, USA, 2000, pp. 454–463.

[88] R. Madge *et al.*, "In search of the optimum test set—Adaptive test methods for maximum defect coverage and lowest test cost," in *Proc. Int. Test Conf.*, Charlotte, NC, USA, 2004, pp. 203–212.

[89] K. R. Gotkhindikar, W. R. Daasch, K. M. Butler, J. M. Carulli, Jr., and A. Nahar, "Die-level adaptive test: Real-time test reordering and elimination," in *Proc. Int. Test Conf.*, Anaheim, CA, USA, 2011, pp. 1–10.

[90] W. R. Daasch and R. Madge, "Variance reduction and outliers: Statistical analysis of semiconductor test data," in *Proc. ITC*, Austin, TX, USA, 2005, p. 9.

[91] H.-G. D. Stratigopoulos, P. Drineas, M. Slamani, and Y. Makris, "Non-RF To RF test correlation using learning machines: A case study," in *Proc. VLSI Test Symp.*, Berkeley, CA, USA, 2007, pp. 9–14.

[92] N. Kupp, H. Huang, P. Drineas, and Y. Makris, "Post-production performance calibration in analog/RF devices," in *Proc. Int. Test Conf.*, Austin, TX, USA, 2010, pp. 1–10.

[93] F. Liu, E. Acar, and S. Ozev, "Test yield estimation for analog/RF circuits over multiple correlated measurements," in *Proc. ITC*, Santa Clara, CA, USA, 2007, pp. 1–10.

[94] E. Yilmaz and S. Ozev, "Defect-based test optimization for analog/RF circuits for near-zero DPPM applications," in *Proc. Int. Conf. Comput.-Aided Design*, Lake Tahoe, CA, USA, 2009, pp. 313–318.

[95] N. Kupp, K. Huang, J. M. Carulli, and Y. Makris. "Spatial estimation of wafer measurement parameters using Gaussian process models," in *Proc. Int. Test Conf.*, Anaheim, CA, USA, 2012, pp. 1–8.

[96] A. Ahmadi, K. Huang, S. Natarajan, J. M. Carulli, Jr., and Y. Makris, "Spatio-temporal wafer-level correlation modeling with progressive sampling: A pathway to HVM yield estimation," in *Proc. Int. Test Conf.*, Seattle, WA, USA, 2014, pp. 1–10.

[97] X. Li, R. R. Rutenbar, and R. D. Blanton, "Virtual probe: A statistically optimal framework for minimum-cost silicon characterization of nanoscale integrated circuits," in *Proc. Int. Conf. Comput.-Aided Design*, San Jose, CA, USA, 2009, pp. 433–440.

[98] W. Zhang, X. Li, E. Acar, F. Liu, and R. Rutenbar, "Multi-wafer virtual probe: Minimum-cost variation characterization by exploring wafer-to-wafer correlation," in *Proc. ICCAD*, San Jose, CA, USA, 2010, pp. 47–54.

[99] H.-M. Chang, K.-T. Cheng, W. Zhang, X. Li, and K. M. Butler, "Test cost reduction through performance prediction using virtual probe," in *Proc. Int. Test Conf.*, Anaheim, CA, USA, 2011, pp. 1–9.

[100] F. Lin, C.-K. Hsu, and K.-T. Cheng, "AdaTest: An efficient statistical test framework for test escape screening," in *Proc. Int. Test Conf. (ITC)*, Anaheim, CA, USA, 2015, pp. 1–8.

[101] R. D. Blanton, B. Niewenhuis, and C. Taylor, "Logic characterization vehicle design for maximal information extraction for yield learning," in *Proc. IEEE Int. Test Conf.*, Seattle, WA, USA, 2014, pp. 1–10.

[102] R. D. S. Blanton, B. Niewenhuis, and Z. D. Liu, "Design reflection for optimal test-chip implementation," in *Proc. Int. Test Conf.*, Anaheim, CA, USA, 2015, pp. 1–10.

[103] F. Ye, Z. Zhang, K. Chakrabarty, and X. Gu. "Knowledge discovery and knowledge transfer in board-level functional fault diagnosis," in *Proc. Int. Test Conf.*, Seattle, WA, USA, 2014, pp. 1–10.

Li-C. Wang received the Ph.D. degree from the University of Texas at Austin, Austin, TX, USA, in 1996.

He is a Professor with the Electrical and Computer Engineering Department, University of California at Santa Barbara, Santa Barbara, CA, USA. He was with the PowerPC Design Center, Motorola/IBM, Austin, from 1996 to 2000. Since 2005, his research group has published over 70 papers on the topics related to data mining in electronic design automation (EDA) and test. In the last two years, he has given a number of tutorials on data mining in EDA and test, and invited talks on data mining in several conferences, including Design Automation Conference, International Conference on Computer-Aided Design, Asia and South Pacific Design Automation Conference, and International Symposium on Physical Design.

Dr. Wang was a recipient of the Technical Excellence Award from Semiconductor Research Cooperation in 2010 for contribution on developing data mining technologies in the areas of test and verification, three best paper awards, and three best paper nominations.