Rethinking Logic Minimization for Tabular Machine Learning

Litao Qiao, Weijia Wang, Sanjoy Dasgupta, and Bill Lin, *Member, IEEE*

***Abstract*—Tabular datasets can be viewed as logic functions that can be simplified using two-level logic minimization to produce minimal logic formulas in disjunctive normal form, which in turn can be readily viewed as an explainable decision rule set for binary classification. However, there are two problems with using logic minimization for tabular machine learning. First, tabular datasets often contain overlapping examples that have different class labels, which have to be resolved before logic minimization can be applied since logic minimization assumes *consistent* logic functions. Second, even without inconsistencies, logic minimization alone generally produces complex models with poor generalization because it exactly fits all data points, which leads to detrimental *overfitting*. How best to remove training instances to eliminate inconsistencies and overfitting is highly nontrivial. In this article, we propose a novel statistical framework for removing these training samples so that logic minimization can become an effective approach to tabular machine learning. Using the proposed approach, we are able to obtain comparable performance as gradient boosted and ensemble decision trees, which have been the winning hypothesis classes in tabular learning competitions, but with human-understandable explanations in the form of decision rules. To the best of authors' knowledge, neither logic minimization nor explainable decision rule methods have been able to achieve the state-of-the-art performance before in tabular learning problems.**

***Impact Statement*—Decision rule sets are an important hypothesis class for tabular learning problems in which the ability to provide human understandable explanations is of critical importance. However, they are generally not the winning hypothesis class in terms of accuracy. Black-box models like gradient boosted and ensemble decision trees are generally the superior models. In this article, we revisit the use of logic minimization to derive explainable decision rule sets from tabular datasets. Logic minimization alone produces complex models with poor generalization because it exactly fits all data points as provided. We overcome this problem by removing instances that cause inconsistencies and overfitting via a novel statistical framework. The proposed approach makes possible the learning of decision rules that achieve the state-of-the-art classification performance in tabular learning problems with explainable rule-based predictions, which has not been achieved before.**

***Index Terms*—Decision rule sets, denoising, logic minimization, tabular machine learning.**

I. Introduction

IN MACHINE learning domains, such as healthcare and criminal justice where human lives may be deeply impacted, creating inherently interpretable models that can provide human understandable explanations is critically important [1]. In these domains, the datasets are often provided as tabular data with naturally meaningful features. Due to their intrinsic explainability, decision rule sets [2], [3], [4], [5] are often a popular hypothesis class of choice in these applications. However, they are not the winning class in these tabular learning problems in terms of accuracy. For example, in Kaggle competitions, gradient boosted, and ensemble decision trees [6], [7], [8] are generally the superior models. While these more complex classifiers can provide some level of feature attributions to predictions, their interpretability is limited compared to rule-based sentences that decision rule sets provide, which can be easily understood by humans.

In this article, we explore the use of two-level logic minimization as a means for deriving explainable decision rule sets for tabular learning. An example of a decision rule set with two conjunctive rules is as follows.

IF (Systolic blood pressure $>$ 120) OR
(age $>$ 60 AND cholesterol $=$ very high).
THEN Presence of cardiovascular disease.

In this example, the model would predict someone to have cardiovascular disease if the person has systolic blood pressure above 120, or if the person is above 60 years of age and has a very high level of cholesterol. The model not only provides a prediction, but the corresponding matching rule also provides an explanation that humans can easily understand.[1]

Manuscript received 4 June 2022; revised 9 August 2022 and 5 October 2022; accepted 10 November 2022. Date of publication 28 November 2022; date of current version 22 September 2023. This work was supported by National Science Foundation under Grant 1956339. This paper was recommended for publication by Associate Editor A. Etemad upon evaluation of the reviewers' comments. *(Corresponding author: Litao Qiao.)*

Litao Qiao, Weijia Wang, and Bill Lin are with the Department of Electrical and Computer Engineering, University of California San Diego, La Jolla, CA 92093 USA (e-mail: l1qiao@eng.ucsd.edu; wweijia@eng.ucsd.edu; billlin@eng.ucsd.edu).

Sanjoy Dasgupta is with the Department of Computer Science and Engineering, University of California San Diego, La Jolla, CA 92093 USA (e-mail: dasgupta@eng.ucsd.edu).

Digital Object Identifier 10.1109/TAI.2022.3224415

[1]As discussed in Section V on related work, prior work on decision rule sets has established the benefits of interpretability of decision rule sets for tabular learning problems over black-box models (e.g., [1]), primarily because the activated IF–THEN rule also provides an explanation in terms of human-understandable features. Beyond what has already been studied in the literature about the interpretability of decision rule sets, we do not make further claims in this article regarding the interpretability of decision rule sets. Instead, our focus is on a new logic minimization approach for deriving decision rules that can achieve the state-of-the-art classification performance in tabular learning problems, which neither logic minimization nor decision rule methods have been able to achieve before. We believe advancing the start-of-the-art in both logic minimization and decision rule learning for tabular machine learning is of important significance.

TABLE I
TEST ACCURACY (AS A PERCENTAGE) FOR THE CARDIOVASCULAR DISEASE DATASET (CARDIO) [9]

Logic minimization (no-denoise)	66.03
RIPPER	70.57
XGBoost	73.06
Logic minimization with denoising	73.20

In particular, the explanations are stated directly in terms of meaningful input features, which can be categorical (e.g., color equal to red, blue, or green) or numerical (e.g., age $>$ 60) attributes, where the binary encoding of categorical and numerical attributes is well-studied [4], [5].

When binary encoded, tabular datasets can be viewed as logic functions to be minimized, and the minimized logic in disjunctive normal form (DNF) can be readily viewed as an explainable decision rule set for binary classification. However, tabular datasets often contain overlapping examples that have different class labels, which have to be resolved before logic minimization can be applied since logic minimization assumes *consistent* logic functions. Such inconsistencies can be resolved by taking the majority class such that the largest consistent subset of nonoverlapping training instances is retained. Logic minimization can then be applied to the derived incompletely-specified logic function to fit the data points *exactly* with a minimal number of rules and a minimal number of conditions in each rule with respect to the provided incompletely specified logic function. However, in practice, the logic minimized decision rule set derived this way tends to perform poorly in test accuracy and contains complex rules.

Consider the cardiovascular disease dataset (cardio) from the Kaggle competition [9]. This task predicts whether a patient has cardiovascular disease or not based on the patient's basic information, the results of medical examinations, and the extra information given by the patient. The performance of the logic minimization derived classifier in the abovementioned manner is shown in Table I with a test accuracy of only 66.03% [shown in the row labeled "logic minimization (no-denoise)"], which is quite poor in comparison, for example, to known decision rule learners like RIPPER [2] that achieves 70.57% test accuracy or a state-of-the-art nonexplainable tabular learner like XGBoost (gradient boosted decision tree) [6], which achieves 73.06% test accuracy.

Our conjecture why logic minimization used in the abovementioned manner is not effective in producing accurate classifiers is due in part to the *overfitting* of the training data. In particular, because logic minimization exactly fits all data points, *noisy* data points (those whose label is not the Bayes-optimal choice) can be quite problematic. These noisy data points can lead to a model that both generalizes poorly and is larger than would be needed. In addition, resolving inconsistencies by means of the majority class is often not the best strategy. How best to remove training instances to eliminate overfitting and inconsistencies is highly nontrivial.

To remedy these problems, we propose a statistical framework for *denoising* (to be detailed later) the training dataset by removing a subset of noisy data points, both for purpose of eliminating overfitting and inconsistencies. Logic minimization can then be applied to this edited dataset to produce simple and accurate decision rules from the minimized DNF formula. With the denoising preprocessing step, logic minimization is able to produce a classifier that achieves 73.20% test accuracy for the cardio dataset, as shown in Table I, which is significantly better than logic minimization without denoising, significantly better than known decision rule learners, and comparable to state-of-art tabular learners like XGBoost. As shown in the evaluation section, our logic minimization approach with denoising is able to achieve accuracies within just 0.7% on average over all datasets evaluated in comparison with the state-of-the-art, but nonexplainable tabular learners. Thus, our approach is able to achieve comparable state-of-the-art results while providing human understandable explanations in the form of decision rules. To the best of authors' knowledge, neither logic minimization nor explainable decision rule methods have been able to achieve the state-of-the-art performance before in tabular learning problems.

The rest of this article is organized as follows. Section II formulates tabular learning as a logic minimization problem. Section III introduces our denoising framework to enable logic minimization to achieve the state-of-the-art performance. Section IV provides extensive evaluation of our proposed approach. Section V outlines related work. Finally, Section VI concludes this article.

II. TABULAR LEARNING AS LOGIC MINIMIZATION

As discussed in the previous section, a tabular dataset can be viewed as an incompletely specified logic function that can be minimized into a DNF formula, which can then be readily translated into independent unordered IF–THEN decision rules. In this section, we first provide further details regarding the binarization of tabular datasets into incompletely specified logic functions. We then summarize the role of two-level logic minimization as a decision rule learner.

A. Binarization of Tabular Data

Although binary features commonly appear in tabular datasets, these datasets also generally include categorical and numerical features, which are naturally used when the data is collected. In this work, we assume all data are binary encoded and, thus, categorical and numerical features need to be first binarized using well established preprocessing steps in the machine learning literature. In particular, we follow exactly the same binarization approach used in some decision ruler learners [4], [5], where we simply one-hot encode all categorical features into binary vectors. For numerical features, we adopt quantile discretization based on the distribution of numerical values in the training data to get a set of thresholds for each feature, where the original numerical value is one-hot-encoded into a binary vector by comparing with the thresholds (e.g., age $\leq$ 25, age $\leq$ 50, age $\leq$ 75) and encoded as 1 if less than the threshold

or 0 otherwise. This binarization approach for numerical features has been widely used by decision rule learners and shown to achieve the better performance than directly discretizing numerical values into intervals [4].

B. Logic Minimization as a Decision Rule Learner

Once binary encoded, the tabular dataset can be viewed as an incompletely specified logic function. As noted earlier, when an instance in the dataset has both positive and negative labels, the majority label can be taken to define the incompletely specified logic function. In particular, a binary encoded tabular dataset can be viewed as an incompletely specified logic function $f : \{0,1\}^m \to \{0,1,*\}$ that maps an m-dimensional binary encoded vector $x \in \{0,1\}^m$ into either 0, 1, or $*$. The set of vectors $\{x \in \{0,1\}^m : f(x) = 1\}$ is referred to as the ON-set, the set of vectors $\{x \in \{0,1\}^m : f(x) = 0\}$ is referred to as the OFF-set, and the set of vectors $\{x \in \{0,1\}^m : f(x) = *\}$ is referred to as the dc-set (the do not care set) [10].

With respect to a binary encoded tabular dataset, all instances x with a positive label would be included in the ON-set (i.e., $f(x) = 1$), and all instances x with a negative label would be included in the OFF-set (i.e., $f(x) = 0$). All other input combinations $x \in \{0,1\}^m$ not specified in the encoded dataset would belong to the dc-set (i.e., all input combinations x not specified in the encoded dataset are implicitly defined to be $f(x) = *$).

Given an incompletely specified logic function, well established two-level logic minimization algorithms can be employed to produce a minimized DNF formula as a disjunction (OR) of conjunctive (AND) terms [10]. Modern two-level logic minimization algorithms are able to guarantee a prime and irredundant cover for a given incompletely specified logic function, which means no conjunctive (AND) term can be made simpler by removing a feature (i.e., the conjunctive term is a *prime*), and no conjunctive term can be removed to further simplify the DNF formula (i.e., the cover is *irredundant*). In terms of the corresponding decision rule set, it means no rules can be further simplified or removed from the rule set. However, as noted earlier, logic minimization alone can lead to models with poor generalization due to the presence of noisy instances in the training data that leads to detrimental overfitting. This problem can be remedied by first removing these noisy data points through a denoising process, as described in Section III.

C. Example Rule Set From Logic Minimization

Consider a toy example shown in Table II, corresponding to a truth table derived from a binary-encoded tabular dataset. Logic minimization can be applied to this incompletely specified logic function to produce the following DNF formula:

$$f(x) = x_1 \vee \neg x_2 \vee (x_3 \wedge x_4).$$

This minimized DNF formula corresponds to the following decision rule set.

IF (Age $\leq$ 50) OR
(NOT smoker) or
(Cholesterol $\leq$ 130 AND blood pressure $\leq$ 120)
THEN Low heart disease risk.

TABLE II
TOY EXAMPLE OF AN INCOMPLETELY SPECIFIED LOGIC FUNCTION, WHERE x_1, x_2, x_3, AND x_4 CORRESPOND TO AGE $\leq$ 50, SMOKER, CHOLESTEROL $\leq$ 130, AND BLOOD PRESSURE $\leq$ 120, RESPECTIVELY, AND $f(x)$ REPRESENTS LOW HEART DISEASE RISK

x_1	x_2	x_3	x_4	$f(x)$
0	0	1	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	1	1	0	1

Overall, given a binary-encoded tabular dataset as an incompletely specified logic function, logic minimization produces a decision rule set in DNF as a classifier.

III. DENOISING FORMULATION

We now present a formal model in which the denoising process can be analyzed and understood.

Consider a binary classification task in which data points lie in an instance space $\mathcal{X}$ and the possible labels are $\mathcal{Y} = \{0,1\}$. There is an unknown distribution P over $\mathcal{X} \times \mathcal{Y}$ from which all instances and labels—past, present, and future—are drawn.

The distribution P over (X, Y) pairs can as usual be broken into two parts: the marginal distribution of X, denoted μ, and the conditional probability distribution of Y given X

$$\eta(x) = \Pr(Y = 1 | X = x).$$

A classifier $h : \mathcal{X} \to \mathcal{Y}$ has error rate, or *risk*, $\text{err}(h) = P(h(X) \neq Y)$. The lowest achievable risk is that of the Bayes-optimal classifier

$$g^*(x) = \begin{cases} 1 & \text{if } \eta(x) \geq 1/2 \\ 0 & \text{otherwise.} \end{cases}$$

Notice that if $\eta(x) = 1/2$, then either prediction is optimal.

The risk of g^*, that is, $R^* = \text{err}(g^*)$, is called the *Bayes risk*. In many applications, a significant part of the instance space has η bounded away from 0 and 1 and, thus, $R^* > 0$.

A. Lack of Consistency of Learning Decision Rules by Logic Minimization

Given a dataset $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, logic minimization will find a DNF formula that *exactly* fits all these points. In cases where there is even a little bit of stochasticity in the labels—that is, $R^* > 0$—this can be problematic.

To see this, consider a situation where $\mathcal{X}$ is finite and $\eta(x) \notin \{0,1\}$ (that is, there is some stochasticity in xs label) for all x. Thus, any point x can potentially occur in the dataset with both labels. Given a sufficiently large dataset, this will happen with every point.

For this reason, logic minimization alone is not a consistent method for learning a classifier: it is not guaranteed to converge to g^* as the size of the training set grows. More generally, stochasticity in the labels can lead to the selection of a model that both generalizes poorly and is larger than would be needed for, say, the Bayes-optimal labeling. In particular, the problem is the presence of *noise* in the dataset, where a point (x, y) is

said to be noisy if $y \neq g^*(x)$ and $\eta(x) \neq 1/2$, i.e., y is not the Bayes-optimal label for x.

B. Preprocessing as a Denoising Step

Our proposed preprocessing step has the effect of denoising the labels in the training set. We now establish this formally.

Given training data $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, we fit a classifier g_n to D and then define an *edited* training set

$$D' = \{(x, y) \in D : y = g_n(x)\}. \tag{1}$$

That is, the edited training set will contain the instances whose labels agree with the predicted labels of the classifier. Finally, we apply logic minimization to the edited data to get a set of decision rules.

In the original data D, the labels y_i can disagree with the Bayes-optimal predictions $g^*(x_i)$ on as many as half of the points, since $\eta(x_i)$ can be arbitrarily close to $1/2$. We will now see that for the edited data, the fraction is much smaller.

In interpreting the following lemma, recall that when $\eta(x) = 1/2$, either prediction (0 or 1) is Bayes-optimal. This leads to some messiness when stating results; to avoid it, we assume that none of the x_i has $\eta(x_i)$ exactly $1/2$. This holds with probability one if the decision boundary has measure zero, i.e., $\mu(\{x : \eta(x) = 1/2\}) = 0$.

Lemma 1: Fix any $x_1, \ldots, x_n \in \mathcal{X}$. Assume that $\eta(x_i) \neq 1/2$ for all of these points. Suppose each label y_i is drawn according to the conditional probability distribution $\eta(x_i)$, and let $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$. Let g_n be any classifier learned from D, and let ϵ_n denote the fraction of the points $\{x_i\}$ for which $g_n(x_i) \neq g^*(x_i)$. Finally, define the edited dataset $D' \subset D$ as in (1) above. Then,

a) $|\{(x, y) \in D' : y \neq g^*(x)\}| \leq \epsilon_n n$.
b) $\mathbb{E}[|D'|] \geq n(1/2 - \epsilon_n)$, where the expectation is over the randomness in the labels y_i.

Proof: Let B denote the set of "bad" data points x_i on which g_n disagrees with g^*

$$B = \{x_i : 1 \leq i \leq n,\ g_n(x_i) \neq g^*(x_i)\}.$$

We are given that $|B| = \epsilon_n n$.

For part (a), note that any x_i that makes it into D' has $y_i = g_n(x_i)$. Therefore, the only way that y_i could differ from $g^*(x_i)$ is if $x_i \in B$.

For part (b), if a data point (x_i, y_i) satisfies both $g_n(x_i) = g^*(x_i)$ and $y_i = g^*(x_i)$, then $y_i = g_n(x_i)$ and thus the point is included in D'. Hence

$$|D'| \geq n - \{1 \leq i \leq n : g_n(x_i) \neq g^*(x_i)\} - \{1 \leq i \leq n : y_i \neq g^*(x_i)\}.$$

The first set in this expression is B. The second set has expected size at most $n/2$, since for any i, $\Pr(y_i \neq g^*(x_i)) = \min(\eta(x_i), 1 - \eta(x_i)) \leq 1/2$. Thus, $\mathbb{E}[|D'|] \geq n - |B| - n/2 = n(1/2 - \epsilon_n)$.

In short, D' contains roughly at least half the original training points, and the fraction of faulty (non-Bayes-optimal) labels in it is at most $\epsilon_n/(1/2 - \epsilon_n) \approx 2\epsilon_n$.

Lemma 1 works for any choice of intermediate classifier g_n. We suggest taking g_n from a family of classifiers that is strongly consistent, that is, for which $\text{err}(g_n) \to R^*$ almost surely as $n \to \infty$. Under this condition, the error ϵ_n defined in the lemma goes to zero. Strong consistency is known to hold for the adaptive nearest neighbor rule [11], for boosted decision trees [12], [13], and for support vector machines with the Gaussian kernel [14].

C. Localization Properties of the Preprocessing

In data drawn from the underlying distribution P, as many as half the points x could have labels that disagree with the Bayes-optimal prediction $g^*(x)$, due to the stochasticity in the conditional probability distributions $\eta(\cdot)$. The preprocessing step selects a subset $D' \subset D$ that is not too much smaller than D and in which at most an $O(\epsilon_n)$ fraction of the labels are noisy.

However, even a small amount of noise can be troublesome if it is scattered throughout the instance space. This is because logic minimization searches for logical rules (conjunctions) that *perfectly* agree with the data, and even one noisy label could falsely invalidate a good rule.

We now show that if estimator g_n is an *adaptive nearest neighbor rule* [11], then any noisy points in D' are *localized:* they are not spread throughout $\mathcal{X}$, but lie in a region around the decision boundary, and this region shrinks as the size of the training set, n, is increased.

We begin with a brief overview of the adaptive nearest neighbor classifier. In contrast with k-nearest neighbor, which makes a prediction on a query point x by looking at its k nearest neighbors in the training set, the adaptive rule does not use a predefined choice of k. Instead, it grows k until the resulting set of training labels has a significant majority, and then predicts accordingly. If a significant majority is never achieved, then it outputs "?" (don't know). The tradeoff between accuracy and level of abstention is managed through a single confidence parameter $0 < \delta < 1$. The smaller this parameter, the higher the required level of significance; this results in more don't-knows as well as higher accuracy when a prediction is actually made.

In the terminology above, the adaptive nearest neighbor classifier produces predictions $g_n(x) \in \{0, 1, ?\}$. Our editing rule will discard any point (x_i, y_i) with $g_n(x_i) \neq y_i$; this includes any point with $g_n(x_i) = ?$.

What are the points on which g_n will fail to predict the correct label (or abstain)? It turns out that these are guaranteed to be near the decision boundary, that is, to have $\eta(x)$ close to $1/2$. The following result is a corollary of the convergence guarantees of the adaptive nearest neighbor estimator, [11, Th. 2].

Lemma 2: Suppose $\mathcal{X} \subset \mathbb{R}^d$ and η is α-Holder continuous. Let g_n denote the adaptive nearest neighbor classifier with confidence parameter $0 < \delta < 1$. Let D' be the edited training set, as defined as in Lemma 1. Then, with probability at least $1 - \delta$ (over the randomness in the original dataset), every point in D' with $y_i \neq g^*(x_i)$ has

$$\left|\eta(x) - \frac{1}{2}\right| \leq \left(\frac{C}{n} \log \frac{n}{\delta}\right)^{\beta}$$

for some constant C and $\beta = \alpha/(d + 2\alpha)$.

Algorithm 1: Denoising Algorithm.

Input: Original dataset
$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, a classifier g_n
Output: Denoised dataset $D' \subseteq D$
1: $D' = \emptyset$
2: **for** $i = 1 \ldots n$ **do**
3: **if** $y_i = g_n(x_i)$ **then**
4: $D' = D' \cup (x_i, y_i)$
5: **end if**
6: **end for**
7: return D'

Proof: We begin by defining a notion of closeness to the decision boundary. For any $\tau \geq 0$, the τ-width decision boundary is the set

$$\text{BD}(\tau) = \{x \in \mathcal{X} : |\eta(x) - 1/2| \leq \tau\}.$$

Thus, $\text{BD}(0)$ is the exact decision boundary, the set of points with $\eta(x) = 1/2$.

The adaptive nearest neighbor work introduced a notion of *margin* at each point $x \in \mathcal{X}$, called the "advantage" at x and denoted $\text{adv}(x)$ [11]. This is a value in the range [0,1] and corresponds to the statistical ease of estimating the Bayes-optimal prediction at x. Roughly speaking, points with low advantage are those near the decision boundary. Under Holder smoothness conditions on η, it can be shown by following [15, Lemma 18] that for any $q \in [0, 1]$

$$\text{adv}(x) < q \;\Rightarrow\; x \in \text{BD}(Lq^{\beta})$$

for some $L > 0$ and $\beta = \alpha/(d + 2\alpha)$.

The key convergence guarantee (see [11, Theorem 2]) of the nearest neighbor estimator g_n is that with probability $> 1 - \delta$, it will correctly classify all points with significant advantage

$$\text{adv}(x) \geq \frac{C}{n} \log \frac{n}{\delta} \;\Rightarrow\; g_n(x) = g^*(x).$$

The statement then follows by tracing the proof of Lemma 1 and observing that any mistake in D' is also a point on which g_n disagrees with g^*.

D. Details of the Denoising Step

In the abovementioned section, the definition of an edited training set is given by (1). We further elaborate on this denoising step in the pseudocode shown in Algorithm 1.

The denoised datasets generated using Algorithm 1 are guaranteed to be functions (there is only one unique output label for each input combination) even though the original dataset may be a relation (each input combination may have multiple output labels). This point can be easily derived from the fact that all classifiers g_n used for denoising are functions, and thus, the label that corresponds to the prediction of a classifier $g_n(x_i)$ is unique for input x_i, i.e., if $x_i = x_j$, then $g_n(x_i) = g_n(x_j)$. On the other hand, as noted before, in the original dataset, the same x_i may have both positive and negative labels. In that case, to derive an incompletely specified logic function, one of the labels has to be taken, for example, by taking the majority label.

IV. Experimental Evaluation

A. Evaluation Setup

1) Datasets: We performed numerical evaluations on seven publicly available tabular datasets, most of which have more than 10 000 instances and comprise categorical and numerical attributes for each instance before binarization. Among them, four are from Kaggle (churn, airline, market, and cardio), two (adult and chess) are from the UCI Machine Learning Repository [16] and the last one (retention) is from the AIX360 package [17]. For all datasets, we adopted the preprocessing approach discussed in Section II to encode categorical and numerical attributes into binarized features.

A fixed number of ten thresholds is used for all numerical features unless there exists less than ten unique values in the feature column, in which case we used the unique values as thresholds. All results in this section were obtained using the nested five-fold cross-validation that selects the best parameters for optimizing the models' performances on each partition.

2) Denoising and Logic Minimization: As discussed in Section III, we first perform a denoising step to remove noisy training samples. In particular, we experimented with three strongly consistent classifiers that are known to theoretically converge to a Bayes-optimal classifier to perform the denoising, namely adaptive nearest neighbor (AKNN) [11], support vector machines (SVM) with the Gaussian kernel [18], and gradient boosted decision trees (XGBoost) [6]. We then applied logic minimization to the denoised training datasets to derive the decision rules in DNF. For logic minimization, we used the ESPRESSO minimizer [19], which is a widely used computer program that efficiently solves the two-level logic minimization problem with iterative improvements. In our experiments, the decision rule set models derived by applying ESPRESSO to the denoised datasets are named "Denoise-A," "Denoise-S," and "Denoise-X" for AKNN, SVM, and XGBoost, respectively. We also included the results of using ESPRESSO directly on the original noisy datasets, which is named "no-denoise."

3) Baselines and Parameter Tuning: Apart from the baseline models that we used to remove the noisy training samples in the datasets, we also included the following five other classifiers: RIPPER [2], CG [5], random forest (RF) [8], classification and regression tree (CART) [20], and a deep neural network (DNN). The first two are representatives of the state-of-the-art decision rule learners, while the next two are popular machine models used on tabular datasets. We also included a neural network as another black-box model for comparison. In particular, we used a 6-layer deep fully connected neural network, with 64 neurons per layer and ReLU activation in the intermediate layers. Overall, we consider three explainable models (CART, RIPPER, and CG) and five nonexplainable models (AKNN,[2] RF, SVM,

[2] Although k-nearest neighbor is often considered as an explainable model, it is much more difficult to explain the predictions for AKNN because it potentially requires a large number of nearest neighbors (possibly hundreds) to reach sufficient confidence, in which case the explanation is not at all apparent.

TABLE III
NUMBER OF TRAINING INSTANCES FOR EACH DATASET (SIZE) AND THE PERCENTAGE OF THE NOISY INSTANCES REMOVED BY EACH CLASSIFIER FOR EACH DATASET

	adult	chess	retention	churn	airline	market	cardio
Size	24 130	22 445	8 000	5 626	20 715	36 169	56 000
Denoise-A	13.26%	2.34%	3.24%	11.77%	1.81%	6.38%	24.52%
Denoise-S	13.82%	0.10%	3.43%	18.04%	1.56%	6.18%	25.72%
Denoise-X	14.26%	3.30%	4.29%	17.00%	4.97%	9.52%	24.08%
mean	13.78%	1.91%	3.65%	15.60%	2.78%	7.36%	24.77%

The results are averaged over five partitions. The last row (mean) is the average percentage of removed noisy instances for each dataset.

XGBoost, and DNN) in our evaluations. In particular, decision trees were constructed using the CART [20] algorithm, RIPPER is an old variant of the sequential covering algorithm that greedily mines rule set from the dataset, and CG formulates the problem of learning a set of decision rule set as a mixed-integer programming problem with a loss function that captures the interpretability and accuracy of the decision rule set at the same time. Since CG cannot implicitly learn the negations of the input binarized features, the negations of the input features were appended to the datasets for CG only so that we can get the best models from it.

As stated before, all classifiers were trained with the best parameters according to the nested five-fold cross-validation. Specifically, we varied the minimum number of samples per leaf for CART and RF, the regularization term for XGBoost, the regularization parameter C for SVM, the parameter A corresponding to the confidence parameter δ as stated in the AKNN paper, and the learning rate for DNN. For rule learners, we tuned the parameters used in the actual implementations that control the complexity of the decision rule set: the maximum number of conditions and the maximum number of rules for RIPPER; the cost of each clause and the cost of each condition for CG. Since there is no parameter to be tuned for ESPRESSO, the results of no-denoise were obtained on the same training and test datasets used by other methods without any parameter tuning. Also, the parameters tuned for Denoise-A, Denoise-S and Denoise-X are exactly the same as the parameters tuned for AKNN, SVM, and XGBoost, respectively. We used the sklearn implementations [21] for RF, CART, and SVM. The implementations of other models are publicly available on GitHub.[3]

B. Classification Results on Popular Tabular Datasets

1) Denoised Datasets Statistics: The sizes (total number of data points) of the training sets and the percentages of noisy instances in the training set removed by the denoising methods, i.e., AKNN, SVM, and XGBoost, are shown in Table III.

The standard benchmarks shown are generally considered to be large and sufficiently representative, with some benchmarks containing up to 56 000 instances. The percentages of the removed instances reflect how noisy each classifier thinks about the datasets, which spans a wide range from 0.10% to 25.72%. Among the seven datasets, chess, retention, and airline comprise the least amount of noise whereas cardio has around a quarter of data points being noisy, which, as we will see later, matches the performance of logic minimization when no denoising is applied. In general, the number of removed noisy instances is relatively limited compared with the size of the dataset, which matches our theoretical expectation.

2) Improvements Over Standard Logic Minimization: As seen in the first four rows of Table IV, logic minimization with denoising techniques (Denoise-A, Denoise-S, and Denoise-X) always achieve significant improvements over standard logic minimization (no-denoise), where the latter on average shows the weakest competitiveness among all models due to the lack of consistency as a classifier. In particular, logic minimization with denoising yields an improvement in test accuracy by as much as 9% (adult, churn) comparing to its no-denoise counterpart when a clear degree of noisiness (e.g., $> 13\%$) is present in the dataset, which validates that preprocessing the dataset by removing the noisy data points is an effective method to enhance logic minimization as a machine learning model. On the other hand, no-denoise outperforms or is on par with all other explainable models (CART, RIPPER, and CG) and AKNN on the chess, retention, and airline datasets, indicating that logic minimization without any preprocessing might be a good choice for the datasets that come with low stochasticity. In both scenarios, we can always expect a performance gain by denoising the datasets first before applying logic minimization, with logic minimization benefiting more substantially from noise removal when the noise percentage is higher.

As already shown in Tables III and IV, denoising noisy datasets can significantly improve the performance of models derived from logic minimization. We further show this in Fig. 1, where we see four quadrants depicted. In the upper-right quadrant, we see that a large reduction in the denoised dataset generally correlates with a significant improvement in test accuracy. This is because a large reduction implies that the dataset is noisy, which causes detrimental overfitting problems for the logic minimizer. Therefore, logic minimization gains significant improvements by first denoising the dataset. On the other hand, we see in the lower-left quadrant that a small reduction in the denoised dataset generally correlates with a more modest improvement in test accuracy. This demonstrates a clear positive correlation between the noise ratio and the corresponding accuracy improvement after denoising.

[3]Here are the GitHub links: CG (https://github.com/Trusted-AI/AIX360); AKNN (https://github.com/b-akshay/aknn-classifier); RIPPER (https://github.com/imoscovitz/wittgenstein).

TABLE IV
TEST ACCURACY (AS A PERCENTAGE) FOR ALL CLASSIFIERS WITH STANDARD DEVIATION

Model	adult	chess	retention	churn	airline	market	cardio	mean
Explainable models and Logic minimization + denoising								
No-denoise	74.69 ±0.35	95.35 ±0.32	93.31 ±0.36	70.61 ±1.33	92.05 ±0.14	85.13 ±0.28	66.03 ±0.17	82.45
Our approach Denoise-A	82.55 ±0.49	95.32 ±0.36	93.31 ±0.44	79.51 ±1.17	93.30 ±0.47	89.24 ±0.09	72.53 ±0.36	86.54
Our approach Denoise-S	**83.62** ±0.30	**96.43** ±0.26	**94.29** ±0.16	**79.58** ±1.39	**94.45** ±0.48	**90.04** ±0.14	**73.20** ±0.43	**87.37**
Our approach Denoise-X	83.33 ±0.14	95.80 ±0.31	93.96 ±0.17	76.93 ±0.88	93.37 ±0.36	89.27 ±0.04	73.03 ±0.41	86.53
CART	82.46 ±0.35	85.51 ±0.40	89.79 ±0.65	78.81 ±0.61	91.07 ±0.43	89.58 ±0.18	72.40 ±0.33	84.23
RIPPER	82.37 ±0.56	85.80 ±0.39	88.82 ±0.36	78.27 ±0.93	93.04 ±0.43	89.45 ±0.27	70.57 ±0.30	84.05
CG	82.60 ±0.47	81.75 ±0.50	90.77 ±0.57	79.21 ±1.07	92.73 ±0.36	89.77 ±0.10	71.46 ±0.16	84.04
Non explainable models								
AKNN	82.94 ±0.45	91.09 ±0.43	91.49 ±0.41	79.15 ±1.05	91.72 ±0.45	89.38 ±0.11	71.69 ±0.32	85.35
RF	83.95 ±0.55	92.49 ±0.42	93.52 ±0.42	**80.20** ±0.82	94.69 ±0.42	89.97 ±0.16	**73.50** ±0.42	86.90
SVM	84.40 ±0.36	**97.56** ±0.35	**94.31** ±0.38	80.15 ±0.99	95.59 ±0.29	**90.42** ±0.14	73.38 ±0.44	**87.97**
XGBoost	84.50 ±0.19	95.49 ±0.36	94.30 ±0.28	78.30 ±1.00	**95.89** ±0.24	90.28 ±0.08	73.06 ±0.35	87.40
DNN	**84.61** ±0.33	96.65 ±0.65	93.81 ±0.53	75.43 ±1.83	94.75 ±0.53	90.11 ±0.15	73.38 ±0.34	86.96

The best accuracies among all classifiers and explainable classifiers are marked with orange and blue background, respectively.

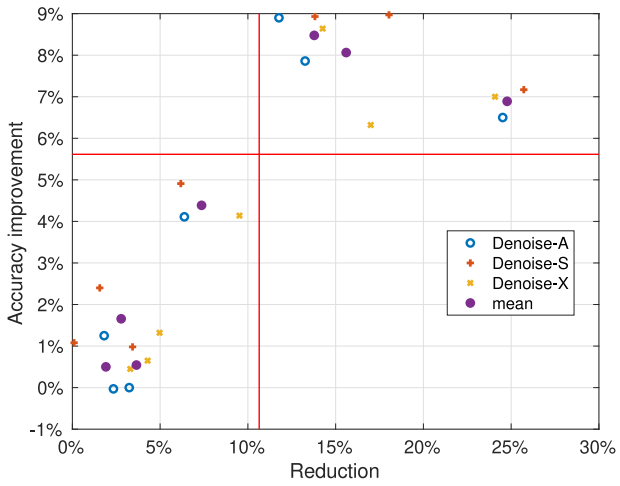


Fig. 1. Percentage of the noisy instances removed by each classifier (X-axis) versus the test accuracy improvement by applying logic minimization to the denoised dataset compared with the no-denoise results (Y-axis).

3) An Overall Better Explainable Model: The benefit of decision rule set in DNF is that the user can always extract satisfied rules to reason a decision. In comparison with other explainable models, our paradigm generates explainable decision rule sets that achieve the superior predictive performance, as can be seen in Table IV (rows 2–7). The accuracies across all datasets of Denoise-A, Denoise-S, and Denoise-X are as much as 11% better than CART, RIPPER, and CG.

4) Competitive Compared to Nonexplainable Models: Not only are the decision rule sets from logic minimization on denoised datasets dominant over other explainable models, but they are also very competitive even when compared with nonexplainable models. As can be seen by comparing the logic minimization models (rows 2–4) and the nonexplainable models (rows 8–12) in Table IV, ESPRESSO applied on the denoised datasets are very closed to the performances of the nonexplainable models with the maximum difference less than 1.5%, while still being completely explainable. This is significant because it has been generally thought that explainable models are not competitive with nonexplainable models, but our results show otherwise.

5) Decision Rule Set Without Loss of Performance: Lastly, we compare the performance of the ESPRESSO models with their corresponding classifiers that were used to remove the noisy data points in the training dataset, and the difference can be seen in Table V. Both Denoise-S and Denoise-X only decreased by less than 1%, providing further evidence that logic minimization applied to the denoised dataset can achieve comparable

TABLE V
DIFFERENCE IN TEST ACCURACY (AS A PERCENTAGE) BETWEEN THE CLASSIFIERS USED TO REMOVE NOISY INSTANCES AND THEIR CORRESPONDING LOGIC MINIMIZATION RESULTS

	adult	chess	retention	churn	airline	market	cardio	mean
Diff(AKNN)	−0.39	4.23	1.82	0.36	1.58	−0.14	0.84	1.19
Diff(SVM)	−0.78	−1.13	−0.02	−0.57	−1.14	−0.38	−0.18	−0.60
Diff(XGBoost)	−1.17	0.31	−0.34	−1.37	−2.52	−1.01	−0.03	−0.88
Diff(Best)	−0.88	−1.13	−0.02	−0.62	−1.44	−0.38	−0.30	−0.68

In the first three rows, Diff(g) = Acc(Denoise-g) − Acc(g), where g is a classifier and Acc($\cdot$) is the test accuracy for the input classifier. In the last row, Diff(Best) gives the difference between the best logic minimization result and the best result among all classifiers for each dataset. Positive numbers mean the denoised logic minimization results are better than their corresponding denoising classifiers and negative numbers mean the other way around.

performance as the corresponding denoising classifier. Moreover, Denoise-A actually outperformed AKNN by more than 1%, further indicating that logic minimization can potentially generalize better on the datasets that have low stochasticity. The last row in the table shows that the difference between logic minimization after the denoising process is very close to the state-of-the-art nonexplainable models with only less than 0.7% discrepancy in the average test accuracy.

C. Denoising Example

In this section, we provide some intuition behind what denoising is doing by means of an example. Consider again the cardiovascular disease dataset (*cardio*) from the Kaggle competition [9], where the classification task is to predict the presence of cardiovascular disease based on the patient's information. This is a large training dataset comprising 56 000 data points. As shown in Table III, this is a noisy dataset, where the denoisers removed on average 24.77% of the dataset. This correlates with the significant accuracy improvements between the logic minimization results with denoising versus no-denoising, with an average improvement of about 7% in accuracy (see Table IV). In the case of no-denoising, one of the decision rules derived by logic minimization is as follows.

IF (Height $>$ 170 cm) AND
(Weight $\leq$ 58 kg) AND
(Systolic blood pressure $>$ 130) AND
(Systolic blood pressure $>$ 140)
THEN Presence of cardiovascular disease.

However, this rule only applies to 14 patients in the original dataset, which is a relatively small number of cases in comparison to the complete training dataset of 56 000 cases (under 0.03% of the cases). This causes logic minimization to introduce many more rules that are more complex than would be needed just to *exactly fit the given dataset*.

On the other hand, after denoising the dataset (using SVM with a Gaussian kernel in this example), the above decision rule simplifies to just the following.

IF Systolic blood pressure $>$ 130
THEN Presence of cardiovascular disease.

This new rule covers 15 740 data points in the original dataset. Our denoising step removed 2603 of these data points as noise (about 16.5% of these data points). In particular, without denoising, the original rule involving height, weight, and an upper limit on the systolic blood pressure was needed to cover relatively rare cases of patients without cardiovascular disease that had systolic blood pressure above 130, for example, with weight above 58 kg or height shorter than 170 cm. These rare anomalous cases detract from the general trend that patients with systolic blood pressure above 130 overwhelmingly have cardiovascular disease. This is an example of denoising that led to significant simplification of rules and much better generalization, as evidenced by the significant improvements in test accuracy.

D. Synthetic Data Experiments to Quantify the Impact of Denoising

1) Quantifying the Impact of Noise on Logic Minimization: To quantitatively evaluate the impact of noisy data points on logic minimization, we manually generated synthetic noisy datasets based on predefined DNF rules. In particular, we consider the input space comprising 16 binarized features, which leads to about 64 000 combinations. Then, we randomly generated five rules as the predefined rules, each of which is a conjunction of three randomly chosen features. To generate a synthetic dataset, we randomly sampled p "ground-truth" data points and labeled them according to the predefined rules. Besides the ground-truth data points, we further added q noisy data points into the synthetic dataset, so that the combined dataset contains $p + q = 10\,000$ data points. To generate the noisy data points, we randomly sampled q new combinations and purposely mislabeled them with the label opposite to the predefined rules. For example, if we want to inject $q = 3000$ noisy data points, we would randomly sample $p = 7000$ ground-truth data points for a total of 10 000 data points in the synthetic dataset.

In Fig. 2, we show the impact of noise on logic minimization. In particular, as shown in Fig. 2(a), we varied the number of noisy data points in the synthetic dataset from $q = 0$ to $q = 3000$ on the X-axis (with a corresponding $p = 10\,000$ to $p = 7000$ ground-truth data points). On the Y-axis, Fig. 2(a) shows the corresponding test accuracies of both no-denoise and Denoise-S with the parameter C for SVM fixed to be 1. The test accuracies are based on sampling another 10 000 test instances labeled according to the predefined rules. As expected, as we

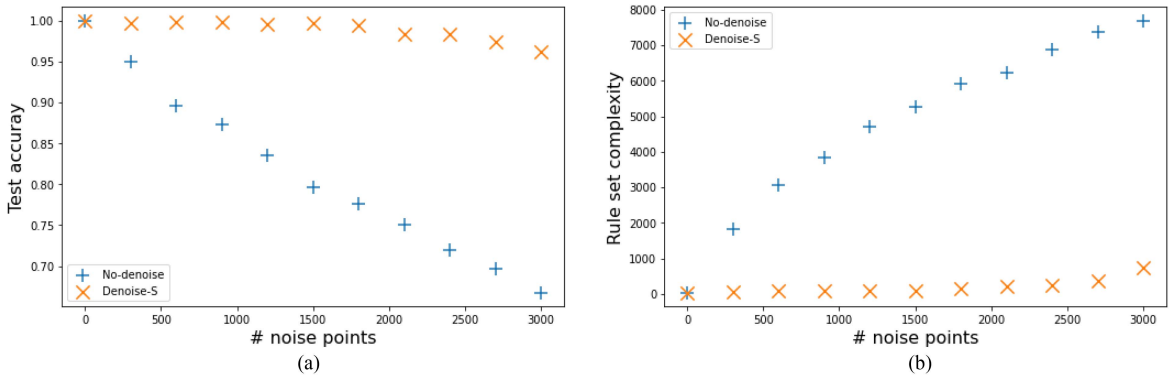


Fig. 2. Impact of noise on logic minimization on synthetic data. (a) Impact on test accuracy with increasing noise. (b) Impact on model complexity with increasing noise.

increase the noise percentage from 0% to 30%, the test accuracy of no-denoise decreases linearly, while the performance of Denoise-S remains relatively the same near 100%. This shows that no-denoise poorly generalizes with increasing amount of noise, whereas Denoise-S is robust to noisy samples.

In Fig. 2(b), we show the complexities of the decision rule sets derived from both no-denoise and Denoise-S, where the complexity of the rule set is calculated by summing the number of features across all rules and the number of rules. As we increase the noise percentage on the X-axis, the complexity of the derived rule set from no-denoise increases dramatically due to the overfitting of the noisy samples. On the other hand, the rule set complexity of Denoise-S remains at a minimum level, verifying that a classifier with provable Bayes-optimal convergence properties can successfully remove most of the noise in the training set so that logic minimization can uncover the underlying distribution, which in this case is the predefined rules.

2) Quantifying the Performance of Denoisers: We next evaluate the effectiveness of different denoisers in identifying noise in datasets. To quantitatively evaluate this, we again manually produced synthetic noisy datasets using the same randomly generated five rules as the predefined rules, as in the previous section. In this experiment, we again generated synthetic datasets with 10 000 data points, but this time, we randomly sampled 9000 "ground-truth" data points labeled according to the predefined rules. We then injected two types of noisy points into the dataset. The first type of noise (Noise 1) is the same type of noise in the previous section: we randomly sampled new combinations and purposely mislabeled them with the label opposite to the predefined rules. We also injected a second type of noise (Noise 2): we randomly selected some combinations among the 9000 ground-truth combinations and assigned the opposite labels to them. In other words, these combinations have both positive and negative labels in the synthetic dataset. We added in total 1000 noisy points to the datasets with different ratios of the two types. We then applied the denoisers to the datasets consisting of $9000 + 1000 = 10\,000$ points and let them identify the noisy data points. The results are shown in Table VI.

In particular, in all cases with different compositions of noise types, SVM always perfectly removes all noisy points without mistakenly removing any correct data, which is consistent with the results shown in Table IV where Denoise-S always achieves the best accuracy. With respect to XGB and AKNN, both are able to correctly identify most of the noisy data points, with XGB successfully finding about 79%–99% of the noisy points, and AKNN successfully finding about 90%–95% of the noisy points. Although XGB incorrectly removes about 0.1%–0.2% of the correct data out of the 9000 points, and AKNN incorrectly removes about 1.5% of the correct data, the percentage of incorrectly removed data points is negligibly small in both cases relative to the total number of clean points. Overall, the amount of correctly identified noise is substantially greater than the missed and incorrectly-removed points, which meets our expectation: the denoisers remove almost all noisy points and not too many of the clean points.

TABLE VI
DENOISING STATISTICS ON SYNTHETIC DATASETS WITH FIVE RULES, EACH AS A CONJUNCTION OF THREE RANDOMLY CHOSEN FEATURES

Noise 1	Noise 2	Denoiser	FP	TP1	TP2	Total
500	500	XGB	15	490	412	917
		SVM	0	500	500	1000
		AKNN	150	487	464	1101
1000	0	XGB	14	791	0	805
		SVM	0	1000	0	1000
		AKNN	169	910	0	1079
0	1000	XGB	13	0	987	1000
		SVM	0	0	1000	1000
		AKNN	152	0	980	1132

Noise 1 and noise 2 correspond to the amounts of the first and second types of noise, respectively. FP and TP stand for false positive and true positive, where FP is the number of points that are not noise but removed by the denoisers, while TP1 and TP2 are the number of noise 1 and noise 2 points identified by the denoisers, respectively. Total is the total number of instances removed by the denoisers.

E. Impact of Denoising on Other Explainable Models

Throughout this article, we extensively discussed the importance of denoising when logic minimization is used to derive

TABLE VII
AVERAGE IMPROVEMENT (AS A PERCENTAGE) IN TEST ACCURACY BY DERIVING OTHER MODELS ON THE DENOISED DATASETS

	ESPRESSO	CART	RIPPER	CG	SVM	XGBoost
Denoise-A	**4.09**	−0.09	1.09	0.07	−2.10	−0.60
Denoise-S	**4.92**	0.21	1.42	0.48	−1.43	−0.50
Denoise-X	**4.08**	−0.25	1.31	0.28	−0.67	−0.25

The best improvements for each denoising method across all models are marked in bold.

decision rule sets from tabular datasets since logic minimization exactly fits all data points. In this section, we also evaluate how denoising affects other models. The results are summarized in Table VII.

The average improvements of applying logic minimization to the denoised datasets are significant, over 4%. This is shown under the column labeled ESPRESSO. It can be seen that denoising also generally improves the test accuracies of CART, RIPPER, and CG. However, the improvements are relatively small in comparison with logic minimization, under 0.25%, 1.5%, and 0.5% for CART, RIPPER, and CG, respectively. The reason why the improvements are much smaller with these methods may be due to the fact that these learners are already somewhat noise-tolerant: they tend to underfit data in favor of simple models and are, therefore, less affected by noisy points, but this comes at the price of inferior performance. Furthermore, we observe that denoising can have a small negative impact on the performances of SVM and XGBoost, which is not suprising as both SVM and XGBoost already have good generalization capabilities without denoising. On the other hand, because logic minimization tends to overfit, it can greatly benefit from our denoising framework, as shown in Table VII.

V. RELATED WORK

Satisfiability-based (SAT-based) logic minimization has been proposed before [22], [23] to derive minimized DNF formula that translate to decision sets. However, these methods *assume there is no inconsistency* in the training data, meaning that there are no overlapping examples that have different class labels. They resolve inconsistencies by taking the majority class such that the largest consistent subset of nonoverlapping training instances is retained. In turn, these SAT-based methods act as a logic minimizer to *exactly fit* the resulting dataset. As explained throughout this article, logic minimization performed this way (corresponding to the "no-denoise" case in our paper) often fail to produce accurate decision models because the resulting dataset often still contains noisy points that lead to detrimental overfitting and poor generalization. How best to remove training instances to create a consistent dataset is highly nontrivial, which is precisely our key contribution: a novel theoretically grounded denoising framework that substantially improves the performance of two-level logic minimization in the learning of accurate decision sets, not simply the use of logic minimization (whether ESPRESSO [10] or a SAT-based approach) to derive decision sets.

The *consistency* of learning algorithms is a central question in statistics and machine learning. For *parametric* classifiers, such as linear separators, the desired outcome is convergence to the best model in the function class as the number of training points grows. This typically depends upon the boundedness of some complexity measure, such as Vapnik–Chervonenkis (VC) dimension [24] and holds broadly. However, such function classes might not be rich enough to capture all the intricacies of the underlying classification task. For *nonparametric* models, it is possible to hope for better: convergence to the Bayes-optimal model. This has been established for various popular methods, including k-nearest neighbor (with suitably growing k) [25], [26], boosting with certain base classes [12], and families of kernel machines including the support vector machine with RBF kernel [14].

There is also existing literature devoted to label noise, focusing mainly on unreliable or erroneous labels. Frénay and Verleysen [27] provided an extensive analysis of label noise and the potential problems that they can cause in classification problems and reviews the existing literature on algorithms for filtering erroneously labeled instances. For example, Thongkam et al. [28] proposed to use SVM as the filterng method to identify misclassified instances in a breast cancer survivability dataset. Jeatrakul et al. [29] proposed to use neural networks to detect misclassification patterns in the training data. Northcutt et al. [30] introduced a model-agnostic denoising framework that identifies the noisy samples using the out-of-sample predicted probabilities of the training instances by a user-specified classifier. However, these works either do not provide insights into why their choice of classifiers will work in the given scenarios or do not provide guidance regarding the choice of a user-specified classifier. In contrast, our denoising framework takes a different view of the problem by assuming that the true labels of the training instances are Bayes-optimal labels. This assumption provides us with guidance on what classifiers to select to give an accurate estimate of the underlying true labels, which is grounded in theory, as discussed in Section III. That is, our denoising framework can in theory correctly identify all noisy instances since we use classifiers with provable Bayes optimal convergence properties to identify noise in the dataset. This is not case, for example, in prior works like [29], [30]. To the best of authors' knowledge, our work is the first to synthesize the ideas of logic minimization, convergence to the Bayes-optimal model for ensuring consistency, and label noise removal to simultaneously achieve the state-of-the-art classification performance in tabular learning problems with explainable rule-based predictions.

Finally, the learning of rule sets has received considerable attention due to their ability to provide human-understandable explanations. Contrary to greedy rule mining methods developed before [2], [31], recently proposed methods [3], [4], [5] explicitly consider the tradeoff between the explanation complexity and the predictive performance and aim to get

the best training accuracy under a certain complexity constraint. However, there is still a noticeable performance discrepancy between decision rule sets and black-box models, such as RF [8] and gradient boosted trees [6] even if there is no constraint on the complexity of the rule set. Although our work also generates human-understandable IF–THEN rules, and therefore falls into the same category of decision rule set learning, we show that our method significantly improves on modern decision rule learners and bridges the gap with black-box models.

VI. Conclusion

In this article, we explore the use of two-level logic minimization as a machine learning paradigm for tabular datasets. Although tabular datasets can be viewed as logic functions that can be simplified with two-level logic minimization to derive minimal logic formulas in DNF, this has not been a successful approach in the past, leading to complex models with poor generalizations. Our conjecture is that these problems are caused by the presence of noisy instances in the training data. Because logic minimization exactly fits all data points, these noisy instances can lead to detrimental overfitting problems, leading to models that both generalize poorly and are far more complex than necessary. We propose a statistical framework for denoising the training data, corresponding to the removal of noisy data points that have anomalous labels or are close to the decision boundary. This denoising approach allows logic minimization to be effective in deriving simple DNF formulas that have good generalization properties. The DNF formulas can in turn be readily converted to explainable decision rules. Using this approach, we are able to obtain comparable performance as gradient boosted and ensemble decision trees, which have been the winning hypothesis classes in tabular data learning competitions, but with human understandable explanations in the form of decision rules. We hope our successful results will open the door to further fruitful research in the underexplored area of logic minimization as a viable machine learning direction.

References

[1] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Mach. Intell.*, vol. 1, pp. 206–215, 2019.

[2] W. W. Cohen, "Fast effective rule induction," in *Proc. 12th Int. Conf. Mach. Learn.*, 1995, pp. 115–123.

[3] H. Lakkaraju, S. H. Bach, and J. Leskovec, "Interpretable decision sets: A joint framework for description and prediction," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 1675–1684, doi: 10.1145/2939672.2939874.

[4] T. Wang, C. Rudin, F. Doshi-Velez, Y. Liu, E. Klampfl, and P. MacNeille, "A Bayesian framework for learning rule sets for interpretable classification," *J. Mach. Learn. Res.*, vol. 18, no. 70, pp. 1–37, 2017. [Online]. Available: http://jmlr.org/papers/v18/16-003.html

[5] S. Dash, O. Günlük, and D. Wei, "Boolean decision rules via column generation," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 4660–4670.

[6] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 785–794.

[7] G. Ke et al., "LightGBM: A highly efficient gradient boosting decision tree," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, vol. 30, pp. 3146–3154.

[8] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001, doi: 10.1023/A:1010933404324.

[9] S. Ulianova, "Cardiovascular disease dataset," Jan. 2019. Accessed: Sep. 20, 2021. [Online]. Available: https://www.kaggle.com/sulianova/cardiovascular-disease-dataset

[10] R. K. Brayton, G. D. Hachtel, C. McMullen, and A. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*, vol. 2. Berlin, Germany: Springer, 1984.

[11] A. Balsubramani, S. Dasgupta, Y. Freund, and S. Moran, "An adaptive nearest neighbor rule for classification," 2019. [Online]. Available: https://arxiv.org/abs/1905.12717

[12] T. Zhang and B. Yu, "Boosting with early stopping: Convergence and consistency," *Ann. Statist.*, vol. 33, no. 4, pp. 1538–1579, 2005.

[13] G. Biau and B. Cadre, "Optimization by gradient boosting," *Advances in Contemporary Statistics and Econometrics: Festschrift in Honor of Christine Thomas-Agnan*, A. Daouia and A. Ruiz-Gazen Eds. Cham, Switzerland: Springer, 2021, pp. 23–44.

[14] I. Steinwart, "Consistency of support vector machines and other regularized kernel classifiers," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 128–142, Jan. 2005.

[15] K. Chaudhuri and S. Dasgupta, "Rates of convergence for nearest neighbor classification," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, Cambridge, MA, USA, 2014, vol. 51, pp. 3437–3445.

[16] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[17] V. Arya et al., "One explanation does not fit all: A toolkit and taxonomy of AI explainability techniques," Sep. 2019. [Online]. Available: https://arxiv.org/abs/1909.03012

[18] N. Cristianini et al., *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[19] R. L. Rudell and A. Sangiovanni-Vincentelli, "Multiple-valued minimization for PLA optimization," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. CAD-6, no. 5, pp. 727–750, Sep. 1987.

[20] L. Breiman, J. Friedman, C. Stone, and R. Olshen, *Classification and Regression Trees (The Wadsworth and Brooks-Cole Statistics-Probability)*. New York, NY, USA: Taylor and Francis, 1984. [Online]. Available: https://books.google.com/books?id=JwQx-WOmSyQC

[21] F. Pedregosa et al., "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[22] A. Ignatiev, E. Lam, P. J. Stuckey, and J. Marques-Silva, "A scalable two stage approach to computing optimal decision sets," *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 5, pp. 3806–3814, May 2021. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/16498

[23] A. Ignatiev, F. Pereira, N. Narodytska, and J. Marques-Silva, "A sat-based approach to learn explainable decision sets," in *Proc. Int. Joint Conf. Automated Reasoning*, 2018, pp. 627–645.

[24] V. Vapnik and A. Chervonenkis, "On the uniform convergence of relative frequencies of events to their probabilities," *Theory of Probability Appl.*, vol. 16, pp. 264–280, 1971.

[25] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 1, pp. 21–27, Jan. 1967.

[26] C. Stone, "Consistent nonparametric regression," *Ann. Statist.*, vol. 5, pp. 595–645, 1977.

[27] B. Frénay and M. Verleysen, "Classification in the presence of label noise: A survey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 5, pp. 845–869, May 2014.

[28] J. Thongkam, G. Xu, Y. Zhang, and F. Huang, "Support vector machine for outlier detection in breast cancer survivability prediction," in *Proc. Asia-Pacific Web Conf. Workshops*, 2008, pp. 99–109.

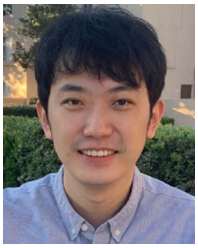
[29] P. Jeatrakul, K. Wong, and C. Fung, "Using misclassification analysis for data cleaning," in *Proc. Int. Workshop Adv. Comput. Intell. Intell. Inform.*, 2009. [Online]. Available: https://researchrepository.murdoch.edu.au/id/eprint/6637/

[30] C. G. Northcutt, L. Jiang, and I. L. Chuang, "Confident learning: Estimating uncertainty in dataset labels," *J. Artif. Intell. Res.*, vol. 70, pp. 1373–1411, 2021.

[31] B. Liu, W. Hsu, and Y. Ma, "Integrating classification and association rule mining," in *Proc. 4th Int. Conf. Knowl. Discov. Data Mining*, 1998, pp. 80–86.

Litao Qiao received the B.S. and M.S. degrees in computer and science and engineering in 2019 and 2020, respectively, from the University of California, San Diego, La Jolla, CA, USA, where he is currently working toward the Ph.D. degree in electrical and computer engineering.

His research interest is data science, explainable machine learning, and deep learning.



Weijia Wang received the B.S. degree in electrical engineering from Zhejiang University, Hangzhou, China, in 2016, and the M.S. degree in electrical and computer engineering in 2018 from the University of California, San Diego, La Jolla, CA, USA, where he is currently working toward the Ph.D. degree in electrical and computer engineering.

His research interest is machine learning and deep learning, including the compression and acceleration of deep convolutional neural networks, algorithms of meta-learning and federated learning, and explainable artificial intelligence.

Sanjoy Dasgupta received the A.B. degree from Harvard University, Cambridge, MA, USA, in 1993 and the Ph.D. degree from the University of California, Berkeley, Berkeley, CA, USA, in 2000, both in computer science.

He spent two years with AT&T Research Labs, Florham Park, NJ, USA, before joining the University of California, San Diego, La Jolla, CA, USA, where he is currently a Professor of Computer Science and Engineering. His area of research is algorithmic statistics, with a focus on unsupervised and minimally supervised learning. He is the author of a textbook, Algorithms (with Christos Papadimitriou and Umesh Vazirani), which appeared in 2006.

Dr. Dasgupta was a Program Co-Chair of the Conference on Learning Theory in 2009 and of the International Conference on Machine Learning in 2013.

Bill Lin (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer sciences from the University of California at Berkeley, Berkeley, CA, USA, in 1985, 1988, and 1991, respectively.

He is currently a Professor in electrical and computer engineering with the University of California, San Diego, La Jolla, CA, USA, where he is actively involved with the Center for Wireless Communications (CWC), the Center for Networked Systems (CNS), and the Qualcomm Institute in industry-sponsored research efforts.

Dr. Lin's research has led to more than 200 journal and conference publications, including a number of Best Paper awards and nominations. He also holds five awarded patents. He was the General Chair and on the executive and technical program committee of many IEEE and ACM conferences, and he was an Associate or Guest Editor for several IEEE and ACM journals as well.