Journal of Artificial Intelligence Research 74 (2022) 1823-1863 Submitted 11/2021; published 08/2022

Efficient Learning of Interpretable Classification Rules

Bishwamittra Ghosh BGHOSH@U.NUS.EDU
National University of Singapore

Dmitry Malioutov DMAL@ALUM.MIT.EDU

Kuldeep S. Meel MEEL@COMP.NUS.EDU.SG
National University of Singapore

Abstract

Machine learning has become omnipresent with applications in various safety-critical domains such as medical, law, and transportation. In these domains, high-stake decisions provided by machine learning necessitate researchers to design *interpretable* models, where the prediction is understandable to a human. In interpretable machine learning, *rule-based classifiers* are particularly effective in representing the decision boundary through a set of rules comprising input features. Examples of such classifiers include decision trees, decision lists, and decision sets. The interpretability of rule-based classifiers is in general related to the size of the rules, where smaller rules are considered more interpretable. To learn such a classifier, the brute-force direct approach is to consider an optimization problem that tries to learn the smallest classification rule that has close to maximum accuracy. This optimization problem is computationally intractable due to its combinatorial nature and thus, the problem is not scalable in large datasets. To this end, in this paper we study the triangular relationship among the accuracy, interpretability, and scalability of learning rule-based classifiers.

The contribution of this paper is an interpretable learning framework IMLI, that is based on maximum satisfiability (MaxSAT) for synthesizing classification rules expressible in proposition logic. IMLI considers a joint objective function to optimize the accuracy and the interpretability of classification rules and learns an optimal rule by solving an appropriately designed MaxSAT query. Despite the progress of MaxSAT solving in the last decade, the straightforward MaxSAT-based solution cannot scale to practical classification datasets containing thousands to millions of samples. Therefore, we incorporate an efficient incremental learning technique inside the MaxSAT formulation by integrating mini-batch learning and iterative rule-learning. The resulting framework learns a classifier by iteratively covering the training data, wherein in each iteration, it solves a sequence of smaller MaxSAT queries corresponding to each mini-batch. In our experiments, IMLI achieves the best balance among prediction accuracy, interpretability, and scalability. For instance, IMLI attains a competitive prediction accuracy and interpretability w.r.t. existing interpretable classifiers and demonstrates impressive scalability on large datasets where both interpretable and non-interpretable classifiers fail. As an application, we deploy IMLI in learning popular interpretable classifiers such as decision lists and decision sets. The source code is available at `https://github.com/meelgroup/mlic`.

1. Introduction

Machine learning models are presently deployed in safety-critical and high-stake decision-making arising in medical (Erickson et al., 2017; Kaissis et al., 2020; Kononenko, 2001), law (Kumar et al., 2018; Surden, 2014), and transportation (Peled et al., 2019; Zantalis

et al., 2019). In safety-critical domains, a surge of interest has been observed in designing interpretable, robust, and fair models instead of merely focusing on an accuracy-centric learning objective (Bhagoji et al., 2018; Doshi-Velez & Kim, 2017; Du et al., 2019; Hancox-Li, 2020; Holstein et al., 2019; Mehrabi et al., 2021; Molnar, 2020; Murdoch et al., 2019). In this study, our focus is on learning an interpretable machine learning classifier (Rudin, 2019), where the rules governing the prediction are explicit (by design) in contrast to black-box models.

In order to achieve the interpretability of predictions, decision functions in the form of classification rules such as decision trees, decision lists, decision sets, etc. are particularly effective (Bessiere et al., 2009; Dash & Goncalves, 2021; Ignatiev et al., 2021b; Izza et al., 2020; Lakkaraju et al., 2017, 2016; Letham et al., 2015; Narodytska et al., 2018; Rivest, 1987; Wang & Rudin, 2015; Yu et al., 2020a). Such models are used to either learn an interpretable model from the start or as proxies to provide post-hoc explanations of pre-trained black-box models (Gill et al., 2020; Lundberg & Lee, 2017; Moradi & Samwald, 2021; Ribeiro et al., 2016; Slack et al., 2020). At this point, it is important to acknowledge that interpretability is an informal notion, and formalizing it in full generality is challenging. In our context of rule-based classifiers, we will use the *sparsity of rules* (that is, fewer rules each having fewer Boolean literals), which has been considered a proxy of interpretability in various domains, specifically in the medical domain (Gage et al., 2001; Lakkaraju et al., 2019; Letham et al., 2015; Malioutov & Varshney, 2013; Myers, 1962).

In this paper, we study decision functions expressible in *propositional logic*. In particular, we focus on learning logical formulas from data as classifiers and define interpretability in terms of the number of Boolean literals that the formula contains. In the following, we illustrate an interpretable classifier that decides if a tumor cell is malignant or benign based on different features of tumors.

A tumor is malignant if
[(compactness SE < 0.1) **OR** $\neg(0.1 \leq$ concave points $< 0.2)$] **AND**
$[\neg(0.2 \leq$ area $< 0.3)$ **OR** $\neg(0.1 \leq$ largest symmetry $< 0.2)]$

Example 1.1. *We learn an interpretable classification rule in Conjunctive Normal Form for predicting the classification of a tumor cell. We consider WDBC (Wisconsin Diagnostic Breast Cancer) dataset (Agarap, 2018) to learn a CNF formula with two* clauses *and four Boolean* literals*. In the formula, clauses are connected by Boolean 'AND', where each clause contains* literals *connected by Boolean 'OR'. Informally, a tumor cell is malignant if at least one literal in each clause becomes true.*

The problem of learning rule-based classifiers is known to be computationally intractable. The earliest tractable approaches for classifiers such as decision trees and decision lists relied on heuristically chosen objective functions and greedy algorithmic techniques (Clark & Niblett, 1989; Cohen & Singer, 1999; Quinlan, 1993). In these approaches, the size of rules is controlled by early stopping, ad-hoc rule pruning, etc. In recent approaches, the interpretable classification problem is reduced to an optimization problem, where the accuracy and the sparsity of rules are optimized jointly (Lakkaraju et al., 2016; Narodytska et al., 2018). Different optimization solvers such as linear programming (Malioutov &

Varshney, 2013), sub-modular optimizations (Lakkaraju et al., 2016), and Bayesian optimizations (Letham et al., 2015) are then deployed to find the best classifier with maximum accuracy and minimum rule-size. In our study, we propose an alternate optimization approach that fits particularly well to rule-learning problems. Particularly, we propose a *maximum satisfiability* (MaxSAT) solution for learning interpretable rule-based classifiers.

The MaxSAT problem is an optimization analog to the satisfiability (SAT) problem, which is complete for the class FP^{NP}. Although the MaxSAT problem is NP-hard, dramatic progress has been made in designing solvers that can handle large-scale problems arising in practice. This has encouraged researchers to reduce several optimization problems into MaxSAT such as optimal planning (Robinson et al., 2010), automotive configuration (Walter et al., 2013), data analysis in machine learning (Berg et al., 2019), and automatic test pattern generation for cancer therapy (Lin & Khatri, 2012).

Contributions. The primary contribution of this paper is a MaxSAT-based formulation, called IMLI, for learning interpretable classification rules expressible in propositional logic. For the simplicity of exposition, our initial focus is on learning classifiers expressible in CNF (Conjunctive Normal Form) formulas; we later discuss how the CNF learning formulation can be extended to popular interpretable classifiers such as decision lists and decision sets. In this paper, IMLI provides precise control to jointly optimize the accuracy and the size of classification rules. IMLI constructs and solves a MaxSAT query of $\mathcal{O}(kn)$ size (number of clauses) to learn a k-clause CNF formula on a dataset containing n samples. Naturally, the naïve formulation cannot scale to large values of n and k. To scale IMLI to large datasets, we propose an incremental learning technique along with the MaxSAT formulation. Our incremental learning is an integration of mini-batch learning and iterative rule-learning, which are studied separately in classical learning problems. In the presented incremental approach, IMLI learns a k-clause CNF formula using an iterative separate-and-conquer algorithm, where in each iteration, a single clause is learned by covering a part of the training data. Furthermore, to efficiently learn a single clause in a CNF, IMLI relies on mini-batch learning, where it solves a sequence of smaller MaxSAT queries corresponding to each mini-batch.

In our experimental evaluations, IMLI demonstrates the best balance among prediction accuracy, interpretability, and scalability in learning classification rules. In particular, IMLI achieves competitive prediction accuracy and interpretability w.r.t. state-of-the-art interpretable classifiers. Besides, IMLI achieves impressive scalability by classifying datasets even with 1 million samples, wherein existing classifiers, including those of the non-interpretable ones, either fail to scale or achieve poor prediction accuracy. Finally, as an application, we deploy IMLI in learning interpretable classifiers such as decision lists and decision sets.

The presented framework IMLI extends our prior papers (Ghosh & Meel, 2019; Malioutov & Meel, 2018) in the following aspects. Malioutov and Meel (2018) presented a MaxSAT-based formulation for interpretable classification rule-learning, which jointly optimizes between the accuracy and size of rules. Ghosh and Meel (2019) improved the MaxSAT formulation by unifying mini-batch learning. This unification allows learning on moderate-size datasets (about 50 thousand samples). This paper takes a significant step in terms of scalability by integrating both iterative learning and mini-batch learning inside the incremental

technique. The result is a practical scalable learning framework that can classify datasets with 1 million samples and can potentially scale beyond that.

We organize the rest of the paper as follows. We discuss related literature in Section 2 and preliminaries in Section 3. In Section 4, we formally define our learning problem and provide a MaxSAT-based solution in Section 5. We improve the MaxSAT-based formulation through an incremental learning technique in Section 6. In Section 7, we apply IMLI in learning different interpretable classifiers. We then discuss our experimental results in Section 8 and conclude in Section 9.

2. Related Work

The progress in designing interpretable rule-based classifiers finds its root in the development of decision trees (Bessiere et al., 2009; Quinlan, 1986, 1987), decision lists (Rivest, 1987), classification rules (Cohen, 1995) etc. In early works, the focus was to improve the efficiency and scalability of the model rather than designing models that are interpretable. For example, decision rule approaches such as C4.5 rules (Quinlan, 1993), CN2 (Clark & Niblett, 1989), RIPPER (Cohen, 1995), and SLIPPER (Cohen & Singer, 1999) rely on heuristic-based branch pruning and ad-hoc local criteria e.g., maximizing information gain, coverage, etc.

Recently, several optimization frameworks have been proposed for interpretable classification, where both accuracy and rule-size are optimized during training. For example, Malioutov and Varshney (2013) proposed exact learning of rule-based classifiers based on Boolean compressed sensing using a linear programming formulation. Su, Wei, Varshney, and Malioutov (2016) presented two-level Boolean rules, where the trade-off between classification accuracy and interpretability is studied. In their work, the Hamming loss is used to characterize prediction accuracy, and sparsity is used to characterize the interpretability of rules. Wang and Rudin (2015) proposed a Bayesian optimization framework for learning falling rule lists, which is an ordered list of if-then rules. Other similar approaches based on Bayesian analysis for learning classification rules are (Letham et al., 2015; Wang et al., 2017). Building on custom discrete optimization techniques, Angelino, Larus-Stone, Alabi, Seltzer, and Rudin (2017) proposed an optimal learning technique for decision lists using a branch-and-bound algorithm. In a separate study, Lakkaraju et al. (2016) highlighted the importance of decision sets over decision lists in terms of interpretability and considered a sub-modular optimization problem for learning a near-optimal solution for decision sets. Our proposed method for interpretable classification, however, relies on the improvement in formal methods over the decades, particularly the efficient CDCL-based solution for satisfiability (SAT) problems (Silva & Sakallah, 2003).

Formal methods, particularly SAT and its variants, have been deployed in interpretable classification recently. In the context of learning decision trees, SAT and MaxSAT-based solutions are proposed by (Alos et al., 2021; Janota & Morgado, 2020; Narodytska et al., 2018; Shati et al., 2021). In addition, researchers have applied SAT for learning explainable decision sets (Ignatiev et al., 2018, 2021a; Schidler & Szeider, 2021; Yu et al., 2020b). In most cases, SAT/MaxSAT solutions are not sufficient in solving large-scale classification tasks because of the NP-hardness of the underlying problem. This observation motivates us in combining MaxSAT with more practical algorithms such as incremental learning.

Incremental learning has been studied in improving the scalability of learning problems, where data is processed in parts and results are combined to use lower computation overhead. In case of non-interpretable classifiers such as SVM, several solutions adopting incremental learning are available (Syed et al., 1999; Ruping, 2001). For example, Cauwenberghs and Poggio (2001) proposed an online recursive algorithm for SVM that learns one support-vector at a time. Based on radial basis kernel function, Ralaivola and d'Alché Buc (2001) proposed a local incremental learning algorithm for SVM. In the context of deep neural networks, stochastic gradient descent is a well-known convex optimization technique—a variant of which includes computing the gradient on mini-batches (Hinton et al., 2012; Li et al., 2014; Masters & Luschi, 2018). Federated learning, on the other hand, decentralizes training on multiple local nodes based on local data samples with only exchanging learned parameters to construct a global model in a central node (Konečnỳ et al., 2015, 2016). Another notable technique is Lagrangian relaxation that decomposes the original problem into several sub-problems, assigns Lagrangian multipliers to make sure that sub-problems agree, and iterates by solving sub-problems and adjusting weights based on disagreements (Fisher, 1981; Johnson et al., 2007; Lemaréchal, 2001). To the best of our knowledge, our method is the first method that unifies incremental learning with MaxSAT based formulation to improve the scalability of learning rule-based classifiers.

Classifiers that are interpretable by design can be applied to improve the explainability of complex black-box machine learning classifiers. There is rich literature on extracting decompositional and pedagogical rules from non-linear classifiers such as support vector machines (Barakat & Diederich, 2004, 2005; Diederich, 2008; Martens et al., 2008; Núñez et al., 2002) and neural networks (Augasta & Kathirvalavakumar, 2012; Hailesilassie, 2016; Setiono & Liu, 1995; Sato & Tsukimoto, 2001; Zilke et al., 2016; Zhou, 2004). In recent years, local model-agnostic approaches for explaining black-box classifiers are proposed by learning surrogate simpler classifiers such as rule-based classifiers (Guidotti et al., 2018; Pastor & Baralis, 2019; Rajapaksha et al., 2020; Ribeiro et al., 2018). The core idea in local approaches is to use a rule-learner that can classify synthetically generated neighboring samples with class labels provided by the black-box classifier. To this end, our framework IMLI can be directly deployed as an efficient rule-learner and can explain the inner-working of black-box classifiers by generating interpretable rules.

3. Preliminaries

In this section, we first discuss propositional satisfiability and MaxSAT, followed by rule-based classification in machine learning. In the paper, we use capital boldface letters such as $\mathbf{X}$ to denote matrices while lower boldface letters $\mathbf{x}$ are reserved for vectors/sets.

3.1 Propositional Satisfiability

Let F be a Boolean formula defined over Boolean variables $\mathbf{b} = \{b_1, b_2, \ldots, b_m\}$. A literal v is a variable b or its complement $\neg b$, and a clause C is a disjunction ($\vee$) or a conjunction ($\wedge$) of literals.[1] F is in Conjunctive Normal Form (CNF) if $F \triangleq \bigwedge_i C_i$ is a conjunction

1. While *term* is reserved to denote disjunction of literals in the literature, we use clause for both disjunction and conjunction.

of clauses where each clause $C_i \triangleq \bigvee_j v_j$ is a disjunction of literals. In contrast, F is in Disjunctive Normal Form (DNF) if $F \triangleq \bigvee_i C_i$ is a disjunction of clauses where each clause $C_i \triangleq \bigwedge_j v_j$ is a conjunction of literals. We use σ to denote an assignment of variables in $\mathbf{b}$ where $\sigma(b_i) \in \{\text{true}, \text{false}\}$. A *satisfying assignment* σ^* of F is an assignment that evaluates F to true and is denoted by $\sigma^* \models F$. The propositional satisfiability (SAT) problem finds a satisfying assignment σ^* to a CNF formula F such that $\forall i, \sigma^* \models C_i$, wherein $\sigma^* \models C_i$ if and only if $\exists v_j \in C_i, \sigma^*(v_j) = \text{true}$. Informally, σ^* satisfies at least one literal in each clause of a CNF formula.

In this paper, we use true as 1 and false as 0 interchangeably. Between two vectors $\mathbf{u}$ and $\mathbf{v}$ of the same length defined over Boolean variables or constants (such as 0 and 1), we define $\mathbf{u} \circ \mathbf{v}$ to refer to their inner product. Formally, $\mathbf{u} \circ \mathbf{v} \triangleq \bigvee_i (u_i \wedge v_i)$ is a disjunction of element-wise conjunction where u_i and v_i denote the i^{th} variable/constant of $\mathbf{u}$ and $\mathbf{v}$, respectively. In this context, the conjunction "$\wedge$" between a variable and a constant follows the standard interpretation: $b \wedge 0 = 0$ and $b \wedge 1 = b$. For example, if $\mathbf{u} = [u_1, u_2, u_3]$ and $\mathbf{v} = [0, 1, 1]$, then $\mathbf{u} \circ \mathbf{v} = u_2 \vee u_3$.

3.2 MaxSAT

The MaxSAT problem is an optimization analog to the SAT problem that finds an optimal assignment satisfying the maximum number of clauses in a CNF. In this work, we consider a *weighted* variant of the MaxSAT problem—more specifically, a weighted-partial MaxSAT problem—that optimizes over a set of hard and soft constraints in the form of a weighted-CNF formula. In a weighted-CNF formula, a weight $\mathsf{wt}(C_i) \in \mathbb{R}^{\geq 0} \cup \{\infty\}$ is defined over each clause C_i, wherein C_i is called a *hard* clause if $\mathsf{wt}(C_i) = \infty$, and C_i is a *soft* clause, otherwise. To avoid notational clutter, we overload $\mathsf{wt}(\cdot)$ to denote the weight of an assignment σ. In particular, we define $\mathsf{wt}(\sigma)$ as the sum of the weights of *unsatisfied clauses* in a CNF, formally, $\mathsf{wt}(\sigma) = \Sigma_{i|\sigma \not\models C_i} \mathsf{wt}(C_i)$.

Given a weighted-CNF formula $F \triangleq \bigvee_i C_i$ with weight $\mathsf{wt}(C_i)$, the weighted-partial MaxSAT problems finds an optimal assignment σ^* that achieves the minimum weight. Formally, $\sigma^* = \mathsf{MaxSAT}(F, \mathsf{wt}(\cdot))$ if $\forall \sigma \neq \sigma^*, \mathsf{wt}(\sigma^*) \leq \mathsf{wt}(\sigma)$. The optimal weight of the MaxSAT problem $\mathsf{wt}(\sigma^*)$ is infinity (∞) when at least one hard clause becomes unsatisfied. Therefore, the weighted-partial MaxSAT problem finds σ^* that satisfies all hard clauses and as many soft clauses as possible such that the total weight of unsatisfied soft clauses is minimum. In this paper, we reduce the classification problem to the solution of a weighted-partial MaxSAT problem. The knowledge of the inner workings of MaxSAT solvers is not required for this paper.

3.3 Rule-based Classification

We consider a binary classification problem where $\mathbf{X} \in \{0,1\}^{n \times m}$ is a binary feature matrix[2] defined over Boolean features $\mathbf{x} = \{x_1, x_2, \ldots, x_m\}$ and $\mathbf{y} \in \{0,1\}^n$ is a vector of binary class labels. The i^{th} row of $\mathbf{X}$, denoted by $\mathbf{X}_i \in \mathcal{X}$, is called a sample, which is generated from a product distribution $\mathcal{X}$. Thus, $\mathbf{X}_i$ is the valuation of features in $\mathbf{x}$ and y_i is its class label. $\mathbf{X}_i$ is called a positive sample if $y_i = 1$, otherwise a negative sample.

2. Our framework allows learning on real-valued and categorical features, as discussed in Section 5.3.

A classifier $\mathcal{R} : \mathbf{x} \rightarrow \hat{y} \in \{0, 1\}$ is a function that takes a Boolean vector as input and outputs the predicted class label $\hat{y}$. In a rule-based classifier, we view $\mathcal{R}$ as a propositional formula defined over Boolean features $\mathbf{x}$ such that, if $\mathcal{R}$ becomes true for an input (or an assignment), the prediction $\hat{y} = 1$ and $\hat{y} = 0$, otherwise. The goal of designing $\mathcal{R}$ is to not only approximate the training data, but also generalize to unseen samples arising from the same distribution. Additionally, we prefer to learn a *sparse* $\mathcal{R}$ in order to favor interpretability. We define the structural complexity of $\mathcal{R}$, referred to as *rule-size*, in terms of the number of literals it contains. Let $\mathsf{clause}(\mathcal{R}, i)$ denote the i^{th} clause of $\mathcal{R}$ and $|\mathsf{clause}(\mathcal{R}, i)|$ be the number of literals in $\mathsf{clause}(\mathcal{R}, i)$. Thus, the size of the classifier is $|\mathcal{R}| = \Sigma_i |\mathsf{clause}(\mathcal{R}, i)|$, which is the sum of literals in all clauses in $\mathcal{R}$.

For example, let $\mathcal{R} \triangleq (x_1 \vee x_3) \wedge (\neg x_2 \vee x_3)$ be a CNF classifier defined over three Boolean features $\{x_1, x_2, x_3\}$. For an input $[0, 1, 1]$, the prediction of $\mathcal{R}$ is 1 whereas for an input $[1, 1, 0]$, the prediction is 0 because $\mathcal{R}$ is true and false in these two cases, respectively. Moreover, $|\mathcal{R}| = 2 + 2 = 4$ is the rule-size of $\mathcal{R}$.

Decision Lists. A decision list is a rule-based classifier consisting of "if-then-else" rules. Formally, a decision list (Rivest, 1987) is an ordered list $\mathcal{R}_L$ of pairs $(C_1, v_1), \ldots, (C_k, v_k)$ where C_i is a conjunction of literals (alternately, a single clause in a DNF formula) and $v_i \in \{0, 1\}$ is a Boolean class label[3]. Additionally, the last clause $C_k \triangleq$ true, thereby v_k is the default class. A decision list is defined as a classifier with the following interpretation: for a feature vector $\mathbf{x}$, $\mathcal{R}_L(\mathbf{x})$ is equal to v_i where i is the least index such that $\mathbf{x} \models C_i$. Since the last clause is true, $\mathcal{R}_L(\mathbf{x})$ always exists. Intuitively, whichever clause (starting from the first) in $\mathcal{R}_L$ is satisfied for an input, the associated class label is its prediction.

Decision Sets. A decision set is a set of *independent* "if-then" rules. Formally, a decision set $\mathcal{R}_S$ is a set of pairs $\{(C_1, v_1), \ldots, (C_{k-1}, v_{k-1})\}$ and a default pair (C_k, v_k), where C_i is—similar to a decision list—a conjunction of literals and $v_i \in \{0, 1\}$ is a Boolean class label. In addition, the last clause $C_k \triangleq$ true and v_k is the default class. For a decision set, if an input $\mathbf{x}$ satisfies one clause, say C_i, then the prediction is v_i. If $\mathbf{x}$ satisfies no clause, then the prediction is the default class v_k. Finally, if $\mathbf{x}$ satisfies ≥ 2 clauses, $\mathbf{x}$ is assigned a class using a tie-breaking (Lakkaraju et al., 2016).

4. Problem Formulation

Given (i) a binary feature matrix $\mathbf{X} \in \{0, 1\}^{n \times m}$ with n samples and m features, (ii) a class-label vector $\mathbf{y} \in \{0, 1\}^n$, (iii) a positive integer $k \geq 1$ denoting the number of clauses, and (iv) a regularization parameter $\lambda \in \mathbb{R}^+$, we learn a classifier $\mathcal{R}$ represented as a k-clause CNF formula to separate samples of class 1 from class 0.

Our goal is to learn classifiers that balance two goals: of being accurate but also interpretable. Various notions of interpretability have been proposed in the context of classification problems. A common proxy for interpretability in the context of decision rules is the sparsity of rule. For instance, a rule involving fewer literals is highly interpretable. In this work, we minimize the total number of literals in all clauses, as discussed in Section 3, which motivates us to find $\mathcal{R}$ with minimum $|\mathcal{R}|$. Let $\mathcal{R}$ classify all samples correctly during

3. In a more practical setting, $v_i \in \{0, \ldots, N\}$ can be multi-class for $N \geq 1$.

training. Among all the classification rules that classify all samples correctly, we choose the sparsest (most interpretable) such $\mathcal{R}$.

$$\min_{\mathcal{R}} |\mathcal{R}| \text{ such that } \forall i, y_i = \mathcal{R}(\mathbf{X}_i)$$

In practical classification tasks, perfect classification is unlikely. Hence, we need to balance interpretability with classification error. Let $\mathcal{E}_{\mathbf{X}} = \{(\mathbf{X}_i, y_i) | y_i \neq \mathcal{R}(\mathbf{X}_i)\}$ be the set of samples in $\mathbf{X}$ that are misclassified by $\mathcal{R}$. Therefore, we balance between classification-accuracy and rule-sparsity and optimize the following function.[4]

$$\min_{\mathcal{R}} \ |\mathcal{E}_{\mathbf{X}}| + \lambda |\mathcal{R}| \tag{1}$$

Higher values of λ generate a rule with a smaller rule-size but of more training errors, and vice-versa. Thus, λ can be tuned to trade-off between accuracy and interpretability for a rule-based classifier, which we experiment extensively in Section 8.

5. MaxSAT-based Interpretable Classification Rule Learning

In this section, we discuss a MaxSAT-based framework for optimal learning of an interpretable rule-based classifier, particularly a CNF classifier $\mathcal{R}$. We first describe the decision variables in Section 5.1 and present the MaxSAT encoding in Section 5.2. Our MaxSAT formulation assumes binary features as input. We conclude this section by learning $\mathcal{R}$ with non-binary features in Section 5.3 and discussing more flexible interpretability constraints of $\mathcal{R}$ in Section 5.4.

5.1 Description of Variables

We initially preprocess $\mathbf{X}$ to account for the negation of Boolean features while learning a classifier.[5] In the preprocessing step, we negate each column in $\mathbf{X}$ to a new column and append it to $\mathbf{X}$. For example, if "age ≥ 25" is a Boolean feature, we add another feature "age < 25" in $\mathbf{X}$ by negating the column "age ≥ 25". Hence, in the rest of the paper, we refer m as the modified number of columns in $\mathbf{X}$. We next discuss the variables in the MaxSAT problem.

We consider two types of Boolean variables: (i) *feature* variables b corresponding to input features and (ii) *error* variables η corresponding to the classification error of samples. We define a Boolean variable b^i_j that becomes true if feature x_j appears in the i^{th} clause of $\mathcal{R}$, thereby contributing to an increase in the rule-size of $\mathcal{R}$, and b^i_j is assigned false, otherwise. Moreover, we define an error variable η_l to attribute to whether the sample $\mathbf{X}_l$ is classified correctly or not. Specifically, η_l becomes true if $\mathbf{X}_l$ is misclassified, and becomes false otherwise. We next discuss the MaxSAT encoding to solve the classification problem.

4. In our formulation, it is straightforward to add class-conditional weights (e.g., to penalize false-alarms more than mis-detects), and to allow instance weights (per sample).
5. This preprocessing is similarly applied in (Malioutov & Varshney, 2013).

5.2 MaxSAT Encoding

We consider a weighted-partial MaxSAT formula, where we encode the objective function in Eq. (1) as soft clauses. For each sample in $\mathbf{X}$, we construct hard clauses specifying that the sample is either classified correctly and η_l is false or it is misclassified and η_l is true. We next discuss the MaxSAT encoding in detail.

- **Soft clauses for maximizing training accuracy:** For each training sample, IMLI constructs a soft unit[6] clause $\neg\eta_l$ to account for a penalty for misclassification. Since the penalty for misclassification of a sample is 1 in Eq. (1), the weight of this soft clause is also 1.

$$E_l := \neg\eta_l; \qquad \mathsf{wt}(E_l) = 1 \tag{2}$$

  Intuitively, if a sample is misclassified, the associated error variable becomes true, thereby dissatisfying the soft clause E_l.

- **Soft clauses for minimizing rule-sparsity:** To favor rule-sparsity, IMLI tries to learn a classifier with as few literals as possible. Hence, for each feature variable b^i_j, IMLI constructs a unit clause as $\neg b^i_j$. Similar to training accuracy, the weight for this clause is derived as λ from Eq. (1).

$$S^i_j := \neg b^i_j; \qquad \mathsf{wt}(S^i_j) = \lambda \tag{3}$$

- **Hard clauses for encoding constraints:** In a MaxSAT problem, constraints that must be satisfied are encoded as hard clauses. In IMLI, we have a learning constraint that, if the error variable is false, the associated sample must be correctly classified, and vice-versa. Let $\mathbf{b}_i = \{b^i_j \mid j \in \{1, \ldots, m\}\}$ be a vector of feature variables corresponding to the i^{th} clause in $\mathcal{R}$. Then, we define the following hard clause.

$$H_l := \neg\eta_l \rightarrow \Big(y_l \leftrightarrow \bigwedge_{i=1}^{k} \mathbf{X}_l \circ \mathbf{b}_i\Big); \qquad \mathsf{wt}(H_l) = \infty \tag{4}$$

  In the hard clause, $\bigwedge_{i=1}^{k} \mathbf{X}_l \circ \mathbf{b}_i$ is a CNF formula including variables b^i_j for which the associated feature-value is 1 in $\mathbf{X}_l$. Since $y_l \in \{0, 1\}$ is a constant, the constraint to the right of the implication "$\rightarrow$" is either $\bigwedge_{i=1}^{k} \mathbf{X}_l \circ \mathbf{b}_i$ or its complement. Therefore, the hard clause enforces that if the sample is correctly classified (using $\neg\eta_l$), either $\bigwedge_{i=1}^{k} \mathbf{X}_l \circ \mathbf{b}_i$ or its complement is true depending on the class-label of the sample. We highlight that the single-implication "$\rightarrow$" in the hard clause H_l acts as a double-implication "$\leftrightarrow$" due to the soft clause E_l. Because, according to the definition of "$\rightarrow$", the left constraint $\neg\eta_l$ can be false while the right constraint of "$\rightarrow$" is true. This, however, incurs unnecessarily dissatisfying the soft clause E_l, which is a sub-optimal solution and hence this solution is not returned by the MaxSAT solver.

We next discuss the translation of soft and hard clauses into a CNF formula, which can be invoked by any MaxSAT solver.

6. A unit clause has a single literal.

Translating E_l, S^i_j, H_l to a CNF Formula. The soft clauses E_l and S^i_j are unit clauses and hence, no translation is required for them. In the hard clause, when $y_l = 1$, the simplification is $H_l := \neg\eta_l \rightarrow \bigwedge_{i=1}^{k} \mathbf{X}_l \circ \mathbf{b}_i$. In this case, we apply the equivalence rule in propositional logic $(a \rightarrow b) \equiv (\neg a \vee b)$ to encode H_l into CNF. In contrast, when $y_i = 0$, we simplify the hard clause as $H_l := \neg\eta_l \rightarrow \neg(\bigwedge_{i=1}^{k} \mathbf{X}_l \circ \mathbf{b}_i) \Rightarrow \neg\eta_l \rightarrow \bigvee_{i=1}^{k} \neg(\mathbf{X}_l \circ \mathbf{b}_i)$. Since $\mathbf{X}_l \circ \mathbf{b}_i$ constitutes a disjunction of literals (defined in Section 3), we apply Tseytin transformation to encode $\neg(\mathbf{X}_l \circ \mathbf{b}_i)$ into CNF. More specifically, we introduce an auxiliary variable $z_{l,i}$ corresponding to the clause $\neg(\mathbf{X}_l \circ \mathbf{b}_i)$. Formally, we replace $H_l := \neg\eta_l \rightarrow \bigvee_{i=1}^{k} \neg(\mathbf{X}_l \circ \mathbf{b}_i)$ with $\bigwedge_{i=0}^{k} H_{l,i}$, where $H_{l,0} := (\neg\eta_l \rightarrow \bigvee_{i=1}^{k} z_{l,i})$ and $H_{l,i} := z_{l,i} \rightarrow \neg(\mathbf{X}_l \circ \mathbf{b}_i)$ for $i = \{1, \ldots, k\}$. Finally, we apply the equivalence of $(a \rightarrow b) \equiv (\neg a \vee b)$ on $H_{l,i}$ to translate them into CNF. For either value of y_l, the weight of each translated hard clause in the CNF formula is ∞.

Once we translate all soft and hard clauses into CNF, the MaxSAT query Q is the conjunction of all clauses.

$$Q := \bigwedge_{l=1}^{n} E_l \wedge \bigwedge_{i=1,j=1}^{i=k,j=m} S^i_j \wedge \bigwedge_{l=1}^{n} H_l$$

Any off-the-shelf MaxSAT solver can output an optimal assignment σ^* of the MaxSAT query $(Q, \mathsf{wt}(\cdot))$. We extract σ^* to construct the classifier $\mathcal{R}$ and compute training errors as follows.

Construction 1. *Let $\sigma^* = \mathsf{MaxSAT}(Q, \mathsf{wt}(\cdot))$. Then $x_j \in \mathsf{clause}(\mathcal{R}, i)$ if and only if $\sigma^*(b^i_j) = 1$. Additionally, $\mathbf{X}_l$ is misclassified if and only if $\sigma^*(\eta_l) = 1$.*

Complexity of the MaxSAT Query. We analyze the complexity of the MaxSAT query in terms of the number of Boolean variables and clauses in the CNF formula Q.

Proposition 1. *To learn a k-clause CNF classifier for a dataset of n samples over m boolean features, the MaxSAT query Q defines $km + n$ Boolean variables. Let n_{neg} be the number of negative samples in the training dataset. Then the number of auxiliary variables in Q is kn_{neg}.*

Proposition 2. *The MaxSAT query Q has $km + n$ unit clauses corresponding to the constraints E_l and S^i_j. For each positive sample, the hard clause H_l is translated into k clauses. For each negative sample, the CNF translation requires at most $km + 1$ clauses. Let n_{pos} and n_{neg} be the number of positive and negative samples in the training dataset. Therefore, the number of clauses in the MaxSAT Query Q is $km+n+kn_{pos}+(km+1)n_{neg} \approx \mathcal{O}(kmn)$ when $n_{pos} = n_{neg} = \frac{n}{2}$.*

In the following, we illustrate the encoding for a toy example.

Example of encoding

We illustrate the MaxSAT encoding for a toy example consisting of four samples and two binary features. Our goal is to learn a two clause CNF classifier that can approximate the

training data.

$$\mathbf{X}_{\mathsf{orig}} = \begin{matrix} x_1 & x_2 \end{matrix} \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \implies \mathbf{X} = \begin{matrix} x_1 & \neg x_1 & x_2 & \neg x_2 \end{matrix} \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}; \quad \mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

In the preprocessing step, we negate the columns in $\mathbf{X}_{\mathsf{orig}}$ and add complemented features $\{\neg x_1, \neg x_2\}$ in $\mathbf{X}$. In the MaxSAT encoding, we define 8 feature variables (4 features $\times$ 2 clauses in the classifier) and 4 error variables. For example, for feature x_2, introduced feature variables are $\{b_3^1, b_3^2\}$ and for feature $\neg x_2$, introduced variables are $\{b_4^1, b_4^2\}$. For four samples, error variables are $\{\eta_1, \eta_2, \eta_3, \eta_4\}$. We next show the soft and hard clauses in the MaxSAT encoding

$$\begin{aligned}
&E_1 := (\neg\eta_1); \quad E_2 := (\neg\eta_2); \quad E_3 := (\neg\eta_3); \quad E_4 := (\neg\eta_4) \\
&S_1^1 := (\neg b_1^1); \quad S_2^1 := (\neg b_2^1); \quad S_3^1 := (\neg b_3^1); \quad S_4^1 := (\neg b_4^1); \\
&S_1^2 := (\neg b_1^2); \quad S_2^2 := (\neg b_2^2); \quad S_3^2 := (\neg b_3^2); \quad S_4^2 := (\neg b_4^2); \\
&H_1 := (\neg\eta_1 \rightarrow ((b_2^1 \vee b_4^1) \wedge (b_2^2 \vee b_4^2))); \\
&H_2 := (\neg\eta_2 \rightarrow (\neg(b_2^1 \vee b_3^1) \vee \neg(b_2^2 \vee b_3^2))); \\
&H_3 := (\neg\eta_3 \rightarrow (\neg(b_1^1 \vee b_4^1) \vee \neg(b_1^2 \vee b_4^2))); \\
&H_4 := (\neg\eta_4 \rightarrow ((b_1^1 \vee b_3^1) \wedge (b_1^2 \vee b_3^2)));
\end{aligned}$$

In this example, we consider regularizer $\lambda = 0.1$, thereby setting the weight on accuracy as $\mathsf{wt}(E_i) = 1$ and rule-sparsity weight as $\mathsf{wt}\left(S_j^i\right) = 0.1$. Intuitively, the penalty for misclassifying a sample is 10 times than the penalty for adding a feature in the classifier. For this MaxSAT query, the optimal solution classifies all samples correctly by assigning four feature variables $\{b_1^1, b_4^1, b_2^2, b_3^2\}$ to true. Therefore, by applying Construction 1, the classifier is $(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2)$.[7]

5.3 Learning with Non-binary Features

The presented MaxSAT encoding requires input samples to have binary features. Therefore, we discretize datasets with categorical and continuous features into binary features. For each continuous feature, we apply equal-width discretization that splits the feature into a fixed number of bins. For example, consider a continuous feature $x_c \in [a, b]$. In discretization, we split the domain $[a, b]$ into three bins with two split points $\{a', b'\}$ such that $a < a' < b' < b$. Therefore, the resulting three discretized features are $a \leq x_c < a'$, $a' \leq x_c < b'$, and $b' \leq x_c \leq b$. An alternate to this close-interval discretization is open-interval discretization, where we consider six discretized features with each feature being

7. The presented MaxSAT-based formulation does not learn a CNF classifier with both x_i and $\neg x_i$ in the same clause. The reason is that a clause with both x_i and $\neg x_i$ connected by OR ($\vee$) does not increase accuracy but increases rule-size and hence, this is not an optimal classifier.

compared with one threshold. In that case, the discretized features are: $x_c \geq a, x_c \geq a', x_c \geq b', x_c < a', x_c < b', x_c \leq b$. Both open-interval and close-interval discretization techniques have their use-cases where one or the other is appropriate. For simplicity, we experiment with close-interval discretization in this paper.

After applying discretization on continuous features, the dataset contains categorical features only, which we convert to binary features using one-hot encoding (Lakkaraju et al., 2019; Ghosh & Meel, 2019). In one-hot encoding, a Boolean vector of features is introduced with cardinality equal to the number of distinct categories. For example, consider a categorical feature having three categories 'red', 'green', and 'yellow'. In one hot encoding, samples with category-value 'red', 'green', and 'yellow' would be converted into binary features by taking values 100, 010, and 001, respectively.

5.4 Flexible Interpretability Objectives

IMLI can be extended to consider more flexible interpretability objectives than the simplified one in Eq. (1). In Eq. (1), we prioritize all features equally by providing the same weight to the clause S^i_j for all values of i and j. In practice, users may prefer rules containing certain features. In IMLI, such an extension can be achieved by modifying the weight function and/or the definition of S^i_j. For example, to constrain the classifier to never include a feature, the weight of the clause $S^i_j := \neg b^i_j$ can be set to ∞. In contrast, to always include a feature, we can define $S^i_j := b^i_j$ with weight ∞. In both cases, we treat S^i_j as a hard clause.

Another use case may be to learn rules where clauses have disjoint set of features. To this end, we consider a pseudo-Boolean constraint $\sum_{i=1}^{k} b^i_j \leq 1$, which specifies that the feature x_j appears in at-most one of the k clauses. This constraint may be soft or hard depending on the priority in the application domain. In either case, we convert this constraint to CNF using pseudo-Boolean to CNF translation (Philipp & Steinke, 2015). Thus, IMLI allows us to consider varied interpretability constraints by only modifying the MaxSAT query without requiring changes in the MaxSAT solving. Therefore, the *separation between modeling and solving* turns out to be the key strength of IMLI.

6. Incremental Learning with MaxSAT-based Formulation

The complexity of the MaxSAT query in Section 5.2 increases with the number of samples in the training dataset and the number of clauses to be learned in a CNF formula. In this section, we discuss an incremental learning technique that along with MaxSAT formulation can generate classification rules efficiently. Our incremental learning is built on two concepts: (i) mini-batch learning and (ii) iterative learning. In the following, we discuss both concepts in detail.

6.1 Mini-batch Learning

Our first improvement is to implement a mini-batch learning technique tailored for rule-based classifiers. Mini-batch learning has two-fold advantages. Firstly, instead of solving a large MaxSAT query for the whole training data, this approach solves a sequence of smaller MaxSAT queries derived from mini-batches of smaller size. Secondly, this approach extends to life-long learning (Chen & Liu, 2018), where the classifier can be updated incrementally

with new samples while also generalizing to previously observed samples. In our context of rule-based classifiers, we consider the following heuristic in mini-batch learning.

A Heuristic for Mini-Batch Learning. Let $\mathcal{R}'$ be a classifier learned on the previous batch. In mini-batch learning, we aim to learn a new classifier $\mathcal{R}$ that can generalize to both the current batch and previously seen samples. For that, we consider a soft constraint such that $\mathcal{R}$ does not *differ much* from $\mathcal{R}'$ while training on the current batch. Thus, we hypothesize that by constraining $\mathcal{R}$ to be syntactically similar to $\mathcal{R}'$, it is possible to generalize well; because samples in all batches originate from the same distribution. Since our study focuses on rule-based classifiers, we consider the Hamming distance between two classifiers as a notion of their syntactic dissimilarity. In the following, we define the Hamming distance between two CNF classifiers $\mathcal{R}$, $\mathcal{R}'$ with the same number of clauses.

$$d_H(\mathcal{R}, \mathcal{R}') = \sum_{i=1}^{k} \Big(\sum_{v \in C_i} \mathbb{1}(v \notin C'_i) + \sum_{v \in C'_i} \mathbb{1}(v \notin C_i) \Big),$$

where C_i and C'_i are the i^{th} clause in $\mathcal{R}$ and $\mathcal{R}'$, respectively and $\mathbb{1}$ is an indicator function that returns 1 if the input is true and returns 0 otherwise. Intuitively, $d_H(\mathcal{R}, \mathcal{R}')$ calculates the total number of different literals in each (ordered) pair of clauses in $\mathcal{R}$ and $\mathcal{R}'$. For example, consider $\mathcal{R} = (x_1 \vee x_2) \wedge (\neg x_1)$ and $\mathcal{R}' = (\neg x_1 \vee x_2) \wedge (\neg x_1)$. Then $d_H(\mathcal{R}, \mathcal{R}') = 2 + 0 = 2$, because in the first clause, the literals $\{x_1, \neg x_1\}$ are absent in either formulas, and the second clause is identical for both $\mathcal{R}$ and $\mathcal{R}'$. In the following, we discuss a modified objective function for mini-batch learning.

Objective Function. In mini-batch learning, we design an objective function to penalize both classification errors on the current batch and the Hamming distance between new and previous classifiers. Let $(\mathbf{X}^b, \mathbf{y}^b)$ be the current mini-batch. The objective function in mini-batch learning is

$$\min_{\mathcal{R}} |\mathcal{E}_{\mathbf{X}_b}| + \lambda d_H(\mathcal{R}, \mathcal{R}'). \tag{5}$$

In the objective function, $\mathcal{E}_{\mathbf{X}_b}$ is the misclassified subset of samples in the current batch $(\mathbf{X}^b, \mathbf{y}^b)$. Unlike controlling the rule-sparsity in the non-incremental approach in the earlier section, in Eq. (5), λ controls the trade-off between classification errors and the syntactic differences between consecutive classifiers. Next, we discuss how to encode the modified objective function as a MaxSAT query.

MaxSAT Encoding of Mini-batch Learning. In order to account for the modified objective function, we only modify the soft clause S^i_j in the MaxSAT query Q. In particular, we define S^i_j to penalize for the complemented assignment of the feature variable b^i_j in $\mathcal{R}$ compared to $\mathcal{R}'$.

$$S^i_j := \begin{cases} b^i_j & \text{if } x_j \in \mathsf{clause}(\mathcal{R}', i) \\ \neg b^i_j & \text{otherwise} \end{cases}; \qquad \mathsf{wt}(S^i_j) = \lambda$$

S^i_j is either a unit clause b^i_j or $\neg b^i_j$ depending on whether the associated feature x_j appears in the previous classifier $\mathcal{R}'$ or not. In this encoding, the Hamming distance of

$\mathcal{R}$ and $\mathcal{R}'$ is minimized while attaining minimum classification errors on the current batch (using soft clause E_l in Eq. (2)) To this end, mini-batch learning starts with an empty CNF formula as $\mathcal{R}'$ with no feature, and thus $S^i_j := \neg b^i_j$ for the first batch. Next, we analyze the complexity of the MaxSAT encoding for mini-batch learning.

Proposition 3. *Let $n' \triangleq |\mathbf{X}_b| \ll |\mathbf{X}|$ be the size of a mini-batch, $n'_{neg} \leq n'$ be the number of negative samples in the batch, and m be the number of Boolean features. According to Proposition 1, to learn a k-clause CNF classifier in mini-batch learning, the MaxSAT encoding for each batch has $km+n'$ Boolean variables and kn'_{neg} auxiliary variables. Let $n'_{pos} = n' - n'_{neg}$ be the number of positive samples in the batch. Then, according to Proposition 2, the number of clauses in the MaxSAT query for each batch is $km+n'+kn'_{pos}+(km+1)n'_{neg} \approx \mathcal{O}(kmn')$ when $n'_{pos} = n'_{neg} = \frac{n'}{2}$.*

Assessing the Performance of $\mathcal{R}$. Since we apply a heuristic objective in mini-batch learning, $\mathcal{R}$ may be optimized for the current batch while generalizing poorly on the full training data. To tackle this, after learning on each batch, we decide whether to keep $\mathcal{R}$ or not by assessing the performance of $\mathcal{R}$ on the training data and keep $\mathcal{R}$ if it achieves higher performance. We measure the performance of $\mathcal{R}$ on the full training data $(\mathbf{X}, \mathbf{y})$ using a weighted combination of classification errors and rule-size. In particular, we compute a combined loss function $\mathsf{loss}(\mathcal{R}) \triangleq |\mathcal{E}_{\mathbf{X}}| + \lambda|\mathcal{R}|$ on the training data $(\mathbf{X}, \mathbf{y})$, which is indeed the value of the objective function that we minimize in the non-incremental learning in Section 4. Additionally, we discard the current classifier $\mathcal{R}$ when the loss does not decrease ($\mathsf{loss}(\mathcal{R}) > \mathsf{loss}(\mathcal{R}')$).

We present the algorithm for mini-batch learning in Algorithm 1.

Algorithm 1 MaxSAT-based Mini-batch Learning

```
1:  function MiniBatchLearning(X, y, λ, k)
2:      R =← ⋀_{i=1}^{k} false                          ▷ Empty CNF formula
3:      loss_max ← ∞
4:      for i ← 1, ..., N do                            ▷ N is the total batch-count
5:          X^b, y^b ← GetBatch(X, y)
6:          Q, wt(·) ← MaxSATEncoding(R, X^b, y^b, λ, k)   ▷ Returns a weighted-partial
    CNF
7:          σ* ← MaxSAT(Q, wt(·))
8:          R_new ← ConstructClassifier(σ*)             ▷ Construction 1
9:          loss ← |E_X| + λ|R_new|                     ▷ Compute loss
10:         if loss < loss_max then
11:             loss_max ← loss
12:             R ← R_new
13:     return R
```

6.2 Iterative Learning

We discuss an iterative learning algorithm for rule-based classifiers. The major advantage of iterative learning is that we solve a smaller MaxSAT query because of learning

a *partial* classifier in each iteration. Our iterative approach is motivated by the set-covering algorithm—also known as separate-and-conquer algorithm—in symbolic rule learning (Fürnkranz, 1999). In this approach, the core idea is to define the *coverage of a partial classifier* (for example, a clause in a CNF classifier). For a specific definition of coverage, this algorithm separates samples covered by the partial classifier and recursively conquers remaining samples in the training data by learning another partial classifier until no sample remains. The final classifier is an aggregation of all partial classifiers—the conjunction of clauses in a CNF formula, for example.

Iterative learning is different from mini-batch learning in several aspects. In mini-batch learning, we learn all clauses in a CNF formula together, while in iterative learning, we learn a single clause of a CNF in each iteration. Additionally, in mini-batch learning, we improve scalability by reducing the number of samples in the training data using mini-batches, while in iterative learning, we improve scalability by reducing the number of clauses to learn at once. Therefore, an efficient integration of iterative learning and mini-batch learning would benefit scalability from both worlds. In the following, we discuss this integration by first stating iterative learning for CNF classifiers.

In iterative learning, we learn one clause of a CNF classifier in each iteration, where the clause refers to a partial classifier. The coverage of a clause in a CNF formula is the set of samples that *do not satisfy* the clause. The reason is that if a sample does not satisfy at least one clause in a CNF formula, the prediction of the sample by the full formula is class 0, because CNF is a conjunction of clauses. As a result, considering covered samples in the next iteration does not change their prediction regardless of whatever clause we learn in later iterations. To this end, a single clause learning can be performed efficiently by applying mini-batch learning discussed before. In Algorithm 2, we provide an algorithm for learning a CNF classifier iteratively by leveraging mini-batch learning. This algorithm is a double-loop algorithm, where in the outer loop we apply iterative learning and in the inner loop, we apply mini-batch learning.

Algorithm 2 Iterative CNF Classifier Learning

```
1:  procedure IterativeCNFLearning(X, y, λ, k)
2:      R ← true                                              ▷ Initial formula
3:      for i ← 1, ..., k and (X, y) ≠ ∅ do
4:          C_i ← MiniBatchLearning(X, y, λ, 1)               ▷ Single clause learning, k = 1
5:          X', y' ← Coverage(X, C_i)
6:          if (X', y') = ∅ then                              ▷ Terminating conditions
7:              break
8:          R ← R ∧ C_i
9:          X, y ← (X, y) \ (X', y')                          ▷ Removing covered samples
10:     return R
```

Terminating Conditions. In Algorithm 2, we terminate iterative learning based on three conditions: (i) when $\mathcal{R}$ contains all k clauses , (ii) the training data $(\mathbf{X}, \mathbf{y})$ is empty (that is, no sample remains uncovered), and (iii) no new sample is covered by the current partial classifier. Since the first two conditions are trivial, we elaborate on the third condition.

When clause C_i cannot cover any new sample from the training dataset $(\mathbf{X}, \mathbf{y})$, the next iteration will result in the same clause C_i because the training data remains the same. In this case, we do not include clause C_i to classifier $\mathcal{R}$ because of zero coverage.

7. Applying IMLI on Learning Other Interpretable Classifiers

In earlier sections, we discuss the learning of CNF classifiers using IMLI. IMLI can also be applied to learn other interpretable rule-based representations. In this section, we discuss how IMLI can be applied in learning DNF classifiers, decision lists, and decision sets.

7.1 Learning DNF Classifiers

For learning DNF classifiers, we leverage De Morgan's law where complementing a CNF formula results in a DNF formula. To learn a DNF classifier, say $\mathcal{R}'(\mathbf{x})$, we can trivially show that $y = \mathcal{R}'(\mathbf{x}) \leftrightarrow \neg(y = \neg\mathcal{R}'(\mathbf{x}))$ for the feature vector $\mathbf{x}$. Here $\neg\mathcal{R}'(\mathbf{x})$ is a CNF formula, by definition. Thus, we learn a DNF classifier by first complementing the class-label $\mathbf{y}$ to $\neg\mathbf{y}$ in the training dataset, learning a CNF classifier on $(\mathbf{X}, \neg\mathbf{y})$, and finally complementing the learned classifier to DNF. For example, the CNF classifier "(Male $\vee$ Age $<$ 50) $\wedge$ (Education = Graduate $\vee$ Income $\geq$ 1500)" is complemented to a DNF classifier as "(not Male $\wedge$ Age $\geq$ 50) $\vee$ (Education $\neq$ Graduate $\wedge$ Income $\leq$ 1500)".

To learn a DNF classifier incrementally, such as through mini-batch and iterative learning, we adopt the following procedure. For learning a DNF classifier using mini-batch learning, we first learn a CNF classifier on dataset $(\mathbf{X}, \neg\mathbf{y})$ and complement the classifier to a DNF classifier at the end of mini-batch learning. To learn a DNF classifier in the iterative approach, we learn a single clause of the DNF classifier in each iteration, remove covered samples, and continue till no training sample remains. In this context, the coverage of a clause in a DNF formula is the set of samples satisfying the clause.

7.2 Learning Decision Lists

In IMLI, we apply an iterative learning approach for efficiently learning a decision list. A decision list $\mathcal{R}_L$ is a list of pairs $(C_1, v_1), \ldots, (C_k, v_k)$, where we propose to learn one pair in each iteration. We note that the clause C_i is a conjunction of literals—equivalently, a single clause DNF formula. Hence, our task is to deploy IMLI to efficiently learn a single clause DNF formula C_i. In particular, we opt to learn this clause for the majority class, say v_i, in the training dataset by setting the majority samples as class 1 and all other samples as class 0. As a result, even if the MaxSAT-based learning, presented in this paper, is targeted for binary classification, we can learn a multi-class decision list in IMLI.

In Algorithm 3, we present the iterative algorithm for learning decision lists. In each iteration, the algorithm learns a pair (C_i, v_i), separates the training set based on the coverage of (C_i, v_i) and conquers the remaining samples recursively. The coverage of (C_i, v_i) is the set of samples that satisfies clause C_i. Finally, we add a default rule $(C_k, v_{\mathsf{default}})$ to $\mathcal{R}_L$ where $C_k \triangleq$ true denoting that the clause is satisfied by all samples. We select the default class v_{default} in the following order: (i) if any class(s) is not present in the predicted classes $\{v_i\}_{i=1}^{k-1}$ of the decision list, v_{default} is the majority class among missing classes, and (ii) v_{default} is the majority class of the original training set $(\mathbf{X}, \mathbf{y})$, otherwise.

Algorithm 3 Iterative learning of decision lists

procedure DecisionListLearning($\mathbf{X}, \mathbf{y}, \lambda, k$)
  $\mathcal{R}_L \leftarrow \{\}$
  for $i \leftarrow 1, \ldots, k-1$ and $(\mathbf{X}, \mathbf{y}) \neq \emptyset$ **do**
    $v_i \leftarrow$ MajorityClass($\mathbf{y}$) ▷ v_i specifies the target class
    $C_i \leftarrow$ MiniBatchDNFLearning($\mathbf{X}, \mathbf{y}, \lambda, 1, v_i$) ▷ Ref. Section 7.1
    $\mathbf{X}', \mathbf{y}' \leftarrow$ Coverage($\mathbf{X}, C_i$)
    if $(\mathbf{X}', \mathbf{y}') = \emptyset$ **then**
      break
    $\mathcal{R}_L \leftarrow \mathcal{R}_L \cup \{(C_i, v_i)\}$
    $\mathbf{X}, \mathbf{y} \leftarrow (\mathbf{X}, \mathbf{y}) \setminus (\mathbf{X}', \mathbf{y}')$
  $\mathcal{R}_L \leftarrow \mathcal{R}_L \cup \{(\text{true}, v_{\mathsf{default}})\}$ ▷ Default rule
  return $\mathcal{R}_L$

7.3 Learning Decision Sets

We describe an iterative procedure for learning decision sets. A decision set comprises of an individual clause-class pair (C_i, v_i) where C_i denotes a single clause DNF formula similar to decision lists. In a decision set, a sample can satisfy multiple clauses simultaneously, which is attributed as an *overlapping between clauses* (Lakkaraju et al., 2016). Concretely, the overlap between two clauses C_i and C_j with $i \neq j$ is the set of samples $\{\mathbf{X}_l | \mathbf{X}_l \models C_i \wedge \mathbf{X}_l \models C_j\}$ satisfying both clauses. One additional objective in learning a decision set is to minimize the overlap between clauses, as studied in (Lakkaraju et al., 2016). Therefore, along with optimizing accuracy and rule-sparsity, we propose an iterative procedure for decision sets that additionally minimizes the overlap between clauses.

Algorithm 4 Iterative learning of decision sets

procedure DecisionSetsLearning($\mathbf{X}, \mathbf{y}, \lambda, k$)
  $\mathcal{R}_S = \{\}$
  $\mathbf{X}^{\text{cc}} = \{\}$ ▷ Contains correctly covered samples
  for $i \leftarrow 1, \ldots, k-1$ and $(\mathbf{X}, \mathbf{y}) \neq \emptyset$ **do**
    $v_i \leftarrow$ MajorityClass($\mathbf{X}, \mathbf{y}$)
    $\mathbf{X}^{\text{w}} \leftarrow \mathbf{X} \cup \mathbf{X}^{\text{cc}}$ ▷ Correctly covered samples are included
    $\mathbf{y}^{\text{w}} \leftarrow \mathbf{y} \cup \{\neg v_i\}^{|\mathbf{X}^{\text{cc}}|}$ ▷ $\mathbf{X}^{\text{cc}}$ have complemented class $\neg v_i$ to minimize overlap
    $C_i \leftarrow$ MiniBatchDNFLearning($\mathbf{X}^{\text{w}}, \mathbf{y}^{\text{w}}, \lambda, 1, v_i$)
    $\mathbf{X}', \mathbf{y}' \leftarrow$ CorrectCoverage($\mathbf{X}, \mathbf{y}, C_i, v_i$)
    if $(\mathbf{X}', \mathbf{y}') = \emptyset$ **then**
      break
    $\mathcal{R}_S \leftarrow \mathcal{R}_S \cup \{(C_i, v_i)\}$
    $\mathbf{X}, \mathbf{y} \leftarrow (\mathbf{X}, \mathbf{y}) \setminus (\mathbf{X}', \mathbf{y}')$
    $\mathbf{X}^{\text{cc}} = \mathbf{X}^{\text{cc}} \cup \mathbf{X}'$
  $\mathcal{R}_S \leftarrow \mathcal{R}_S \cup \{(\text{true}, v_{\mathsf{default}})\}$
  return $\mathcal{R}_S$

In Algorithm 4, we present an iterative algorithm for learning a decision set. The iterative algorithm is a modification of separate-and-conquer algorithm by additionally focusing on minimizing overlaps in a decision set. Given a training data $(\mathbf{X}, \mathbf{y})$, a regularization parameter λ, and the number of clauses k, the core idea of Algorithm 4 is to learn a pair (C_i, v_i) in each iteration, separate covered samples from $(\mathbf{X}, \mathbf{y})$, and conquer remaining samples recursively. In contrast to learning decision lists, we have following modifications in Algorithm 4.

- The first modification is with respect to the definition of coverage for decision sets. Unlike decision lists, we separate samples that are *correctly covered* by (C_i, v_i) in each iteration. Given $(\mathbf{X}, \mathbf{y})$, the correctly covered samples of (C_i, v_i) is a set $\{(\mathbf{X}_l, \mathbf{y}_l) | \mathbf{X}_l \models C_i \wedge \mathbf{y}_l = v_i\} \subseteq (\mathbf{X}, \mathbf{y})$ of samples that satisfy C_i and have matching class-label as v_i.
- The second modification is related to the training dataset considered in each iteration. Let $(\mathbf{X}, \mathbf{y})$ denote *the remaining training dataset* in the current iteration and v_i be the majority class in $(\mathbf{X}, \mathbf{y})$. Hence, v_i is the target class in the current iteration. In Algorithm 4, we learn clause C_i on *a working dataset* $(\mathbf{X}^{\mathrm{w}}, \mathbf{y}^{\mathrm{w}})$, where $\mathbf{X}^{\mathrm{w}} \triangleq \mathbf{X} \cup \mathbf{X}^{\mathrm{cc}}$ comprises of both remaining samples and already covered samples in all previous iterations. However, the vector of class labels $\mathbf{y}^{\mathrm{w}} \triangleq \mathbf{y} \cup \{\neg v_i\}^{|\mathbf{X}^{\mathrm{cc}}|}$ comprises of remaining class-label vector $\mathbf{y}$ and complemented class label vector $\{\neg v_i\}^{|\mathbf{X}^{\mathrm{cc}}|}$ associated with feature-matrix $\mathbf{X}^{\mathrm{cc}}$. Thus, by explicitly labeling covered samples as class $\neg v_i$, the new clause C_i learns to falsify already covered samples. This heuristic allows us to minimize the overlap of C_i compared to previously learned clauses $\{C_j\}_{j=1}^{i-1}$.

Finally, the default clause for decision sets is learned similarly as in decision lists.

8. Empirical Performance Analysis

In this section, we empirically evaluate the performance of IMLI. We first present the experimental setup and the objective of the experiments, followed by experimental results.

8.1 Experimental Setup

We implement a prototype of IMLI in Python to evaluate the performance of the proposed MaxSAT-based formulation for learning classification rules. To implement IMLI, we deploy a state-of-the-art MaxSAT solver Open-WBO (Martins et al., 2014), which returns the current best solution upon reaching a timeout.

We compare IMLI with state-of-the-art interpretable and non-interpretable classifiers. Among interpretable classifiers, we compare with RIPPER (Cohen, 1995), BRL (Letham et al., 2015), CORELS (Angelino et al., 2017), and BRS (Wang et al., 2017). Among non-interpretable classifiers, we compare with Random Forest (RF), Support Vector Machine with linear kernels (SVM), Logistic Regression classifier (LR), and k-Nearest Neighbors classifier (kNN). We deploy the Scikit-learn library in Python for implementing non-interpretable classifiers.

We experiment with real-world binary classification datasets from UCI (Dua & Graff, 2017), Open-ML (Vanschoren et al., 2013), and Kaggle repository (`https://www.kaggle.com/datasets`), as listed in Table 1. In these datasets, the number of samples vary from

about 200 to $1,000,000$. The datasets contain both real-valued and categorical features. We process them to binary features by setting the maximum number of bins as 10 during discretization. For non-interpretable classifiers such as RF, SVM, LR, and kNN, which take real-valued features as inputs, we only convert categorical features to one-hot encoded binary features.

We perform ten-fold cross-validation on each dataset and evaluate the performance of different classifiers based on the median prediction accuracy on the test data. Additionally, we compare the median size of generated rules among rule-based interpretable classifiers. We consider a comparable combination (100) of hyper-parameters choices for all classifiers, which we fine-tune during cross-validation. For IMLI, we vary the number of clauses $k \in \{1, 2, \ldots, 5\}$ and the regularization parameter λ in a logarithmic grid by choosing 5 values between 10^{-4} and 10^{1}. In the mini-batch learning in IMLI, we set the number of samples in each mini-batch, $n' \in \{50, 100, 200, 400\}$. Thus, we consider $\lceil n/n' \rceil$ mini-batches, where n denotes the size of training data. To construct mini-batches from a training dataset, we sequentially split the data into $\lceil n/n' \rceil$ batches with each batch having n' samples. Furthermore, to ignore the effect of batch-ordering, we perform mini-batch learning in two rounds such that each batch participates twice in the training.

For BRL algorithm, we vary four hyper-parameters: the maximum cardinality of rules in $\{2, 3, 4\}$, the minimum support of rules in $\{0.05, 0.175, 0.3\}$, and the prior on the expected length and width of rules in $\{2, 4, 6, 8\}$ and $\{2, 5, 8\}$, respectively. For CORELS algorithm, we vary three hyper-parameters: the maximum cardinality of rules in $\{2, 3, 4, 5\}$, the minimum support of rules in $\{0.01, 0.17, 0.33, 0.5\}$, and the regularization parameter in $\{0.005, 0.01, 0.015, 0.02, 0.025, 0.03\}$. For BRS algorithm, we vary three hyper-parameters: the number of initial rules in $\{500, 1000, 1500, 2000, 2500, 3000\}$, the maximum length of rules in $\{1, 2, 3, 4\}$, and the minimum support of rules in $\{1, 4, 7, 10\}$. For RF and RIPPER classifiers, we vary the cut-off on the number of samples in the leaf node using a linear grid between 3 to 500 and 1 to 300, respectively. For SVM and LR classifiers, we discretize the regularization parameter on a logarithmic grid between 10^{-3} and 10^{3}. For kNN, we vary the number of neighbors in a linear grid between 1 and 500. We conduct each experiment on an Intel Xeon E7 $-$ 8857 v2 CPU using a single core with 16 GB of RAM running on a 64bit Linux distribution based on Debian. For all classifiers, we set the training timeout to 1000 seconds.

Objectives of Experiments. In the following, we present the objectives of our experimental study.

1. How are the accuracy and size of classification rules generated by IMLI compared to existing interpretable classifiers?

2. How is the scalability of IMLI in solving large-scale classification problems compared to existing interpretable classifiers?

3. How does IMLI perform in terms of accuracy and scalability compared to classifiers that are non-interpretable?

4. How does the incremental learning in IMLI perform compared to non-incremental MaxSAT-based learning in terms of accuracy, rule-sparsity, and scalability?

5. How do different interpretable classification rules learned using IMLI perform in terms of accuracy and rule-size?

6. What are the effects of different hyper-parameters in IMLI?

Summary of Experimental Results. To summarize our experimental results, IMLI achieves the best balance among prediction accuracy, interpretability, and scalability compared to existing interpretable rule-based classifiers. Particularly, compared to the most accurate classifier RIPPER, IMLI demonstrates on average 1% lower prediction accuracy, wherein the accuracy of IMLI is higher than BRL, CORELS, and BRS in almost all datasets. In contrast, IMLI generates significantly smaller rules than RIPPER, specifically in large datasets. Moreover, BRL, CORELS, and BRS report comparatively smaller rule-size than IMLI on average, but with a significant decrease in accuracy. In terms of scalability, IMLI achieves the best performance compared to other interpretable and non-interpretable classifiers by classifying datasets with one million samples. While CORELS also scales to such large datasets, its accuracy is lower than that of IMLI by at least 2% on average. Therefore, IMLI is not only scalable but also accurate in practical classification problems while also being interpretable by design. We additionally analyze the comparative performance of different formulations presented in this paper, where the incremental approach empirically proves its efficiency than the naïve MaxSAT formulation. Furthermore, we learn and compare the performance of different interpretable representations: decision lists, decision sets, CNF, and DNF formulas using IMLI and present the efficacy of IMLI in learning varied interpretable classifiers. Finally, we study the effect of different hyper-parameters in IMLI, where each hyper-parameter provides a precise control among training time, prediction accuracy, and rule-sparsity.

8.2 Experimental Results

In the following, we discuss our experimental results in detail.

8.2.1 Comparing IMLI with Interpretable Classifiers

We compare IMLI with existing interpretable classifiers in three aspects: test accuracy, rule-size, and scalability.

Test accuracy and rule-size. We present the experimental results of test accuracy and rule-size among interpretable classifiers in Table 1, where the first, second, and third columns represent the name of the dataset, the number of samples, and the number of features in the dataset, respectively. In each cell from the fourth to the eighth column in the table, the top value represents the median test accuracy and the bottom value represents the median size of rules measured through ten-fold cross-validation.

In Table 1, IMLI and CORELS generate interpretable classification rules in all 15 datasets in our experiments. In contrast, within a timeout of 1000 seconds, RIPPER, BRL, and BRS fail to generate any classification rule in three datasets, specifically in large datasets ($\geq 200,000$ samples).

We compare IMLI with each interpretable classifier in detail. Compared to RIPPER, IMLI has lower accuracy in 9 out of 12 datasets. More specifically, the accuracy of IMLI is

Table 1: Comparison of accuracy and rule-size among interpretable classifiers. Each cell from the fourth to the eighth column contains test accuracy (top) and rule-size (bottom). '—' represents a timeout. Numbers in bold represent the best performing results among different classifiers.

Dataset	Size	Features	RIPPER	BRL	CORELS	BRS	IMLI
Parkinsons	195	202	94.44 7.0	**94.74** 11.5	89.74 **2.0**	84.61 5.0	**94.74** 7.5
WDBC	569	278	**98.08** 13.0	93.81 22.0	92.04 **2.0**	92.98 7.0	94.74 11.5
Pima	768	83	77.14 6.0	68.18 13.5	75.32 **2.0**	75.32 3.0	**78.43** 23.0
Titanic	1,043	38	78.72 6.0	62.98 15.0	**81.9** **4.0**	80.86 **4.0**	81.82 5.5
MAGIC	19,020	100	**82.68** 102.0	76.95 81.0	78.05 4.0	77.5 **3.0**	78.26 8.5
Tom's HW	28,179	946	**85.91** 30.0	— —	83.27 **4.0**	83.13 18.5	85.24 44.5
Credit	30,000	199	**82.39** 32.5	46.12 26.5	81.18 **2.0**	80.45 7.0	82.12 17.5
Adult	32,561	94	**84.37** 115.5	72.08 46.5	79.78 **4.0**	70.75 **4.0**	81.2 30.0
Bank Marketing	45,211	82	**90.01** 36.5	84.66 13.0	89.62 **2.0**	86.75 **2.0**	89.84 24.5
Connect-4	67,557	126	**76.72** 118.0	65.83 18.5	68.68 **4.0**	70.49 11.0	75.36 50.5
Weather AUS	107,696	169	**84.22** 26.0	43.26 22.0	83.67 **2.0**	— —	83.78 22.0
Vote	131,072	16	**97.12** 132.0	94.78 41.5	95.86 3.5	95.14 **1.0**	96.69 15.0
Skin Seg	245,057	30	— —	79.25 6.0	91.62 9.0	68.48 **5.0**	**94.71** 30.0
BNG(labor)	1,000,000	89	— —	— —	88.56 **2.0**	— —	**90.91** 24.0
BNG(credit-g)	1,000,000	97	— —	— —	72.08 **2.0**	— —	**75.48** 27.5

1% lower on average than RIPPER. The improved accuracy of RIPPER, however, results in the generation of higher size classification rules than IMLI in most datasets. In particular, IMLI generates sparser rules than RIPPER in 9 out of 12 datasets, wherein RIPPER times out in 3 datasets. Interestingly, the difference in rule-size is more significant in larger datasets, such as in 'Vote' dataset, where RIPPER learns a classifier with 132 Boolean

literals compared to 15 Boolean literals by IMLI. Therefore, IMLI is better than RIPPER in terms of rule-sparsity, but lags slightly in accuracy.

IMLI performs better than BRL both in terms of accuracy and rule-sparsity. In particular, IMLI has higher accuracy and lower rule-size than BRL in 12 and 8 datasets, respectively, in a total of 12 datasets, wherein BRL times out in 3 datasets. While comparing with CORELS, IMLI achieves higher accuracy in almost all datasets (14 out of 15 datasets). A similar trend is observed in comparison with BRS, where IMLI achieves higher accuracy in all of 12 datasets and BRS times out in 3 datasets. CORELS and BRS, however, generates sparser rules than IMLI in most datasets, but by costing a significant decrease in accuracy. For example, in the largest dataset 'BNG(credit-g)' with 1 Million samples, BRS times out, and CORELS generates a classifier with 72.08% accuracy with rule-size 2. IMLI, in contrast, learns a classifier with 27.5 Boolean literals achieving 75.48% accuracy, which is 3% higher than CORELS. Therefore, IMLI makes a good balance between accuracy and rule-size compared to existing interpretable classifiers while also being highly scalable. In the following, we discuss the results on the scalability of all interpretable classifiers in detail.

Scalability. We analyze the scalability among interpretable classifiers by comparing training time. In Figure 1, we use cactus plots[8] to represent the training time (in seconds) of all classifiers in 1000 instances (10 folds $\times$ 100 choices of hyper-parameters) derived for each dataset. In the cactus plot, the number of solved instances (within 1000 seconds) is on the X-axis, whereas the training time is on the Y-axis. A point (x, y) on the plot implies that a classifier yields lower than or equal to y seconds of training in x many instances.

In Figure 1, we present results in an increasing number of samples in a dataset (from left to right and top to bottom). In WDBC and Adult datasets presented on the first row in Figure 1, CORELS solves lower than 600 instances within a timeout of 1000 seconds. The scalability performance of BRS is even worse, where it solves around 700 and 200 instances in WDBC and Adult datasets, respectively. The other three classifiers: IMLI, BRL, and RIPPER solve all 1000 instances, where BRL takes comparatively higher training time than the other two. The performance of IMLI and RIPPER is similar, with RIPPER being comparatively better in the two datasets. However, the efficiency of IMLI compared to other classifiers becomes significant as the number of samples in a dataset increases. In particular, in 'Weather AUS' dataset, BRS cannot solve a single instance, BRL and CORELS solves 400 instances, and RIPPER solves around 600 instances. IMLI, however, solves all 1000 instances in this dataset. Similarly, in 'BNG(labor)' dataset, all other classifiers except IMLI and CORELS cannot solve any instance. While IMLI mostly takes the maximum allowable time (1000 seconds) in solving all instances in this dataset, CORELS can solve lower than 400 instances. Thus, IMLI establishes itself as the most scalable classifier compared to other state-of-the-art interpretable classifiers.

8.2.2 Comparison with Non-interpretable Classifiers

We compare IMLI with state-of-the-art non-interpretable classifiers such as LR, SVM, kNN, and RF in terms of their median test accuracy in Table 2. In the majority of the datasets, IMLI achieves comparatively lower test accuracy than the best performing non-interpretable

8. Cactus plots are often used in (Max)SAT community to present the scalability of different solvers/methods (Argelich et al., 2008; Balyo et al., 2017).

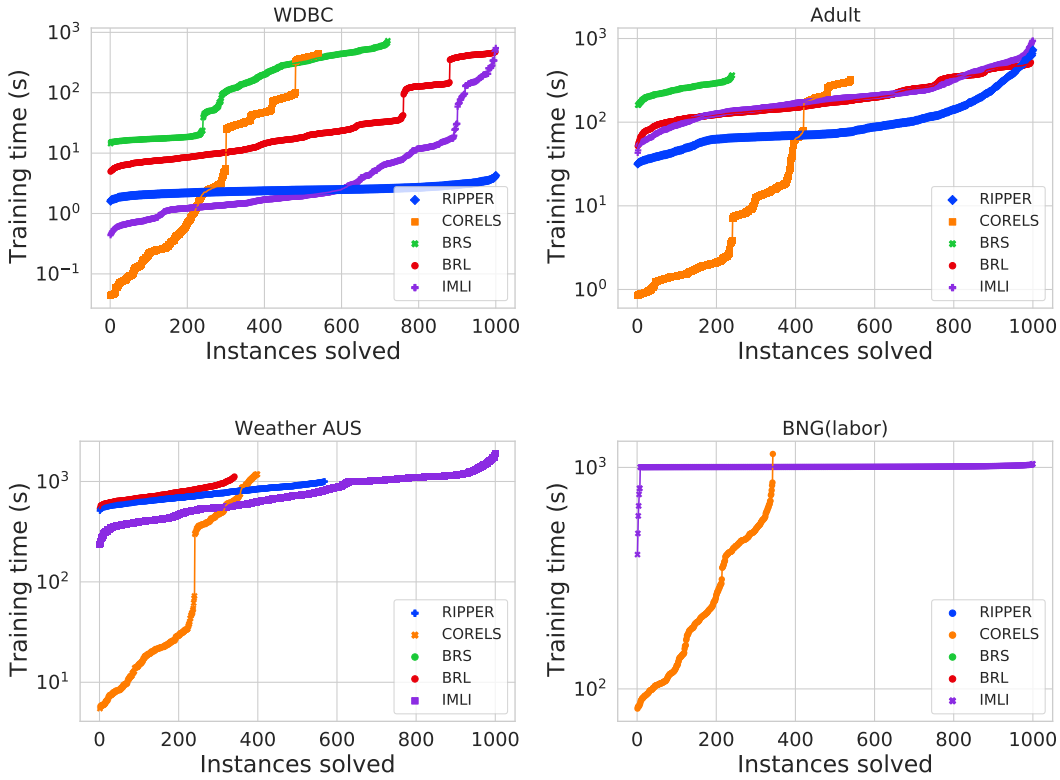

Figure 1: Comparison of scalability among interpretable classifiers. The plots are arranged in increasing sizes of datasets (from left to right). In each cactus plot, IMLI solves all 1000 instances for each dataset, while competitive classifiers often fail to scale, specially in larger datasets.

classifier. The decrease in the test accuracy of IMLI is attributed to two factors. Firstly, while we train IMLI on discretized data, non-interpretable classifiers are trained on non-discretized data and thus IMLI incurs additional classification errors due to discretization. Secondly, IMLI learns a rule-based classifier, whereas non-interpretable classifiers can learn more flexible decision boundaries and thus fit data well. In Table 2, we also observe that IMLI achieves impressive scalability than competing classifiers by solving datasets with $1,000,000$ samples where most of the non-interpretable classifiers fail to learn any decision boundary on such large datasets. Thus, IMLI, being an interpretable classifier, demonstrates lower accuracy than competing non-interpretable classifiers, but higher scalability in practice.

8.2.3 Comparison Among Different Formulations in IMLI

We compare the performance of different formulations for learning classification rules as presented in this paper. In Figure 2, we show cactus plots for assessing training time (in seconds), test error (in percentage), and rule-size among different formulations. In the

Table 2: Comparison of IMLI with non-interpretable classifiers in terms of test accuracy. In the table, '—' represents a timeout. Numbers in bold represent the best performing results among different classifiers.

Dataset	Size	LR	SVM	kNN	RF	IMLI
Parkinsons	195	89.74	89.74	**97.5**	90.0	94.74
WDBC	569	**98.25**	**98.25**	98.23	96.49	94.74
Pima	768	78.43	79.08	74.5	**79.22**	78.43
Titanic	1,043	80.86	80.38	81.34	**82.69**	81.82
MAGIC	19,020	79.18	79.34	84.6	**88.2**	78.26
Tom's HW	28,179	96.2	97.13	88.15	**97.78**	85.24
Credit	30,000	**82.2**	81.9	81.83	82.15	82.12
Adult	32,561	85.26	85.05	83.8	**86.69**	81.2
Bank Marketing	45,211	90.09	89.28	89.43	**90.27**	89.84
Connect-4	67,557	79.39	—	85.51	**88.11**	75.36
Weather AUS	107,696	85.64	—	78.59	**86.26**	83.78
Vote	131,072	96.43	96.37	97.05	**97.38**	96.69
Skin Seg	245,057	91.86	—	**99.96**	**99.96**	94.71
BNG(labor)	1,000,000	—	—	—	—	**90.91**
BNG(credit-g)	1,000,000	—	—	—	**80.58**	75.48

cactus plot, a point (x, y) denotes that the formulation yields lower than or equal to y training time (similarly, test error and rule-size) in x many instances in each dataset.

In Figure 2, we denote the baseline non-incremental MaxSAT-based formulation as IMLI-B[9], incremental MaxSAT-based formulation with only mini-batch learning as IMLI-M, and incremental MaxSAT-based formulation with both mini-batch and iterative learning as IMLI. We first observe the training time of different formulations in the left-most column in Figure 2, where IMLI-B soon times out and solves lower than 300 instances out of 1000 instances in each dataset. This result suggests that the non-incremental formulation cannot scale in practical classification task. Comparing between IMLI and IMLI-M, both formulations solve all 1000 instances in each dataset with IMLI-M undertaking significantly higher training time than IMLI. Therefore, IMLI achieves better scalability than IMLI-M indicating that an integration of mini-batch and iterative learning achieves a significant progress in terms of scalability than mini-batch learning alone.

We next focus on the test error of different formulations in the middle column in Figure 2. Firstly, IMLI-B has a higher test error than the other two formulations since IMLI-B times out in most instances and learns a sub-optimal classification rule with reduced prediction accuracy. In contrast, IMLI has the lowest test error compared to two formulations in all datasets. This result indicates the effectiveness of integrating both iterative and mini-batch learning with MaxSAT-based formulation in generating more accurate classification rules.

Moving focus on the rule-size in the rightmost column in Figure 2, IMLI-B achieves the highest rule-size in WDBC dataset. In contrast, the rule-size of IMLI-B is lowest (zero) in Credit and Adult datasets. In the last two datasets, IMLI-B times out during training

9. Since IMLI-B is a non-incremental formulation and does not involve any mini-batch learning, we consider two hyper-parameters for IMLI-B: the number of clauses in $\{1, 2, \ldots, 10\}$ and the regularization parameter λ as 10 values chosen from a logarithmic grid between 10^{-4} and 10^{1}.

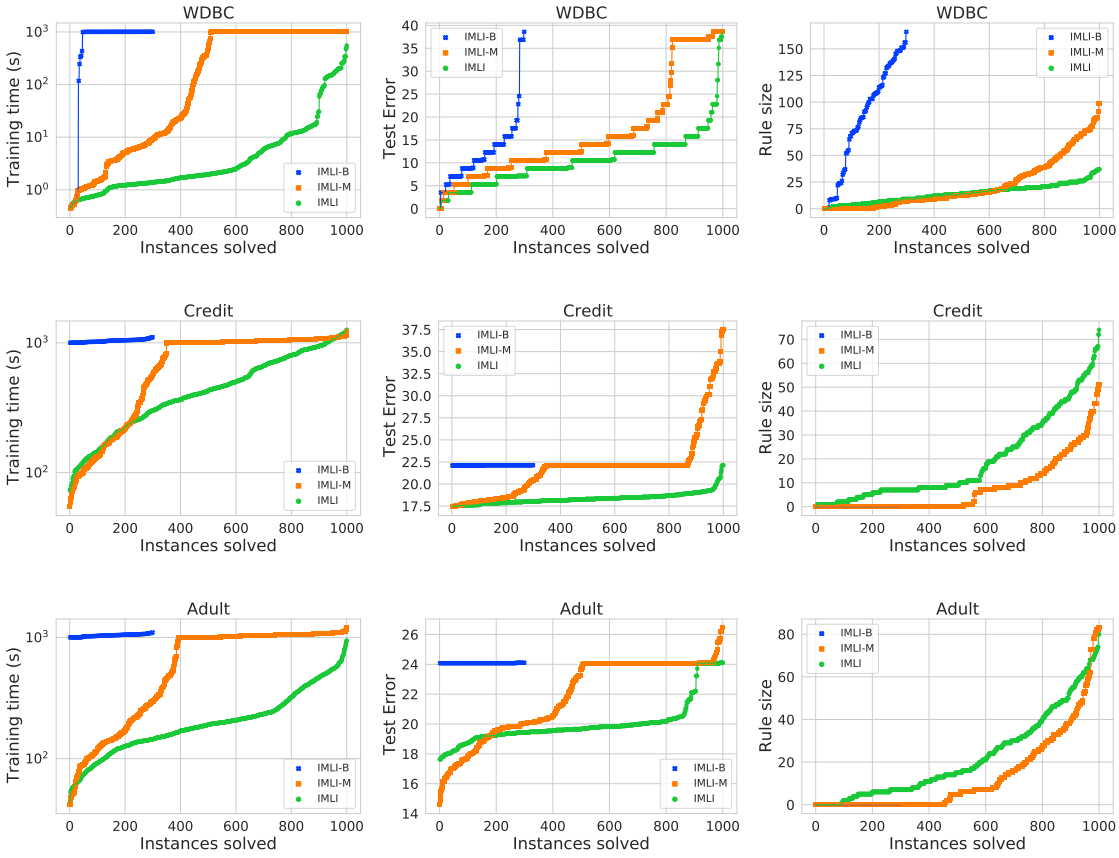

Figure 2: Comparison of training time, test error, and rule-size among different formulations presented in the paper. In each cactus plot, the incremental formulation IMLI with both mini-batch and iterative learning demonstrates the best performance in training time and test error than compared two formulations: non-incremental MaxSAT formulation IMLI-B and incremental formulation with only mini-batch learning IMLI-M. In terms of rule-size, IMLI often generates higher size rules than IMLI-M.

and returns the default rule "true" by predicting all samples as class 1. The other two formulations IMLI and IMLI-M demonstrate a similar trend in rule-size in all datasets with IMLI-M generating comparatively smaller size rules in Credit and Adult datasets. In this context, the improvement of rule-sparsity of IMLI-M is due to a comparatively higher test error (or lower accuracy) than IMLI as observed in all three datasets. Therefore, IMLI appears to be the best performing formulation w.r.t. training time, test error, and rule-size by balancing between accuracy and rule-size while being more scalable.

8.2.4 Performance Evaluation of Different Interpretable Representations in IMLI

We deploy IMLI to learn different interpretable rule-based representations: CNF and DNF classifiers, decision lists, and decision sets, and present their comparative performance w.r.t.

Table 3: Comparison of test accuracy (top value) and rule-size (bottom value) among different rule-based representations learned using IMLI. Numbers in bold denote the best performing results among different representations.

Dataset	CNF	DNF	Decision Sets	Decision Lists
Parkinsons	94.74 7.5	89.47 **6.0**	**94.87** 15.0	89.74 6.5
WDBC	94.74 11.5	**96.49** 15.0	95.61 15.5	95.61 **10.0**
Pima	**78.43** 23.0	77.13 **9.0**	76.97 15.0	76.97 13.5
Titanic	81.82 **5.5**	82.29 10.5	81.82 8.5	**82.3** 8.0
MAGIC	**78.26** **8.5**	77.44 41.5	75.87 10.0	77.79 14.0
Tom's HW	85.24 44.5	85.15 **26.5**	85.72 45.0	**85.95** 59.5
Credit	82.12 17.5	82.15 14.0	82.03 **9.5**	**82.22** 21.5
Adult	81.2 30.0	**84.28** 34.5	80.07 **7.0**	80.96 24.5
Bank Marketing	**89.84** 24.5	89.77 7.5	89.67 **6.0**	89.79 10.5
Connect-4	**75.36** 50.5	70.63 42.0	68.09 **4.5**	69.83 24.0
Weather AUS	83.78 22.0	**84.23** 14.0	83.69 **4.0**	83.85 26.0
Skin Seg	**94.71** 30.0	93.68 15.0	87.92 **3.0**	91.17 7.0

test accuracy and rule-size in Table 3. In each cell in this table, the top value represents the test accuracy and the bottom value represents the size of generated rules.

We learn all four interpretable representations on twelve datasets, where the CNF classifier appears to be the most accurate representation by achieving the highest accuracy in five datasets. In contrast, both DNF and decision lists achieve the highest accuracy in three datasets each; decision sets demonstrate the least performance in test accuracy by being more accurate in one dataset. To this end, the poor accuracy of decision sets is traded off by its rule-size as decision sets generate the sparsest rules compared to other representations. More precisely, decision sets have the smallest rule-size in six datasets, while CNF, DNF, and decision lists have the smallest rule-size in two, three, and one dataset, respectively. These results suggest that CNF classifiers are more favored in applications where higher accuracy is preferred, while decision sets are preferred in applications where

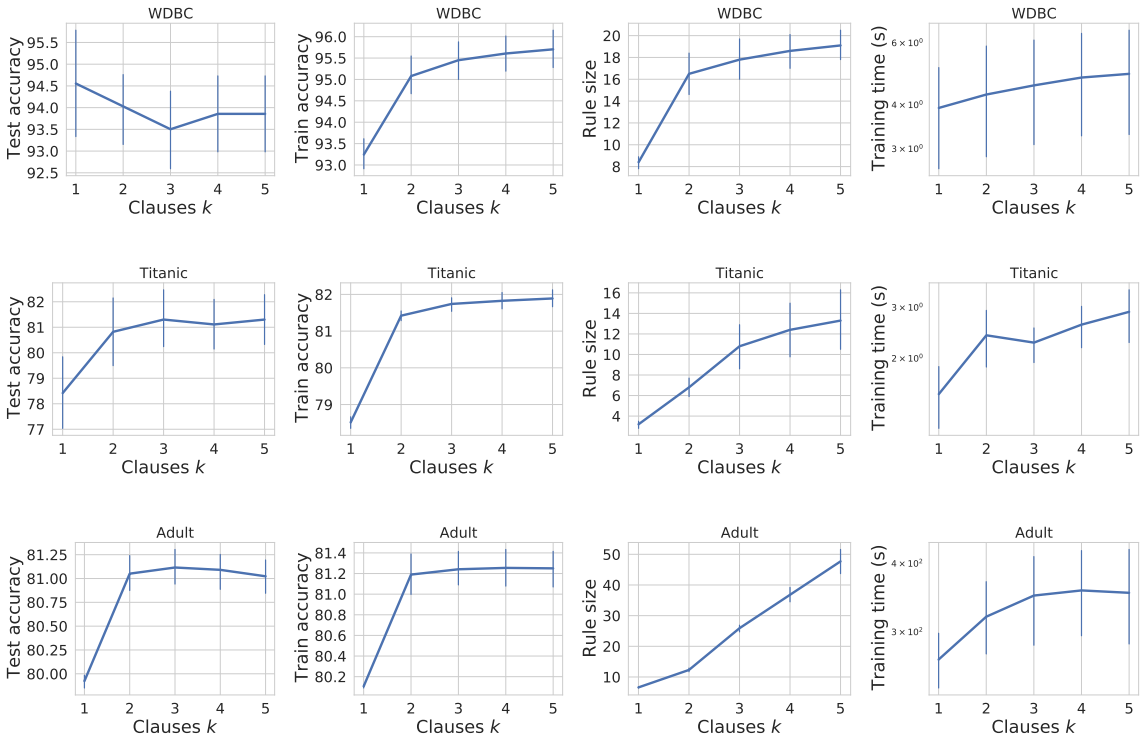

Figure 3: Effect of the number of clauses k on accuracy (test and train), rule-size, and training time. As k increases, both train and test accuracy of IMLI increase while generating rules with higher size by incurring higher training time.

higher interpretability is desired. In both cases, one could deploy IMLI for learning varied representations of classification rules.

8.2.5 Ablation Study

We experiment the effect of different hyper-parameters in IMLI on prediction accuracy, rule-size, and training time in different datasets. In the following, we discuss the impact of the number of clauses, regularization parameter, and size of mini-batches in IMLI.

Effect of the number of clauses k. In Figure 3, we vary k while learning CNF classifiers in IMLI. As k increases, both training and test accuracy generally increase in different datasets (plots in the first and second columns). Similarly, the size of rules increases with k by incurring higher training time (plots in the third and fourth columns). The reason is that a higher value of k allows more flexibility in fitting the data well by incurring more training time and generating higher size classification rules. Therefore, the number of clauses in IMLI provides control on training-time vs accuracy and also on accuracy vs rule-sparsity.

Effect of regularizer λ. In Figure 4, we vary λ in a logarithmic grid between 10^{-4} and 10^{1}. As stated in Eq. (5), a higher value of λ puts more priority on the minimal changes in rules between consecutive mini-batches in incremental learning while allowing higher mini-

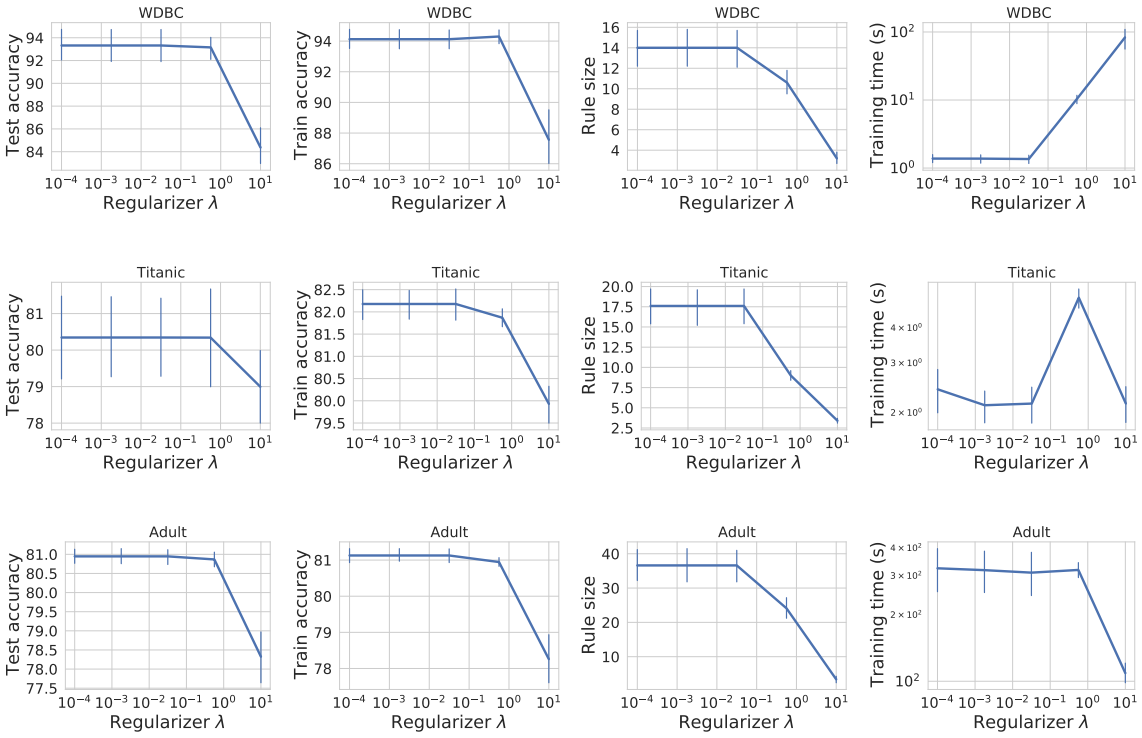

Figure 4: Effect of regularization λ on accuracy (test and train), rule-size, and training time. As λ increases, lower priority is given to accuracy. As a result, both training and test accuracy decrease with λ by generating smaller rules.

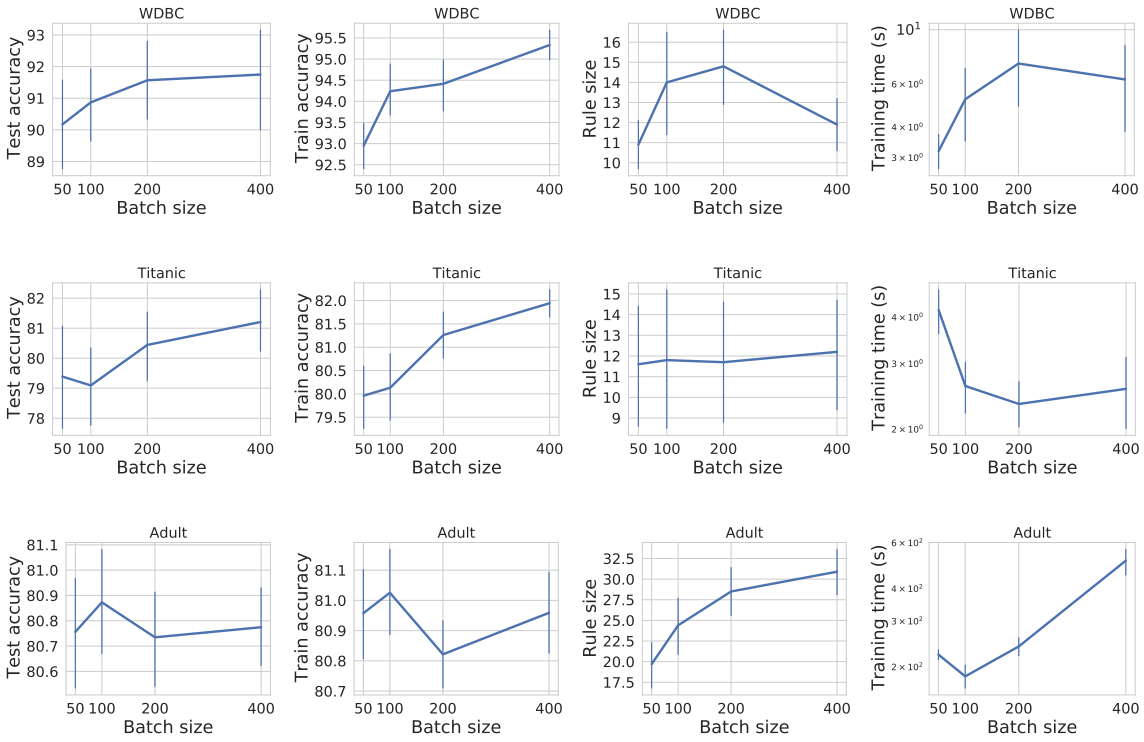

Figure 5: Effect of bath size on accuracy (test and train), rule-size, and training time. As we consider more samples in a mini-batch, IMLI generates more accurate and larger size classification rules.

batch training errors. Thus, in the first and second columns in Figure 4, as λ increases, both training and test accuracy gradually decrease. In addition, the size of rules (plots in the third column) also decreases. Finally, we observe that the training time generally decreases with λ. This observation indicates that higher λ puts lower computational load to the MaxSAT solver as a fraction of training examples is allowed to be misclassified. Thus, similar to the number of clauses, regularization parameter λ in IMLI allows to trade-off between accuracy and rule-size in a precise manner.

Effect of the size of mini-batch. In Figure 5, we present the effect of mini-batch size in IMLI. As we consider more samples in a batch, both test and training accuracy increase in general as presented in the first and second columns in Figure 5. Similarly, the size of generated rules also increases with the number of samples. Due to solving higher size MaxSAT queries, the training time also increases in general with an increase in mini-batch size. Therefore, by varying the size of mini-batches, IMLI allows controlling on training time vs the prediction accuracy (and rule-size) of generated rules.

9. Conclusion

Interpretable machine learning is gaining more focus with applications in many safety-critical domains. Considering the growing demand for interpretable models, it is challenging to design learning frameworks that satisfy all aspects: being accurate, interpretable, and scalable in practical classification tasks. In this paper, we have proposed a MaxSAT-based framework IMLI for learning interpretable rule-based classifiers expressible in CNF formulas. IMLI is built on an efficient integration of incremental learning, specifically mini-batch and iterative learning, with MaxSAT-based formulation. In our empirical evaluation, IMLI achieves the best balance among prediction accuracy, interpretability, and scalability. In particular, IMLI demonstrates competitive prediction accuracy and rule-size compared to existing interpretable rule-based classifiers. In addition, IMLI achieves impressive scalability than both interpretable and non-interpretable classifiers by learning interpretable rules on million-size datasets with higher accuracy. Finally, IMLI generates other popular interpretable classifiers such as decision lists and decision sets using the same framework.

Acknowledgments

Dmitry Malioutov was at the T.J. Watson IBM Research Center while performing this work. This work was supported in part by National Research Foundation Singapore under its NRF Fellowship Programme [NRF-NRFFAI1-2019-0004], Ministry of Education Singapore Tier 2 grant MOE-T2EP20121-0011, and Ministry of Education Singapore Tier 1 Grant [R-252-000-B59-114]. The computational work for this article was performed on resources of Max Planck Institute for Software Systems, Germany and the National Supercomputing Centre, Singapore `https://www.nscc.sg`.

Appendix A. Illustration of Interpretable CNF Classification Rules

In the following, we present representative CNF classifiers learned in different datasets. In each dataset, if an input satisfies the CNF formula, it is predicted class 1 and vice-versa.

Parkinsons

$0.3 \leq$ Average vocal fundamental frequency < 0.4 **OR** $0.5 \leq$ Average vocal fundamental frequency < 0.6 **OR** $0.1 \leq$ Maximum vocal fundamental frequency < 0.2 **OR** $0.5 \leq$ Minimum vocal fundamental frequency < 0.6 **OR** $0.1 \leq$ Shimmer:APQ5 < 0.2 **OR** $0.5 \leq$ DFA < 0.6 **OR** $0.3 \leq$ spread1 < 0.4 **OR** $0.8 \leq$ spread2 < 0.9 **OR** $0.3 \leq$ PPE < 0.4 **OR NOT** $-\infty \leq$ MDVP:APQ < 0.1 **AND**

NOT $0.8 \leq$ Average vocal fundamental frequency < 0.9

WDBC

$0.4 \leq$ perimeter < 0.5 **OR** $0.8 \leq$ symmetry < 0.9 **OR** $0.5 \leq$ largest concave points < 0.6 **OR** $0.6 \leq$ largest concave points < 0.7 **OR** $0.7 \leq$ largest concave points < 0.8 **OR** $0.6 \leq$ largest symmetry < 0.7 **OR NOT** $-\infty \leq$ area SE < 0.1 **AND**

$0.4 \leq$ texture < 0.5 **OR** $0.5 \leq$ texture < 0.6 **OR** $0.5 \leq$ perimeter < 0.6 **OR** $0.5 \leq$ smoothness < 0.6 **OR** $0.4 \leq$ concave points < 0.5 **OR** $0.5 \leq$ concave points < 0.6 **OR** $0.2 \leq$ largest concavity < 0.3 **OR** $0.6 \leq$ largest concave points < 0.7 **OR** $0.7 \leq$ largest concave points < 0.8 **OR** $0.4 \leq$ largest symmetry < 0.5 **OR** $0.6 \leq$ largest symmetry < 0.7

Pima

x1 $= 11$ **OR** x1 $= 14$ **OR** x1 $= 15$ **OR** $0.7 \leq$ x2 < 0.8 **OR** $0.8 \leq$ x2 < 0.9 **OR** $0.9 \leq$ x2 $< \infty$ **OR** $0.9 \leq$ x3 $< \infty$ **OR** $0.4 \leq$ x5 < 0.5 **OR** $0.7 \leq$ x5 < 0.8 **OR** $0.7 \leq$ x6 < 0.8 **OR** $0.4 \leq$ x7 < 0.5 **OR** $0.5 \leq$ x7 < 0.6 **OR** $0.9 \leq$ x7 $< \infty$ **AND**

x1 $= 7$ **OR** x1 $= 8$ **OR** $0.6 \leq$ x2 < 0.7 **OR** $0.8 \leq$ x2 < 0.9 **OR** $0.9 \leq$ x2 $< \infty$ **OR** $-\infty \leq$ x3 < 0.1 **OR** $0.7 \leq$ x3 < 0.8 **OR** $0.9 \leq$ x3 $< \infty$ **OR** $0.2 \leq$ x5 < 0.3 **OR** $0.7 \leq$ x6 < 0.8 **OR** $0.1 \leq$ x7 < 0.2 **OR** $0.3 \leq$ x7 < 0.4 **OR** $0.4 \leq$ x8 < 0.5 **OR** $0.8 \leq$ x8 < 0.9 **AND**

x1 $= 2$ **OR** x1 $= 6$ **OR** x1 $= 7$ **OR** x1 $= 9$ **OR** $-\infty \leq$ x3 < 0.1 **OR** $0.8 \leq$ x3 < 0.9 **OR** $0.1 \leq$ x5 < 0.2 **OR** $0.2 \leq$ x5 < 0.3 **OR** $0.7 \leq$ x5 < 0.8 **OR** $0.5 \leq$ x6 < 0.6 **OR** $0.6 \leq$ x6 < 0.7 **OR** $0.4 \leq$ x7 < 0.5 **OR** $0.9 \leq$ x7 $< \infty$ **OR** $0.1 \leq$ x8 < 0.2 **OR** $0.3 \leq$ x8 < 0.4 **OR** $0.6 \leq$ x8 < 0.7 **OR** $0.8 \leq$ x8 < 0.9 **AND**

x1 $= 8$ **OR** $0.5 \leq$ x2 < 0.6 **OR** $0.8 \leq$ x2 < 0.9 **OR** $0.9 \leq$ x2 $< \infty$ **OR** $0.7 \leq$ x3 < 0.8 **OR** $-\infty \leq$ x4 < 0.1 **OR** $0.1 \leq$ x5 < 0.2 **OR** $0.3 \leq$ x5 < 0.4 **OR** $0.5 \leq$ x7 < 0.6 **OR** $0.1 \leq$ x8 < 0.2 **OR** $0.4 \leq$ x8 < 0.5 **OR** $0.8 \leq$ x8 < 0.9

Titanic

$-\infty \leq$ age < 0.1 **OR** $0.9 \leq$ age $< \infty$ **OR** $0.9 \leq$ fare $< \infty$ **OR NOT** sex **AND**

passenger-class $= 2$ **OR** $0.4 \leq$ age < 0.5 **OR** $0.6 \leq$ age < 0.7 **OR** siblings-or-spouces-aboard $= 0$ **OR** $0.1 \leq$ fare < 0.2 **OR** $0.5 \leq$ fare < 0.6 **OR** embarked $=$ C **AND**

$-\infty \leq$ age < 0.1 **OR** $0.1 \leq$ age < 0.2 **OR** $0.7 \leq$ age < 0.8 **OR** siblings-or-spouces-aboard $= 1$ **OR** parents-or-childred-aboard $= 1$ **OR** embarked $=$ C **OR NOT** passenger-class $=$ 3 **AND**

$-\infty \leq$ age < 0.1 **OR** $0.2 \leq$ age < 0.3 **OR** $0.3 \leq$ age < 0.4 **OR** parents-or-childred-aboard $= 0$ **OR NOT** passenger-class $= 3$ **OR NOT** siblings-or-spouces-aboard $= 0$ **AND**

NOT passenger-class $= 2$ **OR NOT** $0.7 \leq$ age < 0.8

MAGIC

$-\infty \leq$ length < 0.1 **OR** $-\infty \leq$ size < 0.1 **OR** $0.8 \leq$ conc < 0.9 **OR** $-\infty \leq$ alpha < 0.1 **OR** $0.1 \leq$ alpha < 0.2 **OR** $0.2 \leq$ dist < 0.3

Tom's HW

$0.4 \leq$ x10 < 0.5 **OR** $0.1 \leq$ x11 < 0.2 **OR** $0.4 \leq$ x11 < 0.5 **OR** $0.4 \leq$ x12 < 0.5 **OR** $0.3 \leq$ x13 < 0.4 **OR** $0.6 \leq$ x13 < 0.7 **OR** $0.3 \leq$ x15 < 0.4 **OR** $0.6 \leq$ x15 < 0.7 **OR** $0.4 \leq$ x16 < 0.5 **OR** $0.1 \leq$ x57 < 0.2 **OR** $0.5 \leq$ x59 < 0.6 **OR** $0.3 \leq$ x60 < 0.4 **OR** $0.2 \leq$ x62 < 0.3 **OR** $0.2 \leq$ x73 < 0.3 **OR** $0.1 \leq$ x74 < 0.2 **OR NOT** $-\infty \leq$ x74 < 0.1 **OR NOT** $-\infty \leq$ x96 < 0.1 **AND**

$0.4 \leq$ x9 < 0.5 **OR** $0.8 \leq$ x9 < 0.9 **OR** $0.2 \leq$ x12 < 0.3 **OR** $0.3 \leq$ x12 < 0.4 **OR** $0.3 \leq$ x15 < 0.4 **OR** $0.6 \leq$ x15 < 0.7 **OR** $0.1 \leq$ x60 < 0.2 **OR** $0.1 \leq$ x61 < 0.2 **OR** $0.6 \leq$ x78 < 0.7 **OR NOT** $-\infty \leq$ x11 < 0.1 **OR NOT** $-\infty \leq$ x13 < 0.1 **OR NOT** $-\infty \leq$ x46 < 0.1 **OR NOT** $-\infty \leq$ x64 < 0.1 **OR NOT** $-\infty \leq$ x72 < 0.1 **OR NOT** $-\infty \leq$ x77 < 0.1 **AND**

$-\infty \leq$ x50 < 0.1 **OR** $0.1 \leq$ x57 < 0.2 **AND**

$-\infty \leq$ x49 < 0.1 **OR NOT** $-\infty \leq$ x17 < 0.1 **AND**

$0.3 \leq$ x14 < 0.4 **OR** $0.6 \leq$ x74 < 0.7 **OR** $0.9 \leq$ x79 $< \infty$ **OR NOT** $-\infty \leq$ x9 < 0.1 **OR NOT** $-\infty \leq$ x15 < 0.1 **OR NOT** $-\infty \leq$ x73 < 0.1 **OR NOT** $-\infty \leq$ x80 < 0.1

Credit

Repayment-status-in-September $= 2$ **OR** Repayment-status-in-September $= 3$ **OR** Repayment-status-in-August $= 3$ **OR** Repayment-status-in-May $= 3$ **OR** Repayment-status-in-May $=$ 5 **OR** Repayment-status-in-April $= 3$ **AND**

$0.4 \leq$ Age < 0.5 **OR** Repayment-status-in-September $= 1$ **OR** Repayment-status-in-August $= 0$ **OR** Repayment-status-in-July $= 2$ **OR** Repayment-status-in-May $= 2$ **OR** Repayment-status-in-April $= 7$ **OR** $0.3 \leq$ Amount-of-bill-statement-in-April < 0.4 **OR NOT** Repayment-status-in-May $= -1$ **OR NOT** $-\infty \leq$ Amount-of-bill-statement-in-May < 0.1 **AND**

Gender **OR** $-\infty \leq$ Age < 0.1 **OR** $0.3 \leq$ Age < 0.4 **OR** Repayment-status-in-September $= 1$ **OR** $0.1 \leq$ Amount-of-bill-statement-in-August < 0.2 **OR NOT** Education $= 3$ **OR NOT** $0.2 \leq$ Amount-of-bill-statement-in-September < 0.3

Adult

education $=$ Doctorate **OR** education $=$ Prof-school **OR** $0.1 \leq$ capital-gain < 0.2 **OR** $0.2 \leq$ capital-gain < 0.3 **OR** $0.9 \leq$ capital-gain $< \infty$ **OR** $0.4 \leq$ capital-loss < 0.5 **OR** $0.5 \leq$ capital-loss < 0.6 **AND**

relationship = Husband **OR** relationship = Wife **OR** 0.5 ≤ capital-loss < 0.6 **OR** 0.6 ≤ capital-loss < 0.7 **OR** 0.8 ≤ capital-loss < 0.9 **OR NOT** -∞ ≤ capital-gain < 0.1 **AND**

0.1 ≤ age < 0.2 **OR** 0.7 ≤ age < 0.8 **OR** workclass = State-gov **OR** education = Bachelors **OR** marital-status = Separated **OR** occupation = Exec-managerial **OR** occupation = Farming-fishing **OR** occupation = Prof-specialty **OR** occupation = Protective-serv **OR** occupation = Tech-support **OR** relationship = Unmarried **OR** 0.4 ≤ capital-loss < 0.5 **OR** 0.6 ≤ capital-loss < 0.7 **OR** 0.2 ≤ hours-per-week < 0.3 **OR** 0.4 ≤ hours-per-week < 0.5 **OR** 0.7 ≤ hours-per-week < 0.8 **OR** 0.9 ≤ hours-per-week < ∞ **OR NOT** -∞ ≤ capital-gain < 0.1

Bank Marketing

0.7 ≤ age < 0.8 **OR** 0.2 ≤ duration < 0.3 **OR** 0.3 ≤ duration < 0.4 **OR** 0.4 ≤ duration < 0.5 **OR** 0.5 ≤ duration < 0.6 **OR** poutcome = success **AND**

housing **OR** 0.8 ≤ age < 0.9 **OR** job = admin. **OR** job = management **OR** job = self-employed **OR** job = services **OR** job = unemployed **OR** 0.2 ≤ duration < 0.3 **OR** 0.1 ≤ campaign < 0.2 **OR** poutcome = success **AND**

job = services **OR** 0.2 ≤ balance < 0.3 **OR** contact = cellular **OR** 0.1 ≤ duration < 0.2 **OR** 0.2 ≤ duration < 0.3 **OR** 0.2 ≤ pdays < 0.3 **OR** poutcome = unknown **OR NOT** -∞ ≤ balance < 0.1 **OR NOT** -∞ ≤ previous < 0.1 **AND**

0.2 ≤ age < 0.3 **OR** 0.3 ≤ age < 0.4 **OR** 0.8 ≤ age < 0.9 **OR** job = management **OR** job = student **OR** job = unemployed **OR** education = secondary **OR** 0.1 ≤ balance < 0.2 **OR** 0.1 ≤ duration < 0.2 **OR** -∞ ≤ pdays < 0.1 **OR** 0.2 ≤ pdays < 0.3 **OR** poutcome = unknown

Connect-4

b2 = 1 **OR** c2 = 1 **OR** d2 = 1 **OR** d4 = 1 **OR** d5 = 0 **OR** e2 = 1 **OR** f3 = 0 **OR NOT** d1 = 0 **AND**

b2 = 1 **OR** b3 = 1 **OR** b4 = 1 **OR** d4 = 1 **OR** f2 = 1 **OR NOT** d3 = 0 **OR NOT** f3 = 2 **AND**

c1 = 2 **OR** c2 = 2 **OR** c3 = 1 **OR** c3 = 2 **OR** c4 = 1 **OR** c6 = 0 **OR** d2 = 1 **OR** d4 = 1 **OR** e2 = 1 **OR** e3 = 1 **OR** f4 = 1 **OR NOT** a3 = 2 **OR NOT** b6 = 2 **AND**

a1 = 0 **OR** a2 = 0 **OR** a6 = 0 **OR** b2 = 1 **OR** b4 = 1 **OR** b5 = 0 **OR** c2 = 1 **OR** c4 = 1 **OR** c5 = 0 **OR** d1 = 1 **OR** d2 = 1 **OR** e2 = 1 **OR** g1 = 0 **OR** g2 = 0 **OR NOT** d5 = 2 **OR NOT** d6 = 2 **AND**

b2 = 1 **OR** b4 = 1 **OR** c3 = 1 **OR** d3 = 1 **OR** e2 = 1 **OR** f2 = 1 **OR** f3 = 1 **OR** g3 = 0 **OR** g5 = 0 **OR NOT** c2 = 0 **OR NOT** c4 = 2 **OR NOT** d2 = 2

Weather AUS

$0.7 \leq$ Rainfall < 0.8 **OR** $0.8 \leq$ Humidity3pm < 0.9 **OR** $0.9 \leq$ Humidity3pm $< \infty$ **AND**

RainToday **OR** $0.5 \leq$ WindGustSpeed < 0.6 **OR** $0.7 \leq$ Humidity9am < 0.8 **OR** $0.9 \leq$ Humidity3pm $< \infty$ **OR** $0.4 \leq$ Pressure3pm < 0.5 **OR NOT** $0.9 \leq$ Humidity9am $< \infty$ **AND**

$0.1 \leq$ MinTemp < 0.2 **OR** $0.8 \leq$ MinTemp < 0.9 **OR** $0.9 \leq$ WindSpeed9am $< \infty$ **OR** $0.9 \leq$ Humidity9am $< \infty$ **OR** $0.8 \leq$ Humidity3pm < 0.9 **OR** $0.9 \leq$ Humidity3pm $< \infty$ **OR** $0.5 \leq$ Temp3pm < 0.6 **OR NOT** $0.7 \leq$ Rainfall < 0.8 **AND**

WindGustDir = NNW **OR** WindDir3pm = W **OR NOT** $0.8 \leq$ Temp3pm < 0.9 **AND**

WindGustDir = NNW **OR** WindGustDir = NW **OR** $0.1 \leq$ Pressure9am < 0.2 **OR NOT** $0.7 \leq$ Temp3pm < 0.8

Vote

NOT physician-fee-freeze **AND**

NOT adoption-of-the-budget-resolution **OR NOT** anti-satellite-test-ban **OR NOT** synfuels-corporation-cutback **AND**

adoption-of-the-budget-resolution **OR** el-salvador-aid **OR NOT** duty-free-exports **AND**

mx-missile **OR NOT** adoption-of-the-budget-resolution **OR NOT** el-salvador-aid **OR NOT** anti-satellite-test-ban **AND**

adoption-of-the-budget-resolution **OR** mx-missile **OR NOT** synfuels-corporation-cutback **OR NOT** education-spending

Skin Seg

$0.2 \leq$ Red < 0.3 **OR** $0.3 \leq$ Red < 0.4 **OR** $0.4 \leq$ Red < 0.5 **OR** $0.9 \leq$ Red $< \infty$ **OR** $0.9 \leq$ Green $< \infty$ **OR** $0.4 \leq$ Blue < 0.5 **OR NOT** $0.9 \leq$ Blue $< \infty$ **AND**

$0.2 \leq$ Red < 0.3 **OR** $0.7 \leq$ Red < 0.8 **OR** $0.8 \leq$ Red < 0.9 **OR** $0.9 \leq$ Red $< \infty$ **OR NOT** $0.8 \leq$ Blue < 0.9 **AND**

$0.7 \leq$ Red < 0.8 **OR** $0.8 \leq$ Red < 0.9 **OR** $0.9 \leq$ Red $< \infty$ **OR** $-\infty \leq$ Green < 0.1

OR NOT 0.7 ≤ Blue < 0.8

BNG(labor)

0.5 ≤ wage-increase-first-year < 0.6 **OR** pension = empl-contr **OR** contribution-to-dental-plan = full **AND**

0.6 ≤ wage-increase-first-year < 0.7 **OR** 0.7 ≤ wage-increase-second-year < 0.8 **OR** 0.5 ≤ shift-differential < 0.6 **OR NOT** longterm-disability-assistance **AND**

0.2 ≤ wage-increase-first-year < 0.3 **OR** 0.3 ≤ wage-increase-first-year < 0.4 **OR** 0.4 ≤ wage-increase-first-year < 0.5 **OR** 0.5 ≤ wage-increase-first-year < 0.6 **OR** 0.6 ≤ wage-increase-first-year < 0.7 **OR** 0.7 ≤ wage-increase-first-year < 0.8 **OR** 0.6 ≤ wage-increase-second-year < 0.7 **OR** cost-of-living-adjustment = tcf **OR** 0.3 ≤ working-hours < 0.4 **OR** 0.4 ≤ working-hours < 0.5 **OR** 0.5 ≤ working-hours < 0.6 **OR** pension = ret-allw **OR** 0.7 ≤ standby-pay < 0.8 **OR** contribution-to-dental-plan = full **OR** contribution-to-health-plan = half **OR NOT** education-allowance **AND**

0.5 ≤ shift-differential < 0.6 **OR** contribution-to-dental-plan = full **OR** contribution-to-dental-plan = half **OR** contribution-to-health-plan = half **OR** contribution-to-health-plan = none **AND**

0.3 ≤ wage-increase-first-year < 0.4 **OR** 0.6 ≤ wage-increase-first-year < 0.7 **OR** 0.7 ≤ wage-increase-second-year < 0.8 **OR** 0.4 ≤ working-hours < 0.5 **OR** pension = empl-contr **OR** pension = ret-allw **OR** 0.9 ≤ statutory-holidays < inf

BNG(credit-g)

foreign-worker **OR** checking-status = 'no checking' **OR** checking-status = ¿=200 **OR** -∞ ≤ duration < 0.1 **OR** 0.1 ≤ duration < 0.2 **OR** credit-history = 'critical/other existing credit' **OR** purpose = 'used car' **OR** savings-status = 'no known savings' **OR** savings-status = ¿=1000 **OR** other-parties = guarantor **AND**

foreign-worker **OR** checking-status = 'no checking' **OR** checking-status = ¿=200 **OR** credit-history = 'delayed previously' **OR** purpose = 'used car' **OR** purpose = radio/tv **OR** other-parties = guarantor **OR** 0.4 ≤ age < 0.5 **OR NOT** other-payment-plans = bank **AND**

checking-status = 'no checking' **OR** credit-history = 'critical/other existing credit' **OR** purpose = 'used car' **OR** purpose = radio/tv **OR** purpose = retraining **OR** employment = 4≤X<7 **OR** property-magnitude = 'real estate' **OR NOT** checking-status = <0

References

Agarap, A. F. M. (2018). On breast cancer detection: an application of machine learning algorithms on the wisconsin diagnostic dataset. In *Proceedings of the 2nd international conference on machine learning and soft computing*, pp. 5–9.

Alos, J., Ansotegui, C., & Torres, E. (2021). Learning optimal decision trees using MaxSAT..

Angelino, E., Larus-Stone, N., Alabi, D., Seltzer, M., & Rudin, C. (2017). Learning certifiably optimal rule lists for categorical data..

Argelich, J., Li, C.-M., Manya, F., & Planes, J. (2008). The first and second Max-SAT evaluations.. Vol. 4, pp. 251–278. IOS Press.

Augasta, M. G., & Kathirvalavakumar, T. (2012). Rule extraction from neural networks—a comparative study. In *International Conference on Pattern Recognition, Informatics and Medical Engineering (PRIME-2012)*, pp. 404–408. IEEE.

Balyo, T., Heule, M., & Jarvisalo, M. (2017). SAT competition 2016: Recent developments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31.

Barakat, N., & Diederich, J. (2004). Learning-based rule-extraction from support vector machines.. not found.

Barakat, N., & Diederich, J. (2005). Eclectic rule-extraction from support vector machines.. Vol. 2, pp. 59–62. Citeseer.

Berg, O. J., Hyttinen, A. J., & Järvisalo, M. J. (2019). Applications of MaxSAT in data analysis.. EasyChair Publications.

Bessiere, C., Hebrard, E., & O'Sullivan, B. (2009). Minimising decision tree size as combinatorial optimisation. In *International Conference on Principles and Practice of Constraint Programming*, pp. 173–187. Springer.

Bhagoji, A. N., Cullina, D., Sitawarin, C., & Mittal, P. (2018). Enhancing robustness of machine learning systems via data transformations. In *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–5. IEEE.

Cauwenberghs, G., & Poggio, T. (2001). Incremental and decremental support vector machine learning. In *Proc. of NIPS*.

Chen, Z., & Liu, B. (2018). Lifelong machine learning.. Vol. 12, pp. 1–207. Morgan & Claypool Publishers.

Clark, P., & Niblett, T. (1989). The CN2 induction algorithm..

Cohen, W. W., & Singer, Y. (1999). A simple, fast, and effective rule learner. In *Proc. of AAAI*, Orlando, FL.

Cohen, W. W. (1995). Fast effective rule induction. In *Machine learning proceedings 1995*, pp. 115–123. Elsevier.

Dash, S., & Goncalves, J. (2021). LPRules: Rule induction in knowledge graphs using linear programming..

Diederich, J. (2008). Rule extraction from support vector machines: An introduction. In *Rule extraction from support vector machines*, pp. 3–31. Springer.

Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning..

Du, M., Liu, N., & Hu, X. (2019). Techniques for interpretable machine learning.. Vol. 63, pp. 68–77. ACM New York, NY, USA.

Dua, D., & Graff, C. (2017). UCI machine learning repository..

Erickson, B. J., Korfiatis, P., Akkus, Z., & Kline, T. L. (2017). Machine learning for medical imaging.. Vol. 37, pp. 505–515. Radiological Society of North America.

Fisher, M. L. (1981). The lagrangian relaxation method for solving integer programming problems.. Vol. 27, pp. 1–18. INFORMS.

Fürnkranz, J. (1999). Separate-and-conquer rule learning.. Vol. 13, pp. 3–54. Springer.

Gage, B. F., Waterman, A. D., Shannon, W., Boechler, M., Rich, M. W., & Radford, M. J. (2001). Validation of clinical classification schemes for predicting stroke: results from the national registry of atrial fibrillation.. Vol. 285, pp. 2864–2870. American Medical Association.

Ghosh, B., & Meel, K. S. (2019). Imli: An incremental framework for maxsat-based learning of interpretable classification rules. In *Proc. of AIES*.

Gill, N., Hall, P., Montgomery, K., & Schmidt, N. (2020). A responsible machine learning workflow with focus on interpretable models, post-hoc explanation, and discrimination testing.. Vol. 11, p. 137. Multidisciplinary Digital Publishing Institute.

Guidotti, R., Monreale, A., Ruggieri, S., Pedreschi, D., Turini, F., & Giannotti, F. (2018). Local rule-based explanations of black box decision systems..

Hailesilassie, T. (2016). Rule extraction algorithm for deep neural networks: A review..

Hancox-Li, L. (2020). Robustness in machine learning explanations: does it matter?. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pp. 640–647.

Hinton, G., Srivastava, N., & Swersky, K. (2012). Neural networks for machine learning lecture 6a overview of mini-batch gradient descent.. Vol. 14, p. 2.

Holstein, K., Wortman Vaughan, J., Daumé III, H., Dudik, M., & Wallach, H. (2019). Improving fairness in machine learning systems: What do industry practitioners need?. In *Proceedings of the 2019 CHI conference on human factors in computing systems*, pp. 1–16.

Ignatiev, A., Lam, E., Stuckey, P. J., & Marques-Silva, J. (2021a). A scalable two stage approach to computing optimal decision sets..

Ignatiev, A., Marques-Silva, J., Narodytska, N., & Stuckey, P. J. (2021b). Reasoning-based learning of interpretable ML models. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Ignatiev, A., Pereira, F., Narodytska, N., & Marques-Silva, J. (2018). A SAT-based approach to learn explainable decision sets. In *International Joint Conference on Automated Reasoning*, pp. 627–645. Springer.

Izza, Y., Ignatiev, A., & Marques-Silva, J. (2020). On explaining decision trees..

Janota, M., & Morgado, A. (2020). SAT-based encodings for optimal decision trees with explicit paths. In *International Conference on Theory and Applications of Satisfiability Testing*, pp. 501–518. Springer.

Johnson, J. K., Malioutov, D. M., & Willsky, A. S. (2007). Lagrangian relaxation for MAP estimation in graphical models..

Kaissis, G. A., Makowski, M. R., Rückert, D., & Braren, R. F. (2020). Secure, privacy-preserving and federated machine learning in medical imaging.. Vol. 2, pp. 305–311. Nature Publishing Group.

Konečnỳ, J., McMahan, B., & Ramage, D. (2015). Federated optimization: Distributed optimization beyond the datacenter..

Konečnỳ, J., McMahan, H. B., Ramage, D., & Richtárik, P. (2016). Federated optimization: Distributed machine learning for on-device intelligence..

Kononenko, I. (2001). Machine learning for medical diagnosis: history, state of the art and perspective.. Vol. 23, pp. 89–109. Elsevier.

Kumar, R. S. S., O'Brien, D. R., Albert, K., & Vilojen, S. (2018). Law and adversarial machine learning..

Lakkaraju, H., Bach, S. H., & Leskovec, J. (2016). Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1675–1684.

Lakkaraju, H., Kamar, E., Caruana, R., & Leskovec, J. (2017). Interpretable & explorable approximations of black box models..

Lakkaraju, H., Kamar, E., Caruana, R., & Leskovec, J. (2019). Faithful and customizable explanations of black box models. In *Proc. of AIES*.

Lemaréchal, C. (2001). Lagrangian relaxation. In *Computational combinatorial optimization*, pp. 112–156. Springer.

Letham, B., Rudin, C., McCormick, T. H., & Madigan, D. (2015). Interpretable classifiers using rules and Bayesian analysis: Building a better stroke prediction model.. Vol. 9, pp. 1350–1371. Institute of Mathematical Statistics.

Li, M., Zhang, T., Chen, Y., & Smola, A. J. (2014). Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 661–670.

Lin, P.-C. K., & Khatri, S. P. (2012). Application of Max-SAT-based ATPG to optimal cancer therapy design.. Vol. 13, pp. 1–10. Springer.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems*, pp. 4768–4777.

Malioutov, D., & Meel, K. S. (2018). MLIC: A maxsat-based framework for learning interpretable classification rules. In *Proceedings of International Conference on Constraint Programming (CP)*.

Malioutov, D., & Varshney, K. (2013). Exact rule learning via boolean compressed sensing. In *International Conference on Machine Learning*, pp. 765–773. PMLR.

Martens, D., Huysmans, J., Setiono, R., Vanthienen, J., & Baesens, B. (2008). Rule extraction from support vector machines: an overview of issues and application in credit scoring.. pp. 33–63. Springer.

Martins, R., Manquinho, V., & Lynce, I. (2014). Open-WBO: A modular MaxSAT solver. In *International Conference on Theory and Applications of Satisfiability Testing*, pp. 438–445. Springer.

Masters, D., & Luschi, C. (2018). Revisiting small batch training for deep neural networks..

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., & Galstyan, A. (2021). A survey on bias and fairness in machine learning.. Vol. 54, pp. 1–35. ACM New York, NY, USA.

Molnar, C. (2020). *Interpretable machine learning*. Lulu. com.

Moradi, M., & Samwald, M. (2021). Post-hoc explanation of black-box classifiers using confident itemsets.. Vol. 165, p. 113941. Elsevier.

Murdoch, W. J., Singh, C., Kumbier, K., Abbasi-Asl, R., & Yu, B. (2019). Interpretable machine learning: definitions, methods, and applications..

Myers, I. B. (1962). The Myers-Briggs type indicator: Manual (1962)... Consulting Psychologists Press.

Narodytska, N., Ignatiev, A., Pereira, F., Marques-Silva, J., & RAS, I. (2018). Learning optimal decision trees with SAT. In *IJCAI*, pp. 1362–1368.

Núñez, H., Angulo, C., & Català, A. (2002). Rule extraction from support vector machines.. In *Esann*, pp. 107–112.

Pastor, E., & Baralis, E. (2019). Explaining black box models by means of local rules. In *Proceedings of the 34th ACM/SIGAPP symposium on applied computing*, pp. 510–517.

Peled, I., Rodrigues, F., & Pereira, F. C. (2019). Model-based machine learning for transportation. In *Mobility patterns, big data and transport analytics*, pp. 145–171. Elsevier.

Philipp, T., & Steinke, P. (2015). Pblib–a library for encoding pseudo-boolean constraints into cnf. In *International Conference on Theory and Applications of Satisfiability Testing*, pp. 9–16. Springer.

Quinlan, J. R. (1986). Induction of decision trees.. Vol. 1, pp. 81–106. Springer.

Quinlan, J. R. (1987). Simplifying decision trees.. Vol. 27, pp. 221–234. Elsevier.

Quinlan, J. R. (1993). C4. 5: Programming for machine learning..

Rajapaksha, D., Bergmeir, C., & Buntine, W. (2020). LoRMIkA: Local rule-based model interpretability with K-optimal associations.. Vol. 540, pp. 221–241. Elsevier.

Ralaivola, L., & d'Alché Buc, F. (2001). Incremental support vector machine learning: A local approach. In *Proc. of ICANN*.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2018). Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

Rivest, R. L. (1987). Learning decision lists.. Vol. 2, pp. 229–246. Springer.

Robinson, N., Gretton, C., Pham, D. N., & Sattar, A. (2010). Partial weighted MaxSAT for optimal planning. In *Pacific rim international conference on artificial intelligence*, pp. 231–243. Springer.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead.. Vol. 1, pp. 206–215. Nature Publishing Group.

Ruping, S. (2001). Incremental learning with support vector machines. In *Proc. of ICDM*.

Sato, M., & Tsukimoto, H. (2001). Rule extraction from neural networks via decision tree induction. In *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222)*, Vol. 3, pp. 1870–1875. IEEE.

Schidler, A., & Szeider, S. (2021). SAT-based decision tree learning for large data sets. In *Proceedings of AAAI*, Vol. 21.

Setiono, R., & Liu, H. (1995). Understanding neural networks via rule extraction. In *IJCAI*, Vol. 1, pp. 480–485. Citeseer.

Shati, P., Cohen, E., & McIlraith, S. (2021). SAT-based approach for learning optimal decision trees with non-binary features. In *27th International Conference on Principles and Practice of Constraint Programming (CP 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

Silva, J. P. M., & Sakallah, K. A. (2003). GRASP—a new search algorithm for satisfiability. In *The Best of ICCAD*, pp. 73–89. Springer.

Slack, D., Hilgard, S., Jia, E., Singh, S., & Lakkaraju, H. (2020). Fooling LIME and SHAP: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pp. 180–186.

Su, G., Wei, D., Varshney, K. R., & Malioutov, D. M. (2016). Learning sparse two-level Boolean rules. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6. IEEE.

Surden, H. (2014). Machine learning and law.. Vol. 89, p. 87. HeinOnline.

Syed, N. A., Huan, S., Kah, L., & Sung, K. (1999). Incremental learning with support vector machines.. Citeseer.

Vanschoren, J., van Rijn, J. N., Bischl, B., & Torgo, L. (2013). OpenML: Networked science in machine learning.. Vol. 15, pp. 49–60, New York, NY, USA. ACM.

Walter, R., Zengler, C., & Küchlin, W. (2013). Applications of MaxSAT in automotive configuration.. In *Configuration Workshop*, Vol. 1, p. 21. Citeseer.

Wang, F., & Rudin, C. (2015). Falling rule lists. In *Artificial Intelligence and Statistics*, pp. 1013–1022. PMLR.

Wang, T., Rudin, C., Doshi-Velez, F., Liu, Y., Klampfl, E., & MacNeille, P. (2017). A Bayesian framework for learning rule sets for interpretable classification.. Vol. 18, pp. 2357–2393. JMLR. org.

Yu, J., Ignatiev, A., Bodic, P. L., & Stuckey, P. J. (2020a). Optimal decision lists using SAT..

Yu, J., Ignatiev, A., Stuckey, P. J., & Le Bodic, P. (2020b). Computing optimal decision sets with SAT. In *International Conference on Principles and Practice of Constraint Programming*, pp. 952–970. Springer.

Zantalis, F., Koulouras, G., Karabetsos, S., & Kandris, D. (2019). A review of machine learning and IoT in smart transportation.. Vol. 11, p. 94. Multidisciplinary Digital Publishing Institute.

Zhou, Z.-H. (2004). Rule extraction: Using neural networks or for neural networks?.. Vol. 19, pp. 249–253. Springer.

Zilke, J. R., Mencía, E. L., & Janssen, F. (2016). Deepred–rule extraction from deep neural networks. In *International Conference on Discovery Science*, pp. 457–473. Springer.