Journal of Artificial Intelligence Research 72 (2021) 1251-1279 Submitted 02/2021; published 12/2021

Learning Optimal Decision Sets and Lists with SAT

Jinqiang Yu JINQIANG.YU@MONASH.EDU
Alexey Ignatiev ALEXEY.IGNATIEV@MONASH.EDU
Peter J. Stuckey PETER.STUCKEY@MONASH.EDU
Pierre Le Bodic PIERRE.LEBODIC@MONASH.EDU
Department of Data Science and Artificial Intelligence
Monash University, Australia

Abstract

Decision sets and decision lists are two of the most easily explainable machine learning models. Given the renewed emphasis on explainable machine learning decisions, both of these machine learning models are becoming increasingly attractive, as they combine small size and clear explainability. In this paper, we define size as the total number of literals in the SAT encoding of these rule-based models as opposed to earlier work that concentrates on the number of rules. In this paper, we develop approaches to computing minimum-size "perfect" decision sets and decision lists, which are perfectly accurate on the training data, and minimal in size, making use of modern SAT solving technology. We also provide a new method for determining optimal sparse alternatives, which trade off size and accuracy. The experiments in this paper demonstrate that the optimal decision sets computed by the SAT-based approach are comparable with the best heuristic methods, but much more succinct, and thus, more explainable. We contrast the size and test accuracy of optimal decisions lists versus optimal decision sets, as well as other state-of-the-art methods for determining optimal decision lists. Finally, we examine the size of average explanations generated by decision sets and decision lists.

1. Introduction

With the increasing use of Machine Learning (ML) models to automate decisions[1], there has been an upsurge in interest in *explainable artificial intelligence* (XAI)[2] where these models can explain, in a manner understandable by humans, why they made a decision. This in turn has led to a re-examination of machine learning models that are implicitly easy to explain.

1. (Jordan & Mitchell, 2015; LeCun, Bengio, & Hinton, 2015; Mnih, Kavukcuoglu, Silver, Rusu, Veness, Bellemare, Graves, Riedmiller, Fidjeland, Ostrovski, et al., 2015)
2. (Baehrens, Schroeter, Harmeling, Kawanabe, Hansen, & Müller, 2010; Ribeiro, Singh, & Guestrin, 2016; Doshi-Velez & Kim, 2017; Lundberg & Lee, 2017; Ribeiro, Singh, & Guestrin, 2018; Shih, Choi, & Darwiche, 2018; Li, Liu, Chen, & Rudin, 2018; Montavon, Samek, & Müller, 2018; Lipton, 2018; Monroe, 2018; Evans & Grefenstette, 2018; Ignatiev, Narodytska, & Marques-Silva, 2019b, 2019c; Guidotti, Monreale, Ruggieri, Turini, Giannotti, & Pedreschi, 2019b; Guidotti, Monreale, Giannotti, Pedreschi, Ruggieri, & Turini, 2019a; Darwiche, 2020; Darwiche & Hirth, 2020; Ignatiev, 2020; Marques-Silva, Gerspacher, Cooper, Ignatiev, & Narodytska, 2020)

Arguably the most explainable forms of ML models are decision trees[3], decision lists[4] and decision sets[5], since these encode simple logical rules. Of these, decision sets provide the simplest explanation, since if a rule "fires" for a given data instance, then this rule is the only explanation required. To explain decision lists, i.e. ordered sets of decision rules, we additionally need to take the order of rules into account.

In order to make explanations easy for humans to understand they should be as concise as possible.[6] There has been considerable investigation into producing the smallest possible optimal perfect decision trees (Narodytska et al., 2018; Verhaeghe et al., 2019), where the decision tree agrees perfectly with the training data, as well as producing the smallest possible sparse decision trees (Hu et al., 2019; Aglin et al., 2020), where there is a trade-off between size of the decision tree versus its accuracy on the training data.

Recent work has examined learning decision sets where the number of rules is minimized first and the number of literals is minimized afterwards (Ignatiev et al., 2018), as well as constructing a CNF classifier that minimizes the number of literals in the fixed number of rules (Malioutov & Meel, 2018; Ghosh & Meel, 2019) to explain the instances of the positive class. The latter two works also suffer from the limitation that only one class, class 1, is predicted by the rules, and class 0 is represented by the fact that no rule applies. Unfortunately, the explanation of negative instances is not succinct as explaining a class 0 example requires (the negation of) all rules.

We argue that previous work has not used the best measure of explainability, since, for instance, 2 rules each involving 80 conditions are significantly less explainable than 4 rules each involving only 10 conditions. Therefore, we investigate directly building decision sets that minimize the size redefined as the total number of literals used. This contributes to smaller decision sets (with respect to literals) which tend to be attractive to explain decisions. Note that the measure of size can also apply to decision lists.

There has been investigation of the method of optimizing decision lists (Angelino et al., 2017; Rudin & Ertekin, 2018; Angelino et al., 2018) that relies on a two-phase approach. First, decision rules are mined using some association rule mining techniques (Agrawal, Imieliński, & Swami, 1993), then an optimal order of the rules is found via search. In contrast, the method proposed in this paper directly generates all the rules of the optimal decision list as part of the search. This means it can generate decision rules which are meaningful in the context of their position in the target decision list, but could by themselves not provide valuable information on the training data, and therefore would not have been

3. (Hu, Rudin, & Seltzer, 2019; Aglin, Nijssen, & Schaus, 2020; Narodytska, Ignatiev, Pereira, & Marques-Silva, 2018; Verhaeghe, Nijssen, Pesant, Quimper, & Schaus, 2019; Apté & Weiss, 1997; Marques-Silva et al., 2017; Dufour, 2014; Bessiere, Hebrard, & O'Sullivan, 2009; Avellaneda, 2020; Janota & Morgado, 2020; Schidler & Szeider, 2021)
4. (Hancock, Jiang, Li, & Tromp, 1996; Rivest, 1987; Rudin & Ertekin, 2018; Angelino, Larus-Stone, Alabi, Seltzer, & Rudin, 2018; Angelino, Larus-Stone, Alabi, Seltzer, & Rudin, 2017; Wang, Rudin, Doshi-Velez, Liu, Klampfl, & MacNeille, 2017; Clark & Niblett, 1989; Yu, Ignatiev, Le Bodic, & Stuckey, 2020a)
5. (Kim, Khanna, & Koyejo, 2016; Doshi-Velez & Kim, 2017; Miller, 2019; Clark & Niblett, 1989; Clark & Boswell, 1991; Lakkaraju, Bach, & Leskovec, 2016; Ignatiev, Pereira, Narodytska, & Marques-Silva, 2018; Malioutov & Meel, 2018; Dash, Günlük, & Wei, 2018; Ghosh & Meel, 2019; Yu, Ignatiev, Stuckey, & Le Bodic, 2020b; Ignatiev, Lam, Stuckey, & Marques-Silva, 2021)
6. Interpretability is generally accepted to be a *subjective* concept, without a rigorous definition (Lipton, 2018). In this paper we measure interpretability in terms of the overall succinctness of the information provided by a model to explain a given prediction (see Section 6 for details).

mined by the approach of (Angelino et al., 2017; Rudin & Ertekin, 2018; Angelino et al., 2018). The two-step process can also generate many candidate rules to order when the number of features is large. Finally, this earlier approach does not optimize rules in terms of reducing the total number of literals, which may result in larger decision list models.

The previous methods focus on sparse decision lists, which trade size for accuracy on the training data. In contrast, we can also find optimal perfect decision lists, which agree completely with the training data.

Although the definition of size triggers harder computation in SAT models, the resulting machine learning models can be considerably smaller. However, for *sparse* decision sets and decision lists, the new measure is no more challenging for computation than the traditional rule count measure, and provides better granularity decisions on sparseness.

In summary, the contributions of this paper are:

- An approach to building optimal decision sets and decision lists in terms of the total number of literals required.
- The first method we are aware of to determine optimal decision lists, which agree with all the training data.
- The first SAT-based approach to find optimal sparse decision sets and decision lists that allow a trade-off between size and accuracy.
- We introduce the notion of *explanation size* which, for a particular example, determines how much information is required to explain the classification given by a machine learning model. We compare explanation size for decision lists and decision sets.
- Experiments demonstrating that our optimal sparse decision sets are as accurate as the best heuristic methods, but more concise.
- Experiments demonstrating that our approach to optimal sparse decision lists generates more accurate decision lists than previous methods

The paper is organized as follows. Section 2 presents the notation and definitions used throughout the paper. Section 3 outlines related work. The innovative SAT-based encodings for the inference of decision sets are described in Section 4. Section 5 illustrates encodings for computing decision lists. Section 6 introduces the notion of explanation size. The analysis of experimental results is discussed in Section 7. Finally, Section 8 concludes the paper.

2. Preliminaries

(Maximum) Satisfiability. The input of a Boolean satisfiability problem (SAT) (Biere, Heule, van Maaren, & Walsh, 2009) consists of a formula over a set of propositional variables using various logic operators on these variables. Solving a SAT problem consists in determining whether there exists an assignment of *true* or *false* value to each variable, called a *truth assignment*, such that the entire formula is *satisfied*. Otherwise, the formula is *unsatisfiable*. In the more specific *conjunctive normal form* (CNF), the formula is a conjunction

of clauses, and each clause is a disjunction of *literals*. A literal is a variable or its negation. Hence, a CNF formula can be satisfied if and only if at least one literal per clause can be set to *true*.

In the context of unsatisfiable formulas, the maximum satisfiability (MaxSAT) problem consists in finding a truth assignment that maximizes the number of satisfied clauses. In the Partial Weighted MaxSAT variant (Biere et al., 2009, Chapter 19), each clause c is either *soft* and has a weight w_c, or it is *hard*. An optimal solution then consists of a truth assignment that satisfies all hard clauses and maximizes the sum of the weights of the satisfied soft clauses.

Classification Problems. We consider a classification problem with a set of features $\mathcal{F} = \{f_1, \ldots, f_K\}$ and a label $\mathcal{C}$, all binary (or binarized using standard techniques (Pedregosa, Varoquaux, Gramfort, Michel, Thirion, Grisel, Blondel, Prettenhofer, Weiss, Dubourg, Vanderplas, Passos, Cournapeau, Brucher, Perrot, & Duchesnay, 2011)). The complete space of feature values (or *feature space* (Han, Kamber, & Pei, 2012)) is $\mathcal{U} \triangleq \prod_{r=1}^{K} \{f_r, \neg f_r\}$. The training set is denoted by $\mathcal{E} = \{e_1, \ldots, e_M\}$ and each instance $e_i \in \mathcal{E}$ is a pair (π_i, c_i) where $\pi_i \in \mathcal{U}$ and $c_i \in \mathcal{C}$. Furthermore, we assume without loss of generality in our context that dataset $\mathcal{E}$ *partially* defines a Boolean function $\phi : \mathcal{U} \to \mathcal{C}$, i.e. for any examples e_i and e_j in $\mathcal{E}$ associated with a same set of feature values, their classes are also the same.

The objective of classification in machine learning is to devise a function $\hat{\phi}$ that matches the actual function ϕ on the training data $\mathcal{E}$ and generalizes *suitably well* on unseen test data (Fürnkranz, Gamberger, & Lavrac, 2012; Han et al., 2012; Mitchell, 1997; Quinlan, 1993). In many settings (including *sparse*[7] decision sets and lists), function $\hat{\phi}$ is not required to match the actual function ϕ on the complete set of examples $\mathcal{E}$ and instead an *accuracy* measure is considered. Moreover, in classification problems one conventionally has to deal with an optimization problem, to optimize either with respect to the complexity of $\hat{\phi}$, or with respect to the accuracy of the learnt function (to make it match the actual function ϕ on a maximum number of examples), or both.

Rules, Decision Sets and Decision Lists. We can naturally represent a binary feature $f \in \mathcal{F}$ as a Boolean variable, and its two possible values as a literal or its negation, denoted f and $\neg f$. A *rule* has the form IF "instance satisfies formula" THEN "predict c", where the formula is a conjunction on a subset of the feature literals.

A *Decision Set* (DS) is an *unordered* set of rules. A decision set misclassifies an instance if no rule matches the instance, or if there exists a rule that predicts a wrong class.

A *Decision List* (DL) is an *ordered* set of rules. The first rule of a decision list that matches an instance is the one that classifies the instance. Decision lists are often written as a single cascade of IF-THEN-ELSEs, with the last rule often a "catch-all", or *default rule*, which matches all (remaining) instances to some class.

Example 1. Consider the following set of 8 items (shown as columns):

7. In contrast to *perfect* rule-based ML models, which aim at getting the highest possible training accuracy, *sparse* rule-based ML models trade off training accuracy for model size.

Item No.		1	2	3	4	5	6	7	8
Features	A	1	1	0	0	0	0	0	0
	B	0	1	1	1	0	0	0	0
	C	1	0	0	1	1	1	0	0
	D	0	1	1	0	0	1	1	1
	E	0	1	0	1	1	0	0	1
Class	H	1	1	0	0	1	1	0	0

A valid and optimal decision set for this data is

$$\begin{array}{llll} \mathsf{if} & A & \mathsf{then} & H \\ \mathsf{if} & \neg A \wedge \neg C & \mathsf{then} & \neg H \\ \mathsf{if} & \neg B \wedge C & \mathsf{then} & H \\ \mathsf{if} & \neg A \wedge B & \mathsf{then} & \neg H \end{array}$$

The *size* of this decision set is 11 (one for each literal on the left side and one for each rule, or alternatively, one for each literal on the left hand side and right hand side). Note how rules can overlap: both the first and third rule classify item 1.

A valid and optimal decision list for the data above is

$$\begin{array}{ll} & \mathsf{if}\ A\ \mathsf{then}\ H \\ \mathsf{else} & \mathsf{if}\ B\ \mathsf{then}\ \neg H \\ \mathsf{else} & \mathsf{if}\ C\ \mathsf{then}\ H \\ \mathsf{else} & \mathsf{if}\ \mathit{true}\ \mathsf{then}\ \neg H \end{array}$$

The size of the decision list is 7, and there is no overlap of rules by definition: item 1 is classified by the first rule only. Note that the last rule is a default rule.

Throughout the paper, each rule will be represented as a *path graph*, i.e. a graph reduced to a single path, where each literal is a *node* on the path. For example, the first two rules of the decision set would be represented as the two path graphs:

By construction, the *leaf* (i.e. last) node of each path corresponds to a class literal (see Example 2 for full example). Our SAT models will represent a Decision set (or list) as a concatenation of these path graphs into a single path graph. □

3. Related Work

The first work we are aware of for synthesizing a formula to cover a given partially defined (by $\mathcal{E}$ in our notation) Boolean function ϕ is by Michalski (1969).[8] Their general algorithm called AQ can be applied to the classification problem when $\mathcal{C} = \{c\}$ to construct a DNF formula by greedily choosing a minimal subset of features of each positive instance which does not cover any negative instance, until all positive instances are covered. The approach is also extended to multi-classification problems.

8. We thank the anonymous reviewers for pointing this out.

Decision sets and decision lists are rule-based predictive models that date back to the late 80s (Rivest, 1987; Clark & Niblett, 1989; Clark & Boswell, 1991). To the best of our knowledge, Kamath, Karmarkar, Ramakrishnan, and Resende (1992) proposed the first logic-based approach to constructing decision sets. Specifically, they introduced a SAT-based algorithm to synthesize a formula in disjunctive normal form (DNF) that matches a given set of training samples, which is tackled by an interior point method afterwards. Later, Lakkaraju et al. (2016) defined an approach to yielding a set of rules and minimizing a linear combination of criteria, e.g. the upper bound of the size in a rule, the number of rules, the number of overlapping rules, and misclassification.

The latest related approach to constructing decision sets is provided by Ignatiev et al. (2018). They develop a SAT model to iteratively compute minimal perfect decision sets where the training examples are perfectly classified and the number of rules is minimized. Then, the total number of literals required in a decision set is lexicographically minimized. However, as will be demonstrated later, the method computes larger decision sets than our approach that aims to minimize the total number of literals in a target decision set. Their method achieves better scalability to generate perfect decision sets as the more coarse-grained optimization measure is applied, i.e. the number of rules. The SAT method (Ignatiev et al., 2018) is also demonstrated to comprehensively outstrip the heuristic approach of Lakkaraju et al. (2016).

Malioutov and Meel (2018) provide a MaxSAT approach for the problem of binary classification, where they *fix the number of rules* and define the size in the model as the total number of literals across all clauses. Their proposed model minimizes a linear combination of Hamming loss and size, controlling the trade-off between explainability and accuracy instead of computing a perfect binary classifier. The scalability of this method is enhanced by Ghosh and Meel (2019), where rules are computed iteratively on partitions of the training data. However, only a single formula that classifies the positive items is computed rather than a *decision set* as defined in other work (Clark & Boswell, 1991; Lakkaraju et al., 2016; Ignatiev et al., 2018). Although the representation in the approach is smaller, explainability is reduced for negative examples where the whole formula is used to explain their classification.

Integer Programming (IP) has also been used to construct optimal rule-based models that only classify positive examples. Dash et al. (2018) introduce an IP approach for binary classification where an instance is classified as positive if and only if at least one rule of the model fires the example. The objective function of the model is to minimize a variation of Hamming loss, which is the number of misclassified positive instances, plus, for each misclassified negative example, the number of clauses misclassifying it. The bound on the size of each clause determines the complexity of this model. Column generation (Barnhart, Johnson, Nemhauser, Savelsbergh, & Vance, 1998) is used in the IP model as it has one binary variable for each possible clause. However, since the model is only solved heuristically for large datasets, the pricing problem tends to be too expensive.

One recent approach (Rudin & Ertekin, 2018) provides decision lists that have some optimality guarantee. Given a fixed set of decision rules, it chooses a minimum-size ordered subset of these rules; the order essentially terminates when a default rule is chosen. The authors model the problems as an IP and solve it with a MIP (Mixed IP) solver. The objective is a combination of training accuracy and sparsity, minimizing misclassifications where every rule used incurs a "cost" of C misclassifications, and every literal used costs

C_1 misclassifications. The method is slow, and somewhat restricted by the time required to generate all potential possible rules as input. They consider datasets with up to 3000 examples and 60 features, but cannot prove optimality of their solutions on the data tested. One advantage of the approach is that it is easy to customize, for example, favoring the use of certain features, or extending to cost-sensitive learning.

To the best of our knowledge, the first method to generate *optimal decision lists* extends the approach of Rudin and Ertekin (2018) using the same idea of ordering a fixed set of decision rules, but using a bespoke branch-and-bound algorithm (Angelino et al., 2018). The method makes use of bounding methods and symmetry elimination techniques. They minimize regularized misclassification, where each rule costs ρM misclassification errors where M is the number of training examples. The approach relies on the sparsification parameter ρ to limit the set of rules it needs to consider. It can find and prove optimal solutions to large problems (hundreds of thousands of examples), the main limitation is on the number of features, since the number of possible decision rules grows exponentially in the number of features. The datasets they consider have up to 28 (binary) features, and at most 189 decision rules are considered.

4. Decision Set Encoding

This section introduces two SAT-based approaches to learning decision sets of minimum *total* size, referring to the total number of literals used in a model. We recall that the number of training data is M, whereas the number of features is K. For binary classification problems we consider that there is one class pseudo-feature $\mathcal{C} = \{c\}$ and examples have this feature set to *true* or *false*. For three or more classes, we assume a one-hot encoding (Pedregosa et al., 2011), i.e. each class value is represented by a single binary feature, with each example having exactly one such feature from $\mathcal{C}$ set to *true*. We will describe the approaches to computing perfect decision sets that perfectly classify the training examples and sparse models that can trade off accuracy for size.

4.1 Iterative SAT Model for Perfect Decision Sets

We first introduce a SAT-based model which determines whether there exists a perfect decision set of a given size N. In order to find a decision set of minimum size, the SAT model is iteratively solved while increasing N by 1 until it is satisfied. All rules are encoded as a single sequence of feature literals, and a class literal ends each rule. The model also keeps track of which examples are valid (agree) with previous literals in the rule, and checks whether the examples that reach the end of a rule (i.e. a leaf node) match the correct class. The Boolean variables used in the encoding are described below:

- s_{jr}: node j is a literal on feature $f_r \in \mathcal{F} \cup \mathcal{C}$;
- t_j: truth value of the literal for node j;
- v_{ij}: example $e_i \in \mathcal{E}$ is valid at node j;

The model is as follows:

- Only one feature (or a class feature) is used in a node:

$$\forall_{j\in[N]} \sum_{r\in[K+|\mathcal{C}|]} s_{jr} = 1 \tag{1}$$

- The last node is a leaf:

$$\bigvee_{c\in\mathcal{C}} s_{Nc} \tag{2}$$

- All examples are valid at the first node:

$$\forall_{i\in[M]}\ v_{i1} \tag{3}$$

- An example e_i is valid at node $j+1$ iff j is a leaf node, or e_i is valid at node j and e_i and node j agrees on the value of the feature s_{jr} selected for that node:

$$\forall_{i\in[M]}\forall_{j\in[N-1]}\ v_{ij+1} \leftrightarrow (\bigvee_{c\in\mathcal{C}} s_{jc}) \vee (v_{ij} \wedge \bigvee_{r\in[K]} (s_{jr} \wedge (t_j = \pi_i[r]))) \tag{4}$$

- If example e_i is valid at a leaf node j, it should agree on the class feature:

$$\begin{aligned} \forall_{i\in[M]}\forall_{j\in[N]}\ (s_{jc} \wedge v_{ij}) \rightarrow (t_j = c_i) \quad & \mathcal{C} = \{c\} \\ \forall_{i\in[M]}\forall_{j\in[N]}\forall_{c\in\mathcal{C}}\ (s_{jc} \wedge v_{ij}) \rightarrow c_i \quad & |\mathcal{C}| \geq 3 \end{aligned} \tag{5}$$

- When there are 3 or more classes we restrict leaf nodes to only consider *true* examples of the class:

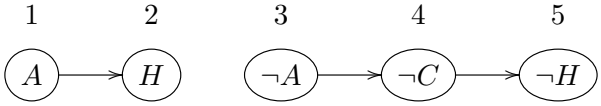
$$\forall_{j\in[N]}\forall_{c\in\mathcal{C}}\ s_{jc} \rightarrow t_j \qquad |\mathcal{C}| \geq 3 \tag{6}$$

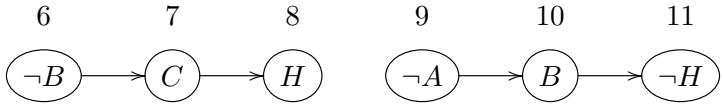
- For every example there should be at least one leaf node where it is valid:

$$\forall_{i\in[M]} \bigvee_{j\in[N]} ((\bigvee_{c\in\mathcal{C}} s_{jc}) \wedge v_{ij}) \tag{7}$$

The model described above represents a non-clausal Boolean formula, which can be clausified with the utilization of auxiliary variables (Tseitin, 1968). Also note that any of the known cardinality encodings can be used to denote the sum in constraint (1) (Biere et al., 2009, Chapter 2) (also see (Asín, Nieuwenhuis, Oliveras, & Rodríguez-Carbonell, 2009; Bailleux & Boufkhad, 2003; Batcher, 1968; Sinz, 2005)). Finally, the size (in terms of the number of literals) of the suggested SAT encoding is $\mathcal{O}(N \times M \times K)$, which results from constraint (4).

Example 2. Consider a solution for 11 nodes for the dataset of Example 1. The sequence of nodes that represent the rules are shown below:





The interesting (true) decisions for each node are given in the following table

Node j	1	2	3	4	5	6	7	8	9	10	11
s_{jr}	s_{1A}	s_{2H}	s_{3A}	s_{4C}	s_{5H}	s_{6B}	s_{7C}	s_{8H}	s_{9A}	s_{10B}	s_{11H}
t_j	1	1	0	0	0	0	1	1	0	1	0
v_{ij}	v_{11}	v_{12}	v_{13}	v_{24}	v_{75}	v_{16}	v_{17}	v_{18}	v_{19}	v_{210}	v_{311}
	$\vdots$	v_{22}	$\vdots$	$\vdots$	v_{85}	$\vdots$	v_{57}	v_{58}	$\vdots$	$\vdots$	v_{411}
	v_{81}		v_{83}	v_{84}		v_{86}	$\vdots$	v_{68}	v_{89}	v_{810}	
							v_{87}				v_{411}

The selected variable at the end of each rule is class H. Note that all examples are valid at the start node in each rule, and each feature literal leads to the reduction of the valid set for the next node. In each leaf node j, the valid instances of the correct class are determined by the truth value t_j of that node. □

The iterative SAT model tries to compute a perfect decision set of size N. If this fails, we increase the predefined size by 1 and resolve, until a solution is found or resource limits (typically computation time) are reached. A potential question is why binary search is not used instead. The issue is that the complexity of deciding satisfiability of the described model grows (potentially) exponentially with N, and therefore, predefining a large N can lead to the failure of solving the whole problem.

Example 3. Consider the data shown in Example 1. The iterative SAT model initially tries to find a decision set of size 1, which fails, then of size 2, etc. until the model reaches size 11 where it can determine the perfect decision set: $A \Rightarrow H$, $\neg A \wedge \neg C \Rightarrow \neg H$, $\neg B \wedge C \Rightarrow H$, $\neg A \wedge B \Rightarrow \neg H$ of size 11 by finding the model shown in Example 2. □

4.2 MaxSAT Model for Perfect Decision Sets

Instead of using the proposed iterative SAT-based model, which iterates over varying size N of the perfect decision set, we can modify the model to determine a perfect decision set of size at most N, and formulate a MaxSAT problem such that the number of nodes used can be minimized. Let us add a flag variable u_j for every available node. Variable u_j is *true* whenever the node j is unused and *false* otherwise. The model is as follows:

- A node either uses a feature or is unused:

$$\forall_{j\in[N]}\ u_j + \sum_{r\in[K+|\mathcal{C}|]} s_{jr} = 1 \tag{8}$$

- If a node j is unused then so are all the following nodes

$$\forall_{j\in[N-1]}\ u_j \rightarrow u_{j+1} \tag{9}$$

- The last used node is a leaf:

$$\forall_{j\in[N-1]}\ u_{j+1} \rightarrow u_j \vee \bigvee_{c\in\mathcal{C}} s_{jc} \tag{10}$$

$$u_N \vee \bigvee_{c\in\mathcal{C}} s_{Nc} \tag{11}$$

The constraints above together with constraints (3), (4), (5), (6), and (7) make up the hard clauses of the MaxSAT formula, i.e. every clause of them must be satisfied. The model maximizes $\sum_{j\in[N]} u_j$, which is the representation of the list of unit soft clauses of the form $(u_j, 1)$.

In the worst case, the model is still run iteratively. We first propose an upper bound N on the size of the decision set. The model tries to search for a decision set of size no more than N. If this fails, we need to increase N by some number (say 10) and retry, until the solution is found or resource limits are reached.

Example 4. Revisiting the solution illustrated in Example 1 when N is set to 13 we find the solution shown in Example 2 extended in which the last two nodes are unused: $u_{12} = u_{13} = \mathit{true}$. The last used node 11 is clearly a leaf. Note that truth value (t_j) and validity (v_{ij}) variables are irrelevant to unused nodes j. □

4.3 MaxSAT Model for Sparse Decision Sets

We can extend the MaxSAT model to look for *sparse* decision sets, which trade off training accuracy for model size, rather than *perfect* decision sets, which aim at the highest possible training accuracy. The objective to minimize is the total number of misclassifications (including non-classifications where no prediction information is provided for the example by any decision rule) plus the product of the number of nodes in a decision set and a discount factor Λ, which records that Λ fewer misclassifications are worth the addition of one node to the decision set. Typically we consider $\Lambda = \lceil \lambda M \rceil$ where λ is the regularized penalty of nodes in terms of misclassifications.

We introduce variable m_i to represent that example $i \in [M]$ is misclassified. Consider the following constraints:

- If example e_i is valid at a leaf node j then e_i agrees on the class feature or e_i is misclassified:

$$\begin{aligned} &\forall_{i\in[M]}\forall_{j\in[N]}\ (s_{jc} \wedge v_{ij}) \rightarrow (t_j = c_i \vee m_i) \quad && \mathcal{C} = \{c\} \\ &\forall_{i\in[M]}\forall_{j\in[N]}\forall_{c\in\mathcal{C}}\ (s_{jc} \wedge v_{ij}) \rightarrow (c_i \vee m_i) \quad && |\mathcal{C}| \geq 3 \end{aligned} \tag{12}$$

- For every example there should be at least one leaf literal where it is valid or the example is misclassified (actually *non-classified*):

$$\forall_{i\in[M]}\ m_i \vee \bigvee_{j\in[N]} (\bigvee_{c\in\mathcal{C}} s_{jc} \wedge v_{ij}) \tag{13}$$

together with all the MaxSAT constraints from Section 4.2 except constraints (5) and (7). The objective function of the proposed model is

$$\sum_{i\in[M]} m_i + \sum_{j\in[N]} \Lambda(1-u_j) + N\Lambda$$

represented as soft clauses $(\neg m_i, 1)$, $i \in [M]$, and (u_j, Λ), $j \in [N]$.

Note that the value of regularized penalty λ is critical. As the value of λ becomes higher, the emphasis of the problem shifts more to "sparsification" of the target decision set, rather than its accuracy. In other words, higher values of λ (and hence of Λ as well) contribute to simpler decision sets where its accuracy on training examples is sacrificed.

4.4 Separated Models for Decision Sets

A convenient feature of optimal decision sets is the following: the union of minimal decision sets for each $c \in \mathcal{C}$ that correctly classifies all examples of class c and does not misclassify any examples not of class c as class c, is a minimal target decision set for the whole problem.

That means we can compute perfect decision sets for $|\mathcal{C}|$ classes by separately computing $|\mathcal{C}|$ perfect decision sets, one for each class. The union of these $|\mathcal{C}|$ models, which we call *"separated model"*, is clearly not much smaller than the complete model, as each separated model still includes constraint (4) for each example leading to the size $\mathcal{O}(N \times M \times K)$. The advantage arises because the minimal size required for each separated model is smaller.

Example 5. Consider the dataset shown in Example 1. We can iteratively compute decision rules for the positive examples: $A \Rightarrow H$, $\neg B \wedge C \Rightarrow H$ of size 5, and the negative examples: $\neg A \wedge \neg C \Rightarrow \neg H$, $\neg A \wedge B \Rightarrow \neg H$ of size 6. This is faster than solving the problems together where the complete model iterates from size 1 to size 11 to ultimately find the same solution. □

For sparse decision sets, the definition of *misclassifications* needs to be modified in order that a separated solution is available. Suppose that an example $e_i \in \mathcal{E}$ is of class $c_i \in \mathcal{C}$ then the *number of misclassifications* of example e_i is counted as follows:

- If example e_i is not classified as class c_i, then this counts as one misclassification.
- If example e_i is classified as class $c_j \in \mathcal{C}$, $c_j \neq c_i$ that counts as one misclassification per class.

With the modified definition of misclassification, we can construct minimal decision sets per class separately and then join them together.

5. Decision List Encoding

We can modify the SAT encoding for decision sets described in Section 4 fairly naturally to instead define decision lists.

5.1 MaxSAT Model for Perfect Decision Lists

The MaxSAT model introduced in Section 4.2 determines whether there exists a perfect decision set of at most a given size N. We can modify this to determine a perfect *decision*

list of size at most N by keeping track of which items have previously been classified by a previous rule, and preventing them from being considered (in)valid in later rules. The sequence of literals is viewed as a path graph, with one feature literal per node.

We introduce a new variable n_{ij} to represent that example $e_i \in \mathcal{E}$ is not previously classified by any nodes before j.

The model is as follows:

- All examples are not previously classified at the first node:

$$\forall_{i\in[M]}\ n_{i1} \tag{14}$$

- An example e_i is previously unclassified at node $j+1$ iff it was previously unclassified, and either j is not a leaf node or it was invalid at the previous leaf node (so not classified by the rule that finished there):

$$\forall_{i\in[M]}\forall_{j\in[N-1]}\ n_{ij+1} \leftrightarrow n_{ij} \wedge ((\bigwedge_{c\in\mathcal{C}} \neg s_{jc}) \vee \neg v_{ij}) \tag{15}$$

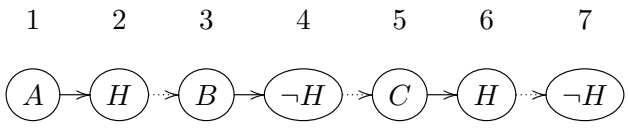
- An example e_i is valid at node $j+1$ iff j is a leaf node and it was previously unclassified, or e_i is valid at node j and e_i and node j agree on the value of the feature s_{jr} selected for that node:

$$\forall_{i\in[M]}\forall_{j\in[N-1]} \quad \begin{array}{l} v_{ij+1} \leftrightarrow (((\bigvee_{c\in\mathcal{C}} s_{jc}) \wedge n_{ij+1}) \vee \\ (v_{ij} \wedge \bigvee_{r\in[K]} (s_{jr} \wedge (t_j = \pi_i[r])))) \end{array} \tag{16}$$

The constraints above together with constraints (3), and (5)–(10) make up the hard constraints of the MaxSAT model. The model maximizes $\sum_{j\in[N]} u_j$, which can be trivially represented as a list of unit soft clauses of the form $(u_j, 1)$. The size (in terms of the number of literals) of the proposed SAT encoding is $\mathcal{O}(N \times M \times K)$, which results from constraints (15) and (16).

The differences between the above model and the MaxSAT model of decision sets is the addition of the n_{ij} variables to track which items have been previously classified, and their use in constraint (16), as well as the rules to compute them given in constraints (14) and (15).

Example 6. Consider a solution for 7 nodes for the data of Example 1. The representation of the decision list is shown below:



The interesting (true) decisions for each node are as follows:

Node j	1	2	3	4	5	6	7
s_{jr}	s_{1A}	s_{2H}	s_{3B}	s_{4H}	s_{5V}	s_{6H}	s_{7H}
t_j	1	1	1	0	1	1	0
v_{ij}	v_{11}	v_{12}	v_{33}	v_{34}	v_{55}	v_{56}	v_{77}
	$\vdots$	v_{22}	$\vdots$	v_{44}	$\vdots$	v_{66}	v_{87}
	v_{81}		v_{83}		v_{85}		
n_{ij}	n_{11}	n_{12}	n_{33}	n_{34}	n_{55}	n_{56}	n_{77}
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	n_{87}
	n_{81}	n_{82}	n_{83}	n_{84}	n_{85}	n_{86}	

The selected variable at the end of each rule is class H. Note that at the start and after each leaf node all previously unclassified examples are valid, and each feature literal triggers the decrease of the valid set for the next node. The valid instances of the correct class are determined by the truth value t_j in each leaf node j. □

The MaxSAT model tries to find a decision list of size at most N. If this fails, we can increase N by some amount and resolve, until either resource limits (typically computation time) are reached or a solution is found.

5.2 MaxSAT Model for Sparse Decision Lists

We modify the MaxSAT model for sparse decision sets introduced in Section 4.3 to look for sparse decision lists that trade off accuracy for size. The same objective function and discount factor Λ are used in the model where Λ fewer misclassifications are worth the addition of one node to the decision list.

The hard part of the model consists of the constraints (12) and (13) with all the hard constraints of the model for perfect decision lists except constraints (5) and (7). The objective function is

$$\sum_{i\in[M]} m_i + \sum_{j\in[N]} \Lambda(1-u_j) + N\Lambda$$

represented as soft clauses $(\neg m_i, 1)$, $i \in [M]$, and (u_j, Λ), $j \in [N]$.

5.3 Separated Models for Decision Lists

Separated models for decision sets have been introduced in Section 4.4 where the size of the union of $|\mathcal{C}|$ models is the same as the size of the complete model.

For decision lists this property no longer holds. If we compile decision lists separately for each class, we must still order the decision lists of different classes. Therefore, no optimal decision list might be expressed as rules for one class, followed by another class.

Example 7. Consider the dataset of Example 1. Recall that an optimal decision list shown in Example 1 has 7 literals.

An optimal decision list that is separated in class order is

$$\begin{array}{ll} & \text{if } A \text{ then } H \\ \text{else} & \text{if } \neg B \wedge C \text{ then } H \\ \text{else} & \text{if } B \text{ then } \neg H \\ \text{else} & \text{if } \mathit{true} \text{ then } \neg H \end{array}$$

requiring one more literal.

Given that separated models are important for scaling this approach to larger problems, we need to consider approaches for defining decision lists in a separated form. We consider a number of different approaches:

fixed σ Given a permutation σ of classes, find an optimal decision list for the first class in σ, then make an optimal decision list for the second class ignoring items already classified by the decision list for the first class. Then consider the third class, etc.

greedy Make an optimal decision list for each class independently: choose the one that is best under some metric. Fix its solution as the first part of the decision list. Calculate I' as the items not classified by this decision list. Make an optimal decision list for each remaining class independently. Again, choose the best one and fix it. Continue until all classes are considered, or I' becomes empty.

For the fixed permutation case, one can try all possible permutations, if there are not too many, e.g. $|\mathcal{C}| \leq 3$, or use a heuristic to choose a permutation σ. One heuristic we consider is sorting the classes by increasing/decreasing number of their respective items in the training set. Alternatively, we consider ordering the classes greedily based on the post-hoc analysis of the accuracy or cost of individual class representations obtained on the training data. Here, training accuracy for the representation of class c is $1 - \frac{\sum_{i \in [M], c_i = c} m_i}{|\{e_i \in \mathcal{E}, c_i = c\}|}$ while the cost of representation of class c is assumed to be $N - \sum_{j \in [N]} u_j + \left\lceil \frac{\sum_{i \in [M]} m_i}{\Lambda} \right\rceil$.

Note that for separated sparse models, the objective is effectively different. Using the same objective for each class separately means that we count a misclassification once for every class it is detected by. This is arguably more informative. As we cannot guarantee the same optimal solutions anyway (due to order restrictions), this seems acceptable.

6. Explanation Size

Given two different ML models, we can ask *which model gives the smallest explanation* on a particular data instance. By optimizing the size of a decision list or decision set, we believe the size of the explanations it creates will be small, but this is not completely accurate. The explanation size of an ML model can be far smaller than the whole model. The implicit notion of *explanation size* we are trying to capture is, if a customer/user were to ask why our model made a decision for their case, how would we explain that decision? Note that we also define explanation size for the cases where a decision set makes no decision, either since no rule fires, or two contradictory rules fire. We define the explanation size of a model $\hat{\phi}$ on an example instance e as follows (note that alternative definitions of explanation size may exist, both for decision sets and decision lists).

If $\hat{\phi}$ is a decision set and the rules in $\hat{\phi}$ that fire on example e are $\{\textbf{if } \pi_i \textbf{ then } c_i\}$, $\forall i \in [n]$, $n \leq N = |\hat{\phi}|$, then

- if all the classes $c_1, \ldots, c_n$ agree, i.e. $c_i = c'$, $\forall i \in [n]$, $c' \in \mathcal{C}$, then the explanation size for example e is $\frac{\sum_{i=1}^{n} |\textbf{if } \pi_i \textbf{ then } c_i|}{n}$, that is, the average of the rules, any of which could explain the example.
- if not all classes agree for e then the explanation size is the sum of averages of the rules for all the conflicting classes predicted for e; wlog. assume that $c_i = c'$, $c' \in \mathcal{C}$, $1 \leq i \leq k < n$ and $c_j = c''$, $c'' \in \mathcal{C}$, $k+1 \leq j \leq n$, then the explanation size for example e is $\frac{\sum_{i=1}^{k} |\textbf{if } \pi_i \textbf{ then } c_i|}{k} + \frac{\sum_{j=k+1}^{n} |\textbf{if } \pi_j \textbf{ then } c_j|}{n-k}$; similar reasoning can be applied to situations of more than two conflicting classes.
- if no rule fires then the explanation size is $|\hat{\phi}|$, i.e. we need the whole decision set to explain why e is not classified.

If $\hat{\phi}$ is a decision list and $\textbf{if } \pi_j \textbf{ then } c_j$ is the first rule in $\hat{\phi}$ that fires on example e then

- the explanation size is $\sum_{i=1}^{j} |\pi_i| + 1$ as we need to explain why none of the previous rules fired, and why rule j did.
- if no rule fires for e then the explanation size is $|\hat{\phi}|$, i.e. we need the whole model to explain why e is not classified. Note that in practice this does not occur since the last rule will be a default rule, and all examples will be classified.

Note that it is easy to extend the notion of explanation size to decision trees (as the path from root to leaf) though decision tree models are not considered in this paper.

7. Experimental Results

This section describes the results of experimental assessment of the proposed approach to computing optimal decision sets and decision lists. Firstly, we discuss the comparison between the decision set approaches and state-of-the-art decision sets in terms of performance, accuracy and model size. Afterwards, we compare the proposed approach to decision lists with the SAT-based decision sets as well as the only previous approach to optimal sparse decision lists we are aware of (Angelino et al., 2017; Wang et al., 2017).

Experimental results are obtained on the StarExec cluster[9] (Stump, Sutcliffe, & Tinelli, 2014), each computing node of which uses an Intel Xeon E5-2609 2.40GHz CPU with 128GByte of RAM, running CentOS 7.7. The time limit and memory limit used per process are 1800 seconds and 16 GB.

For the evaluation, we use the 71 datasets from the UCI Machine Learning Repository (Dua & Graff, 2017) and Penn Machine learning Benchmarks (Olson, La Cava, Orzechowski, Urbanowicz, & Moore, 2017). We also use 5-fold cross validation, which results in 355 pairs of training and test data split with the ratio 80% and 20%, respectively. Finally, feature domains are quantized into 2, 3, and 4 intervals and then *one-hot encoded* (Pedregosa et al., 2011). The number of one-hot encoded features (training instances, resp.)

9. `https://www.starexec.org/`

per dataset in the benchmark suite varies from 3 to 384 (from 14 to 67557, resp.). The total number of benchmark datasets is 1065 ($71 \times 5 \times 3$).

All the proposed models are implemented as collections of Python scripts[10] and solving is done by instrumenting incremental calls to SAT solver Glucose 3 (Audemard, Lagniez, & Simon, 2013) and MaxSAT solver RC2-B (Ignatiev, Morgado, & Marques-Silva, 2018, 2019a).

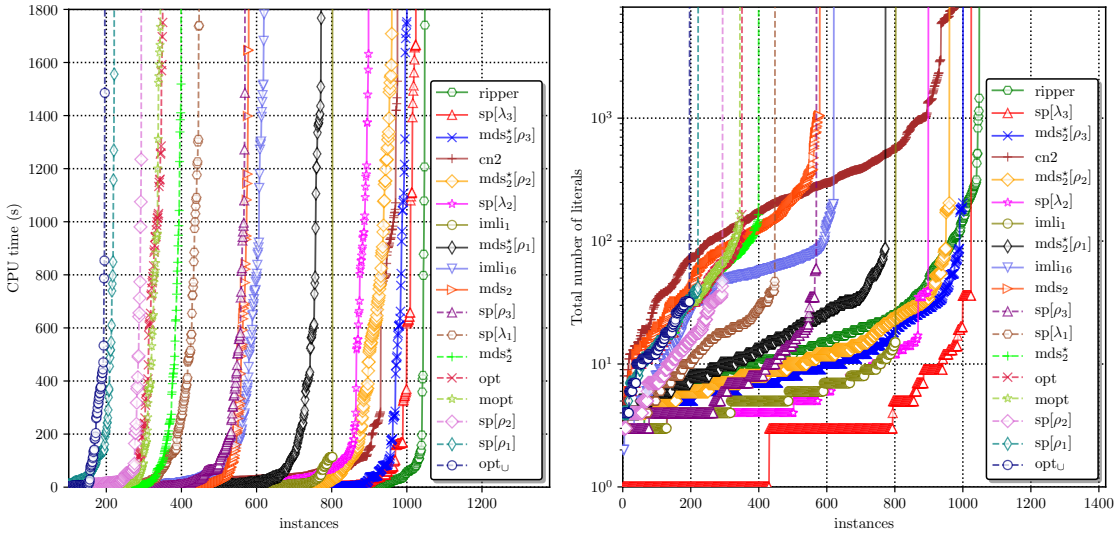
7.1 Experimental Results for Decision Sets

Implementation. All models in the implementation target independent computation of each class[11], as described in Section 4.4. The iterative SAT and MaxSAT models targeting perfect decision sets are referred to as *opt* and *mopt*. To explain the benefits of separated models over the models aggregating all classes, a complete SAT model is also included, which is referred to as $opt_{\cup}$. Lastly, some variants of the MaxSAT model for sparse decision sets named $sp[\lambda_i]$ are assessed with three different values of regularized penalty: $\lambda_1 = 0.005$, $\lambda_2 = 0.05$, and $\lambda_3 = 0.5$.

Competitors. MinDS$_2$ and MinDS$_2^\star$ (Ignatiev et al., 2018) for perfect decision sets referred to as mds_2 and $mds_2^\star$ are considered as the competitors in the implementation. The former algorithm minimized the number of rules, whereas the latter does lexicographic optimization, i.e. it minimizes the number of rules first and the total number of literals afterwards. In addition, we modified MinDS to be able to compute *sparse* decision sets similar to what is described in Section 4.3. The implementation is called $mds_2[\rho_i]$, tested with three values of regularized penalty $\rho_1 = 0.05$, $\rho_2 = 0.1$, and $\rho_3 = 0.5$. Note that $\rho_i \neq \lambda_i$, $i \in [3]$, as the two models use different measures where MinDS targets rules but the other one targets literals. To consider fair comparison of performance between the new model *sp* and of the sparse variant of $mds_2^\star$, we provide the configuration of $sp[\rho_i]$ with $\rho_i = \frac{\lambda_i}{K}$, where K refers to the number of features in the dataset. In other words, a rule is considered equivalent to K literals.

Apart from MinDS, some state-of-the-art algorithms were also considered, including a MaxSAT-based method IMLI (Ghosh & Meel, 2019) (i.e. the direct successor of MLIC (Malioutov & Meel, 2018)) as well as two heuristic approaches CN2 (Clark & Niblett, 1989; Clark & Boswell, 1991), and RIPPER (Cohen, 1995). Note that IMLI and RIPPER compute only one class given the training data[12]. Both of these competitors incorporate a *default rule* which classifies a class (1) different from the constructed one and (2) represented by the majority of the training examples. Finally, IMLI aims to compute a target decision set given the number of rules k, which varies from 1 to 16. The best results (both regarding accuracy and performance) are demonstrated by the configuration aiming at the smallest possible number of rules, i.e. $k = 1$, while the worst results were shown for $k = 16$. Therefore, only the extreme values of k are used to compare the results, represented by $imli_1$ and $imli_{16}$.

10. `https://github.com/alexeyignatiev/minds/`
`https://github.com/jinqiang-yu/dlsat/`
11. The prototype adapts all the developed models to the case of multiple classes, which is motivated by the practical importance of non-binary classification.
12. The implementation of RIPPER considered in our experimental evaluation is taken from `https://github.com/imoscovitz/wittgenstein`. A reader may find other implementations having different capabilities.



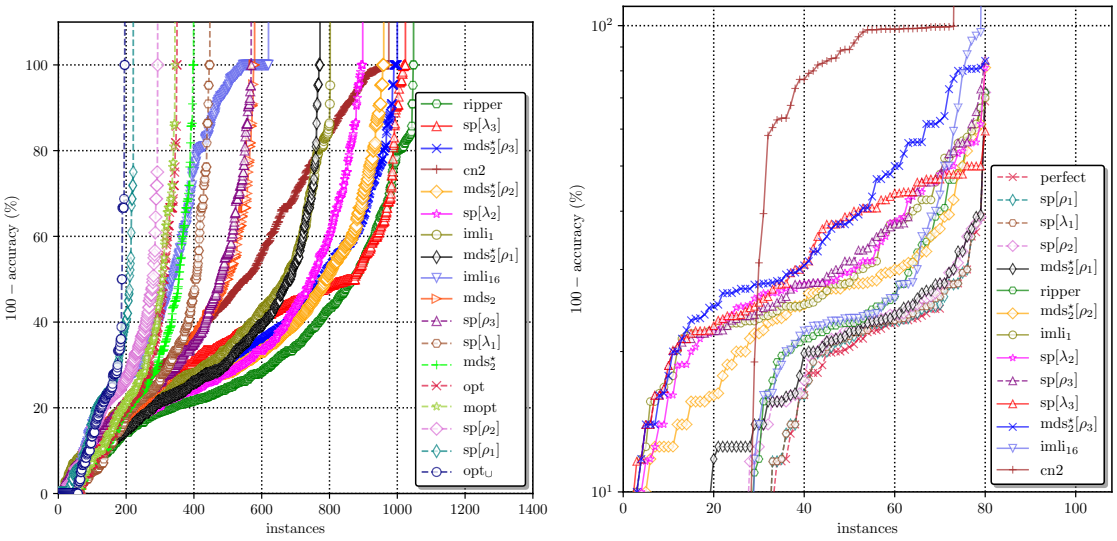
(a) Performance (b) Decision set size

Figure 1: Performance of all decision set competitors and the quality of solutions in terms of decision set size.

Performance. The performance of these models is shown in Figure 1a using a cactus plot. The plot shows for each method how many instances were solved by the method within each CPU time. As can be observed, the best results are demonstrated by *ripper*, which is able to compute 1048 selected datasets within the 1800s time limit. The second and third positions are occupied by the proposed MaxSAT models for sparse decision sets $sp[\lambda_3]$ and $mds^{\star}_2[\rho_3]$, which can successfully cope with 1024 and 1000 datasets respectively. *cn2* takes the fourth place with 975 datasets solved. $imli_1$ is the best configuration of *imli*, finishing with 802 instances solved, whereas the worst configuration $imli_{16}$ computes only 620 datasets. Finally, the worst performance is shown by the approaches targeting *perfectly accurate* decision sets *opt*, *mopt*, and $opt_{\cup}$ as well as the MaxSAT approaches for sparse decision sets with low regularized penalty $sp[\rho_1]$ and $sp[\rho_2]$. For example, $opt_{\cup}$ solves only 196 datasets.

Test Accuracy. The calculation of accuracy is as follows: (1) if a rule of a wrong class *"covers"* an instance, the instance is considered misclassified (though there are other rules of the correct class covering this instance); (2) if an instance is not covered by any rule of a correct class, the instance is considered misclassified. Afterwards, the accuracy is calculated as $\frac{M-E}{M} \times 100\%$, where M is the total number of instances and E is the total number of misclassified instances.

Confirmed by Figure 2b which depicts the accuracy among all the approaches for the datasets solved by *all* these approaches, perfect decision sets tend to have highest accuracy once they are computed. As can be observed, the virtual *perfect* approach, which represents all the tools aiming at perfect decision sets, i.e. *opt*, *mopt*, $opt_{\cup}$, mds_2, and $mds^{\star}_2$, outperforms the other approaches regarding testing accuracy. The average testing accuracy



(a) On the complete set of datasets

(b) Instances solved by all tools

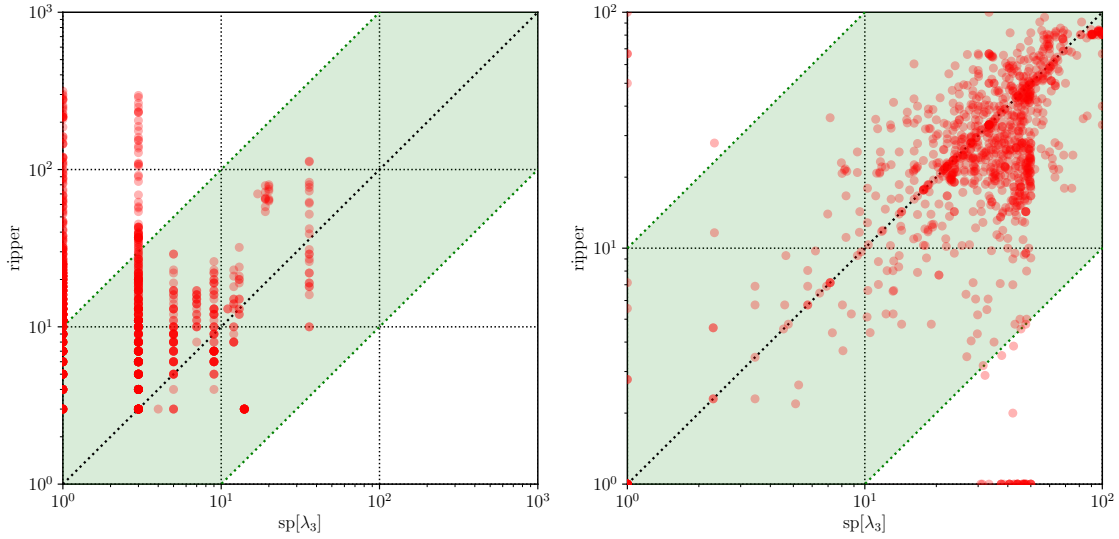
Figure 2: Accuracy of the considered decision set approaches.

on these solved datasets is 85.89%[13]. Conversely, *cn2* (43.73%) is the worst in terms of testing accuracy. Also, the average accuracy of $mds_2^\star[\rho_3]$, $sp[\lambda_3]$, $imli_1$, $imli_{16}$, and *ripper* is 61.71%, 67.42%, 76.06%, 77.42%, and 80.50% respectively.

The result changes significantly if we compare testing accuracy on the entire set of benchmark datasets, demonstrated in Figure 2a. Here, we *assume* that the accuracy for a dataset is 0% if a tool fails to train a model for the dataset within the given time limit. The figure shows that *ripper* (68.13% on average) achieves the best accuracy, followed by the sparse decision sets computed by $mds_2^\star[\rho_3]$ and $sp[\lambda_3]$ (61.23% and 60.91%, respectively). The average accuracy of $imli_1$ and $imli_{16}$ is 50.66% and 28.26%, whereas the average accuracy achieved by *cn2* is 47.49%.

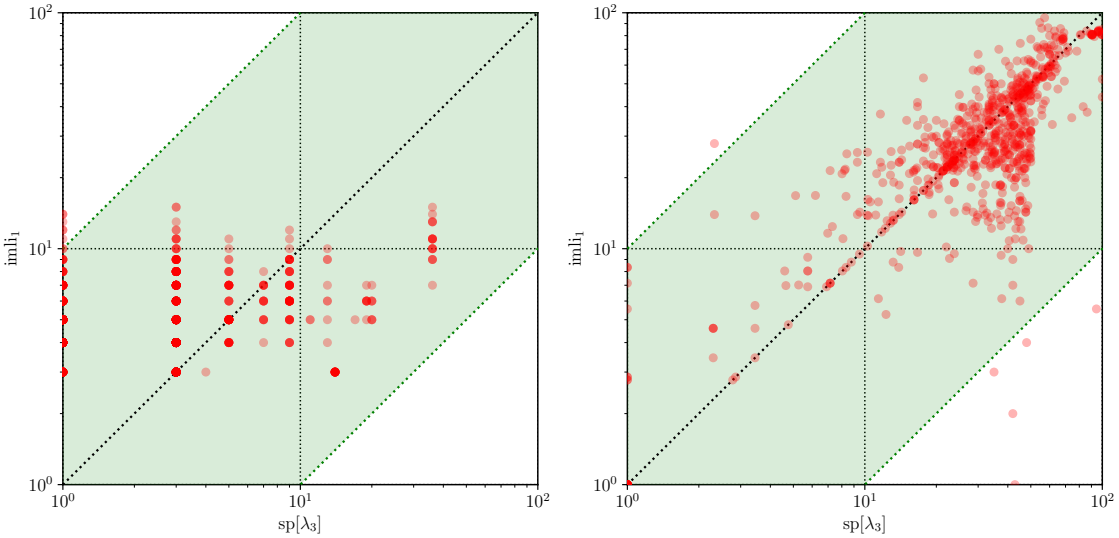
Decision Set Size Figure 1b illustrates the size in terms of the number of literals in a decision set model. The figure demonstrates that the winner is $sp[\lambda_3]$. The decision sets of $sp[\lambda_3]$ make up only one literal[14] for more than 400 datasets. For another bunch of around 400 instances, the solutions of $sp[\lambda_3]$ consist of 3 literals. Computing small solutions is a noticeable achievement as $sp[\lambda_3]$ also achieves overall high accuracy. The average size of decision sets computed by $sp[\lambda_3]$ is 4.18. Note that $imli_1$ occupies the second place with 5.57 literals per dataset though its solution always consists of only one rule. In comparison to this, the average solution size of *cn2* is 598.05. Finally, the result of $imli_{16}$ is 46.29, while the average solution of *ripper* has 35.14 literals.

13. This average value is the highest possible accuracy that can be achieved on these datasets whatever machine learning model is considered.
14. In a unit-size decision set, the literal is meant to assign a constant class. This can be seen as applying a *default rule*.



(a) Decision set size (b) Test accuracy

Figure 3: Comparison of $sp[\lambda_3]$ and *ripper*.

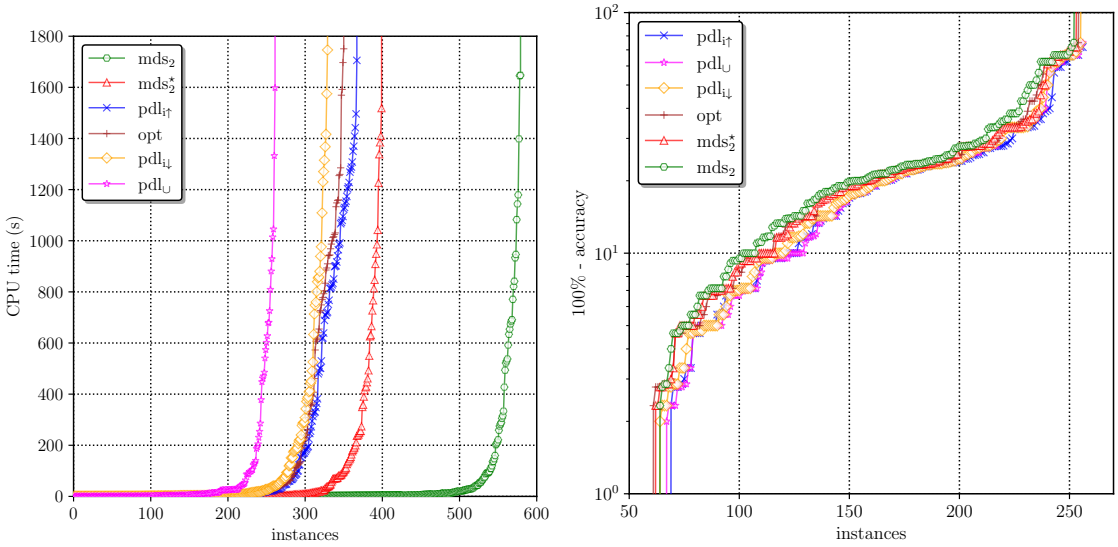


(a) Decision set size (b) Test accuracy

Figure 4: Comparison of $sp[\lambda_3]$ and $imli_1$.

As expected, perfect decision sets, i.e. those constructed by *opt*, *mopt*, $opt_{\cup}$, as well as mds_2, and $mds_2^{\star}$, tend to be larger in general. It should also be noted that the $mds_2^{\star}$ obtains perfect decision sets of size 40.70, whereas the sparse version gets 14.52 literals per solution.

A few more details. The comparison of $sp[\lambda_3]$ with *ripper* and $imli_1$ is depicted in Figure 3 and Figure 4 using scatter plots that plot the compared statistic of each method on the two axes. Note that the axes are log scaled since the differences in the methods



(a) Performance (b) Test accuracy

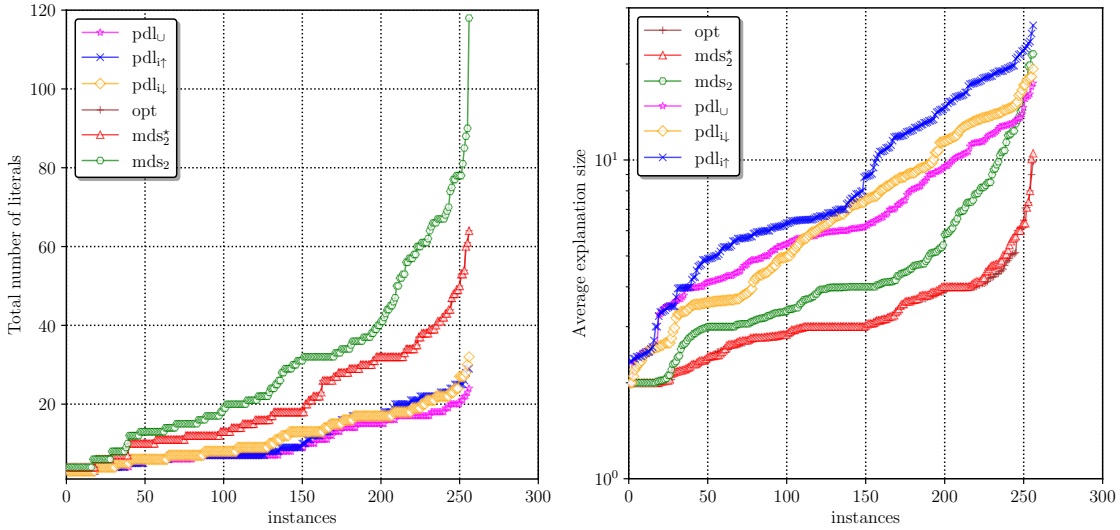
Figure 5: Performance and accuracy of perfect decision list and decision set models.

can be significant. All the results in these figures are obtained for the datasets solvable by each pair of contestants. As can be observed in Figure 4a and Figure 4b, $imli_1$ and $sp[\lambda_3]$ are comparable with respect to size and accuracy. Nevertheless, as $imli_1$ constructs decision sets representing only one class and it considerably underperforms $sp[\lambda_3]$, the latter method is considered a better alternative. Even if the accuracy of *ripper* is comparable with the accuracy of $sp[\lambda_3]$ (see Figure 3b) and the former approach achieves the best overall performance, the size of decision sets computed by *ripper* tends to be significantly larger than the one of its opponent (see Figure 3a).

Finally, a drawback of both RIPPER and IMLI is that they construct a representation for one class only. Therefore, a concise explanation for the instances of non-computed classes cannot be provided by the two approaches. This is in clear contrast with our approaches, which offer users a concise representation of every class in the dataset.

7.2 Experimental Results for Decision Lists

Implementation The complete MaxSAT model is referred to as $pdl_{\cup}$. As was shown in Section 5.3, separated models do not guarantee optimality of the size of decision lists, and so we tested various ordering of the classes when computing separated decision lists. Concretely, $pdl_{i\uparrow}$ and $pdl_{i\downarrow}$ refer to the separated models that order classes by the increasing/decreasing number of training data in the classes. Sparse models are referred to as $sdl[\lambda]_{\circ}$, where λ is a regularized penalty and ordering $\circ$ is from $\{\cup, i\downarrow, i\uparrow, a\downarrow, a\uparrow, c\downarrow, c\uparrow\}$ meaning that decision list computation is all classes at once, or each class computed separately with the classes being ordered based on the increasing/decreasing *number/accuracy/cost* of training data in the classes, as defined in Section 5.3.



(a) Decision set/list size

(b) Average explanation size

Figure 6: Decision set/list size and average explanation size of perfect decision list and decision set models.

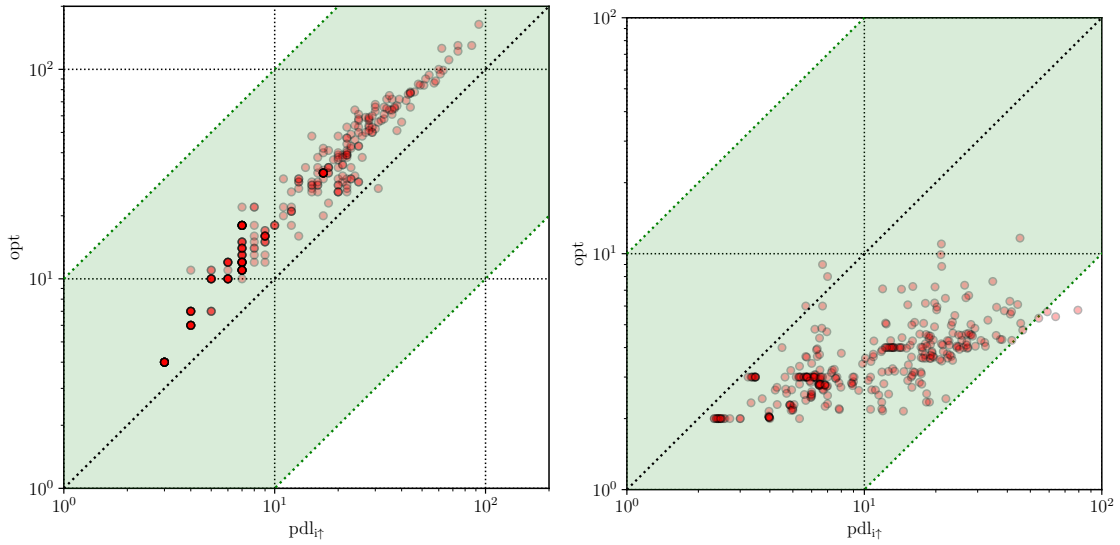
7.2.1 Perfect Models

We compare our prototype against state-of-the-art perfect decision set methods (Ignatiev et al., 2018) and the perfect decision set model described in Section 4.2, namely mds_2, $mds_2^\star$ and *opt*. While mds_2 generates a decision set with the smallest number of rules and *opt* minimizes the number of literals, $mds_2^\star$ does rule minimization followed by literal minimization. The comparison of perfect models is illustrated in Figure 5, Figure 6, and Figure 7.

Performance. The performance of the perfect models is shown in Figure 5a. As can be seen, mds_2 outperforms all the other rivals and trains 579 models. This should not come as a surprise since mds_2 minimizes the number of rules. It is followed by $mds_2^\star$, which sequentially applies rule and literal minimization – $mds_2^\star$ can solve 399 benchmarks. The best performing decision list model $pdl_{i\uparrow}$ comes third with 369 datasets handled successfully. The optimal decision set approach *opt* solves 350 instances. Finally, $pdl_{i\downarrow}$ and $pdl_\cup$ can train 329 and 261 decision lists respectively.

Test Accuracy. Test accuracy computed for the benchmarks solved by all the competitors is shown in the cactus plot of Figure 5b. Concretely, the plot depicts the value of *test error* in percent. On average, all the approaches perform similarly here and have test accuracy $\approx 80\%$. This is not surprising as all of them target perfectly accurate models.

Decision Set/List Size. The size calculated as the total number of literals in the decision set/list model is shown in Figure 6a. (Note that the plot is generated for the datasets successfully handled by all the shown competitors.) Observe that optimal perfect decision lists $pdl_\cup$ are the smallest among all the approaches with the average size being 9.9 per



(a) Decision set/list size detailed: $\text{pdl}_{i\uparrow}$ vs. opt (b) Explanation size detailed: $\text{pdl}_{i\uparrow}$ vs. opt

Figure 7: Comparison of $\text{pdl}_{i\uparrow}$ and opt in terms of decision set/list size and average explanation size.

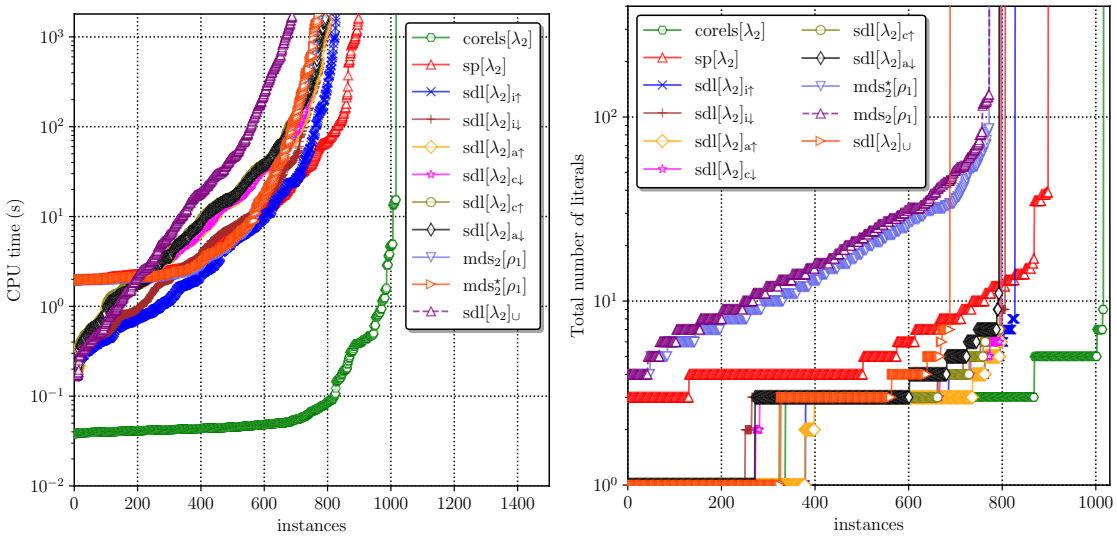
model. The second best model is $pdl_{i\uparrow}$ with 11.3 literals per model. Note that the smallest size decision sets obtained by *opt* have 20.6 literals on average. The largest models are of mds_2 with 29.6 literals per model on average. The pairwise comparison of decision set/list size for *opt* and $pdl_{i\uparrow}$ is detailed in the scatter plot of Figure 7a, which clearly demonstrates that perfect smallest size decision sets are usually larger than decision lists even when these are not guaranteed to be smallest in size.

Average Explanation Size. Although decision lists are smaller, the advantage of perfect decision sets is clearly the average explanation size per instance, which is calculated as described in Section 6. This data is shown in Figure 6b. For instance, it takes 3.3 literals on average to explain a prediction of decision sets produced by *opt*. For $mds_2^\star$ and mds_2 the numbers are 3.3 and 4.9, respectively. Explanations for decision lists are larger; the best result is shown by $pdl_\cup$, which has 6.9 literals per explanation. The best performing decision list model $pdl_{i\uparrow}$ has 9.6 literals per explanation. The detailed comparison of the average explanation size for *opt* and $pdl_{i\uparrow}$ is shown in the scatter plot of Figure 7b.

7.2.2 Sparse Models

The second part of our decision list evaluation compares sparse models. Here, the proposed approach is compared against sparse versions of decision sets *sp* and mds_2 of Section 4.2 and optimal sparse decision lists produced by *corels* (Angelino et al., 2017; Wang et al., 2017).[15] Although we tested 3 values for regularized penalty $\lambda \in \{0.005, 0.05, 0.5\}$, we report the results only for $\lambda_2 = 0.05$. As Section 7.1 showed, the best trade-off for sparse

15. There is a implementation of the RIPPER algorithm referred to as *JRip* that produces decision lists. However, we were unable to run it in our experimental environment. Since RIPPER is known to be more



(a) Performance

(b) Decision set/list size

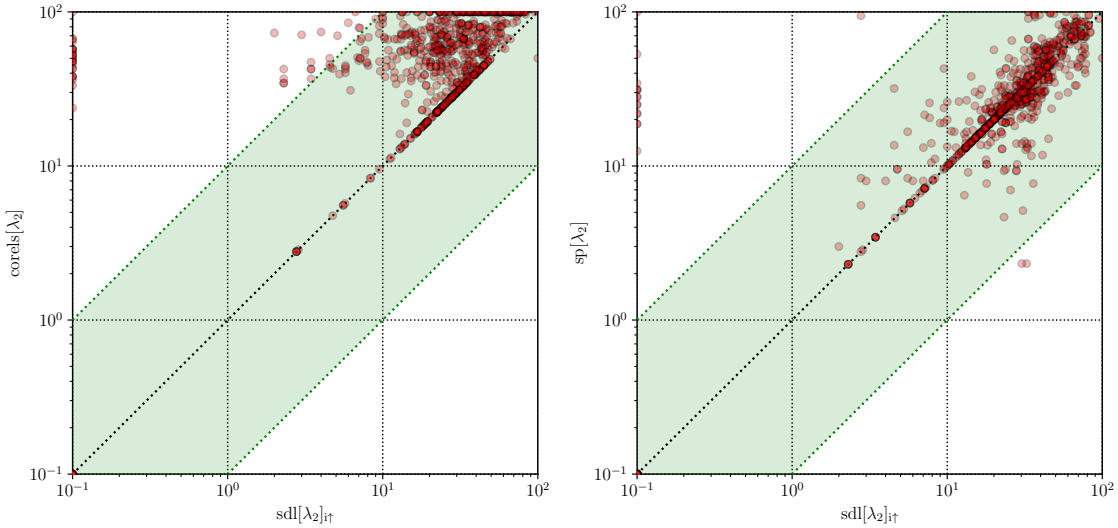
Figure 8: Performance and decision set/list size of sparse models.

decision sets was obtained for $\lambda_2 = 0.05$ and $\lambda_3 = 0.5$. However, decision lists obtained for λ_3 are usually too sparse as they end up having a single rule predicting a constant class. Therefore, hereinafter, the results are reported for configurations $sdl[\lambda_2]_*$ as well as for $sp[\lambda_2]$, $mds_2[\rho_1]$, $mds_2^\star[\rho_1]$, and $corels[\lambda_2]$. (Note that the value of regularized penalty $\rho_1 = 0.05$ is unchanged, also taken from Section 7.1. As mds_2 and $mds_2^\star$ minimize the number of rules, regularized penalty ρ_1 is applied wrt. the number of rules, which contrasts λ_2 applied to the number of literals). The results are shown in Figure 8, Figure 9, and Figure 10.

Performance. As can be observed in Figure 8a, $corels_{\lambda_2}$ is the fastest among the approaches for sparse models. It solves 1016 benchmarks. Sparse decision sets can be trained by $sp[\lambda_2]$ for 898 datasets while decision lists can be trained by $sdl[\lambda_2]_{i\uparrow}$ for 827 of them. Observe that class ordering $i\uparrow$ based on the increasing number of instances per class outperforms the other configurations of $sdl[\lambda_2]$, which can tackle ≈ 800 datasets each. The decision set competitors $mds_2[\lambda_2]$ and $mds_2^\star[\lambda_2]$ solve 772 benchmarks. Finally, aggregated computation of smallest decision lists of $sdl[\lambda_2]_\cup$ handles 688 datasets.

Test Accuracy. Although $corels[\lambda_2]$ outperforms its rivals in performance, the accuracy of its decision lists is not the best. The scatter plot in Figure 9a depicts the value of test error $e = 100\% - a$, where a is test accuracy, for $corels[\lambda_2]$ and $sdl[\lambda_2]_{i\uparrow}$. Observe that in many cases the accuracy of $sdl[\lambda_2]_{i\uparrow}$ is significantly higher than of $corels[\lambda_2]$: the average accuracy of $corels[\lambda_2]$ is 40.2% while the average accuracy of $sdl[\lambda_2]_{i\uparrow}$ is 69.9%. This clearly suggests that the sparsity measure used in our work enables us to train more

accurate (but less concise) than *corels* (Angelino et al., 2017; Wang et al., 2017), it not clear what the comparison of our approach against RIPPER would look like.



(a) Error: $\text{sdl}[\lambda_2]_{i\uparrow}$ vs. $\text{corels}[\lambda_2]$

(b) Error: $\text{sdl}[\lambda_2]_{i\uparrow}$ vs. $\text{sp}[\lambda_2]$

Figure 9: Test error comparison of $\text{sdl}[\lambda_2]_{i\uparrow}$, $\text{corels}[\lambda_2]$, and $\text{sp}[\lambda_2]$.

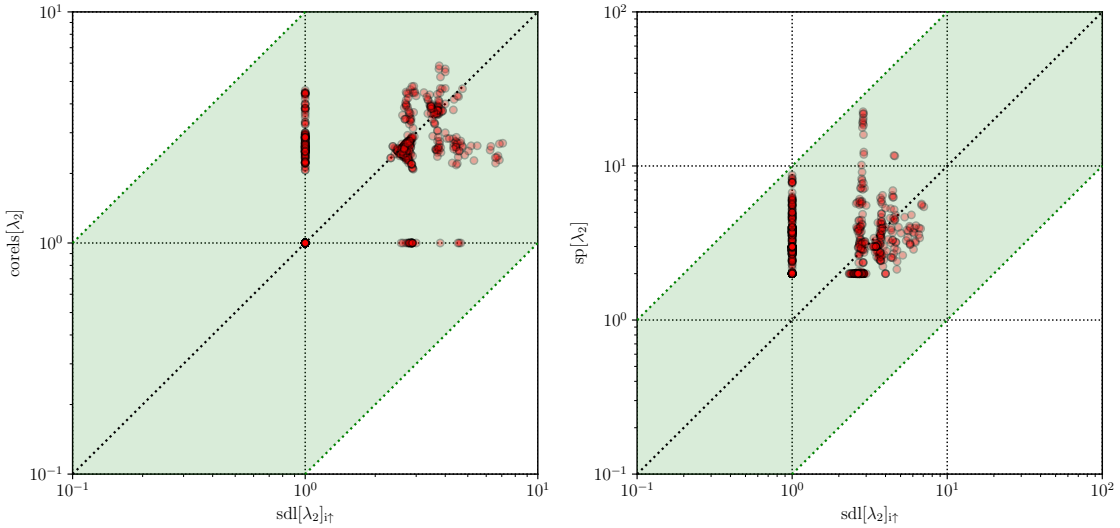
accurate decision lists. Also, as shown in Figure 9b, the accuracy of $sdl[\lambda_2]_{i\uparrow}$ is on a par with the accuracy of sparse decision sets of $sp[\lambda_2]$, which on average equals 67.6%.

Decision Set/List Size. As detailed in Figure 8b, the smallest models are obtained with sparse decision lists of $corels[\lambda_2]$ and $sdl[\lambda_2]_*$. The average number of literals in the lists produced by $corels[\lambda_2]$, $sdl[\lambda_2]_\cup$, and $sdl[\lambda_2]_{i\uparrow}$ is 2.7, 2.3, and 2.4, respectively (these numbers are calculated across the instances *solved* by the corresponding tools). Similar results are demonstrated by the other configurations of $sdl[\lambda_2]_*$. In contrast, the average size of sparse decision sets of $sp[\lambda_2]$, $mds_2[\lambda_2]$, and $mds_2^\star[\lambda_2]$ is 6.9, 22.5, and 17.9, respectively.

Average Explanation Size. Figure 10a and Figure 10b provide a comparison of $sdl[\lambda_2]_{i\uparrow}$ against $corels[\lambda_2]$ and $sp[\lambda_2]$ in terms of the average explanation size. In contrast to the case of perfect models, an average explanation for decision lists of $sdl[\lambda_2]_{i\uparrow}$ has 2.1 literals while explanations of sparse decision sets of $sp[\lambda_2]$ are of size 3.4. This suggests that sparse decision lists not only are smaller than sparse decision sets but they also provide a user with explanations that are more succinct. The average explanation size of the decision lists of $corels[\lambda_2]$ is 2.3. (The average numbers shown here are collected across all benchmarks solved by the corresponding tools).

8. Conclusion

In this paper, we developed the first approach to computing decision sets and decision lists where the total number of literals is minimized. The method can construct perfect decision sets and decision lists, i.e. those that perfectly classify the training instances; or sparse decision sets and lists, i.e. those that trade off model accuracy on training instances for size. The experimental results demonstrate that sparse models can outperform perfectly accurate models due to their (1) high accuracy overall and (2) better scalability. Second, the



(a) Explanations: sdl$[\lambda_2]_{i\uparrow}$ vs. corels$[\lambda_2]$

(b) Explanations: sdl$[\lambda_2]_{i\uparrow}$ vs. sp$[\lambda_2]$

Figure 10: Comparison of sdl$[\lambda_2]_{i\uparrow}$, corels$[\lambda_2]$, and sp$[\lambda_2]$ in terms of average explanation size.

regularized penalty substantially influences the efficiency of sparse decision sets and lists as these models are harder to compute but more accurate when the regularized penalty is smaller. This provides a reasonable trade-off in practice. Points 1 and 2 hold for the proposed models in this paper and also for the *sparse variants* of prior work aiming to minimize the number of rules (Ignatiev et al., 2018).

While existing heuristic approaches like RIPPER might scale significantly well and compute accurate decision sets, their solutions are often much larger than the sparse decision sets proposed in this paper, which causes them to be less explainable. The proposed method for sparse decision sets improves upon the previous state-of-the-art algorithms represented by prior logic-based approaches (Ignatiev et al., 2018; Malioutov & Meel, 2018; Ghosh & Meel, 2019) as well as by efficient heuristic methods (Cohen, 1995; Clark & Niblett, 1989; Clark & Boswell, 1991).

Although existing bespoke methods for optimal sparse decision lists are considerably more scalable, interestingly the accuracy of the models they construct are lower, probably because the size measure we use is more fine grained. There is also a question of how these methods scale with number of features. Finally, we provide the first comparison of decision sets and lists in terms of model size and explanation size. For perfect models, decision sets are preferable, but surprisingly this reverses for sparse models.

There is an interesting direction to extend this work. There is considerable symmetry in the proposed models, and while we tried adding symmetry breaking constraints to improve the models, what we tried did not make a significant difference. This deserves further exploration.

References

Aglin, G., Nijssen, S., & Schaus, P. (2020). Learning optimal decision trees using caching branch-and-bound search. In *AAAI*, pp. 3146–3153.

Agrawal, R., Imieliński, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. In *SIGMOD*, p. 207–216.

Angelino, E., Larus-Stone, N., Alabi, D., Seltzer, M., & Rudin, C. (2017). Learning certifiably optimal rule lists. In *KDD*, pp. 35–44.

Angelino, E., Larus-Stone, N., Alabi, D., Seltzer, M., & Rudin, C. (2018). Learning certifiably optimal rule lists for categorical data. *Journal of Machine Learning Research*, *18*(234), 1–78.

Apté, C., & Weiss, S. (1997). Data mining with decision trees and decision rules. *Future generation computer systems*, *13*(2-3), 197–210.

Asín, R., Nieuwenhuis, R., Oliveras, A., & Rodríguez-Carbonell, E. (2009). Cardinality networks and their applications. In *SAT*, pp. 167–180.

Audemard, G., Lagniez, J., & Simon, L. (2013). Improving Glucose for incremental SAT solving with assumptions: Application to MUS extraction. In *SAT*, pp. 309–317.

Avellaneda, F. (2020). Efficient inference of optimal decision trees. In *AAAI*, pp. 3195–3202.

Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., & Müller, K. (2010). How to explain individual classification decisions. *Journal of Machine Learning Research*, *11*, 1803–1831.

Bailleux, O., & Boufkhad, Y. (2003). Efficient CNF encoding of Boolean cardinality constraints. In *CP*, pp. 108–122.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., & Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Oper. Res.*, *46*(3), 316–329.

Batcher, K. E. (1968). Sorting networks and their applications. In *AFIPS*, Vol. 32, pp. 307–314.

Bessiere, C., Hebrard, E., & O'Sullivan, B. (2009). Minimising decision tree size as combinatorial optimisation. In *CP*, pp. 173–187.

Biere, A., Heule, M., van Maaren, H., & Walsh, T. (Eds.). (2009). *Handbook of Satisfiability*. IOS Press.

Clark, P., & Boswell, R. (1991). Rule induction with CN2: some recent improvements. In *EWSL*, pp. 151–163.

Clark, P., & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning*, *3*, 261–283.

Cohen, W. W. (1995). Fast effective rule induction. In *ICML*, pp. 115–123.

Darwiche, A. (2020). Three modern roles for logic in AI. In *PODS*, pp. 229–243. ACM.

Darwiche, A., & Hirth, A. (2020). On the reasons behind decisions. In *ECAI*, pp. 712–720.

Dash, S., Günlük, O., & Wei, D. (2018). Boolean decision rules via column generation. In *NeurIPS*, p. 4660–4670.

Doshi-Velez, F., & Kim, B. (2017). A roadmap for a rigorous science of interpretability. *CoRR*, *abs/1702.08608*.

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

Dufour, D. (2014). Finding cost-efficient decision trees. Master's thesis, University of Waterloo.

Evans, R., & Grefenstette, E. (2018). Learning explanatory rules from noisy data. *J. Artif. Intell. Res.*, *61*, 1–64.

Fürnkranz, J., Gamberger, D., & Lavrac, N. (2012). *Foundations of Rule Learning*. Springer.

Ghosh, B., & Meel, K. S. (2019). IMLI: an incremental framework for maxsat-based learning of interpretable classification rules. In *AIES*, pp. 203–210. ACM.

Guidotti, R., Monreale, A., Giannotti, F., Pedreschi, D., Ruggieri, S., & Turini, F. (2019a). Factual and counterfactual explanations for black box decision making. *IEEE Intelligent Systems*, *34*(6), 14–23.

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., & Pedreschi, D. (2019b). A survey of methods for explaining black box models. *ACM Comput. Surv.*, *51*(5), 93:1–93:42.

Han, J., Kamber, M., & Pei, J. (2012). *Data Mining: Concepts and Techniques, 3rd edition*. Morgan Kaufmann.

Hancock, T. R., Jiang, T., Li, M., & Tromp, J. (1996). Lower bounds on learning decision lists and trees. *Inf. Comput.*, *126*(2), 114–122.

Hu, X., Rudin, C., & Seltzer, M. (2019). Optimal sparse decision trees. In *Advances in Neural Information Processing Systems*, pp. 7265–7273.

Ignatiev, A. (2020). Towards trustable explainable AI. In *IJCAI*, pp. 5154–5158.

Ignatiev, A., Lam, E., Stuckey, P. J., & Marques-Silva, J. (2021). A scalable two stage approach to computing optimal decision sets. In *34th AAAI Conference on Artificial Intelligence (AAAI 2021)*, p. To Appear.

Ignatiev, A., Morgado, A., & Marques-Silva, J. (2018). PySAT: A Python toolkit for prototyping with SAT oracles. In *SAT*, pp. 428–437.

Ignatiev, A., Morgado, A., & Marques-Silva, J. (2019a). RC2: an efficient maxsat solver. *J. Satisf. Boolean Model. Comput.*, *11*(1), 53–64.

Ignatiev, A., Narodytska, N., & Marques-Silva, J. (2019b). Abduction-based explanations for machine learning models. In *AAAI*, pp. 1511–1519.

Ignatiev, A., Narodytska, N., & Marques-Silva, J. (2019c). On relating explanations and adversarial examples. In *NeurIPS*, pp. 15857–15867.

Ignatiev, A., Pereira, F., Narodytska, N., & Marques-Silva, J. (2018). A SAT-based approach to learn explainable decision sets. In *IJCAR*, pp. 627–645.

Janota, M., & Morgado, A. (2020). SAT-based encodings for optimal decision trees with explicit paths. In *SAT*, pp. 501–518.

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, *349*(6245), 255–260.

Kamath, A. P., Karmarkar, N., Ramakrishnan, K. G., & Resende, M. G. C. (1992). A continuous approach to inductive inference. *Math. Program.*, *57*, 215–238.

Kim, B., Khanna, R., & Koyejo, O. O. (2016). Examples are not enough, learn to criticize! criticism for interpretability. In *Advances in neural information processing systems*, pp. 2280–2288.

Lakkaraju, H., Bach, S. H., & Leskovec, J. (2016). Interpretable decision sets: A joint framework for description and prediction. In *KDD*, pp. 1675–1684.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436.

Li, O., Liu, H., Chen, C., & Rudin, C. (2018). Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In *AAAI*.

Lipton, Z. C. (2018). The mythos of model interpretability. *Commun. ACM*, *61*(10), 36–43.

Lundberg, S. M., & Lee, S. (2017). A unified approach to interpreting model predictions. In *NIPS*, pp. 4765–4774.

Malioutov, D., & Meel, K. S. (2018). MLIC: A maxsat-based framework for learning interpretable classification rules. In *CP*, pp. 312–327.

Marques-Silva, J., Gerspacher, T., Cooper, M. C., Ignatiev, A., & Narodytska, N. (2020). Explaining naive bayes and other linear classifiers with polynomial time and delay. In *NeurIPS*.

Marques-Silva et al., J. (2017). Learning decision trees with SAT. Tech. rep., LASIGE, FCUL.

Michalski, R. S. (1969). On the quasi-minimal solution of the general covering problem. In *International Symposium on Information Processing*, pp. 125–128.

Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, *267*, 1–38.

Mitchell, T. M. (1997). *Machine learning*. McGraw-Hill.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529.

Monroe, D. (2018). AI, explain yourself. *Commun. ACM*, *61*(11), 11–13.

Montavon, G., Samek, W., & Müller, K. (2018). Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, *73*, 1–15.

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

Olson, R. S., La Cava, W., Orzechowski, P., Urbanowicz, R. J., & Moore, J. H. (2017). Pmlb: a large benchmark suite for machine learning evaluation and comparison. *BioData Mining*, *10*(1), 36.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. Morgan Kauffmann.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "why should I trust you?": Explaining the predictions of any classifier. In Krishnapuram, B., Shah, M., Smola, A. J., Aggarwal, C. C., Shen, D., & Rastogi, R. (Eds.), *KDD*, pp. 1135–1144. ACM.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2018). Anchors: High-precision model-agnostic explanations. In *AAAI*.

Rivest, R. L. (1987). Learning decision lists. *Machine Learning*, *2*(3), 229–246.

Rudin, C., & Ertekin, S. (2018). Learning customized and optimized lists of rules with mathematical programming. *Mathematical Programming Computation*, *10*, 659–702.

Schidler, A., & Szeider, S. (2021). SAT-based decision tree learning for large data sets. In *AAAI*, pp. 3904–3912.

Shih, A., Choi, A., & Darwiche, A. (2018). A symbolic approach to explaining bayesian network classifiers. In *IJCAI*, pp. 5103–5111.

Sinz, C. (2005). Towards an optimal CNF encoding of Boolean cardinality constraints. In *CP*, pp. 827–831.

Stump, A., Sutcliffe, G., & Tinelli, C. (2014). Starexec: A cross-community infrastructure for logic solving. In *IJCAR*, pp. 367–373.

Tseitin, G. S. (1968). On the complexity of derivations in the propositional calculus. *Studies in Mathematics and Mathematical Logic*, *Part II*, 115–125.

Verhaeghe, H., Nijssen, S., Pesant, G., Quimper, C.-G., & Schaus, P. (2019). Learning optimal decision trees using constraint programming. In *Proceedings of CP-19*.

Wang, T., Rudin, C., Doshi-Velez, F., Liu, Y., Klampfl, E., & MacNeille, P. (2017). A bayesian framework for learning rule sets for interpretable classification. *Journal of Machine Learning Research*, *18*, 70:1–70:37.

Yu, J., Ignatiev, A., Le Bodic, P., & Stuckey, P. J. (2020a). Optimal decision lists using SAT. *CoRR*, *abs/2010.09919*.

Yu, J., Ignatiev, A., Stuckey, P. J., & Le Bodic, P. (2020b). Computing optimal decision sets with SAT. In *CP*, pp. 952–970.