SAT-Based Rigorous Explanations for Decision Lists *

Alexey Ignatiev[1] and Joao Marques-Silva[2]

[1] Monash University, Melbourne, Australia
alexey.ignatiev@monash.edu
[2] IRIT, CNRS, Toulouse, France
joao.marques-silva@irit.fr

Abstract. Decision lists (DLs) find a wide range of uses for classification problems in Machine Learning (ML), being implemented in a number of ML frameworks. DLs are often perceived as interpretable. However, building on recent results for decision trees (DTs), we argue that interpretability is an elusive goal for some DLs. As a result, for some uses of DLs, it will be important to compute (rigorous) explanations. Unfortunately, and in clear contrast with the case of DTs, this paper shows that computing explanations for DLs is computationally hard. Motivated by this result, the paper proposes propositional encodings for computing abductive explanations (AXps) and contrastive explanations (CXps) of DLs. Furthermore, the paper investigates the practical efficiency of a MARCO-like approach for enumerating explanations. The experimental results demonstrate that, for DLs used in practical settings, the use of SAT oracles offers a very efficient solution, and that complete enumeration of explanations is most often feasible.

1 Introduction

Decision lists (DLs) [64] find a wide range of uses for classification problems in Machine Learning (ML) [1,2,12,15–18,65,71–73], being implemented in a number of ML frameworks (e.g. [10,22]). DLs can be viewed as ordered rules, and so are often perceived as interpretable[3]. This explains in part the recent interest in DLs [1, 2, 12, 65, 71–73], most of which is premised on the interpretability of DLs. However, building on recent results for decision trees (DTs) [37], which demonstrate the possible non-interpretability of DTs when representing specific functions, we show that interpretability can also be an elusive goal for some

* This work is supported by the AI Interdisciplinary Institute ANITI, funded by the French program "Investing for the Future - PIA3" under Grant agreement n^o ANR-19-PI3A-0004.

[3] Interpretability is a subjective concept, for which no rigorous accepted definition exists [46]. As clarified later in the paper, for a given pair ML model and instance, we equate interpretability with how succinct is the justification for the model's prediction.

DLs. As a result, and for some concrete applications of DLs, it is important to compute (rigorous) explanations.

Explanations can be broadly categorized into heuristic [47, 62, 63] and non-heuristic [32, 66]. Recent work has provided extensive evidence regarding the lack of quality of heuristic explanation approaches [11, 25, 34, 40, 57, 68]. Non-heuristic (or rigorous) approaches for computing explanations can be organized into abductive (AXp) [19, 32, 33, 66] and contrastive (CXp) [31, 54]. (Abductive explanations are also referred to as PI-explanations [66] (i.e. prime implicant explanations), since these represent subset-minimal sets of feature value pairs that are sufficient for a prediction.) Most work on rigorous explanations either exploits knowledge compilation approaches [3, 19, 66, 67], or approaches based on iterative calls to some oracle for NP (e.g. SAT, SMT, MILP, etc.) [31–33]. As a result, improvements to automated reasoning tools, can have a profound impact on the deployment of rigorous explanation approaches.

Furthermore, recent work proposed polynomial time algorithms for finding explanations of a number of ML models, including DTs [37], naive-Bayes classifiers [49], and also different knowledge representation languages [3]. Unfortunately, and in contrast with these recent tractability results, this paper proves that finding one PI-explanation for a DL is NP-hard.

Motivated by the NP-hardness of finding explanations of DLs, the paper proposes propositional encodings for computing abductive and contrastive explanations of DLs. Furthermore, the paper investigates the practical efficiency of a MARCO-like [43] approach for enumerating explanations. The experimental results demonstrate that, for DLs used in practical settings, the use of SAT oracles offers a very efficient solution, and that complete enumeration of explanations is most often feasible.

The paper is organized as follows. The notation and definitions used throughout the paper are introduced in Section 2. Section 3 proves the NP-hardness of finding rigorous explanations for DLs. In addition, this section develops a propositional encoding for finding one AXp or one CXp, and briefly overviews the online enumeration of explanations. Section 4 presents the experimental results. The paper concludes in Section 5.

2 Preliminaries

2.1 Propositional Satisfiability

Definitions standard in propositional satisfiability (SAT) and maximum satisfiability (MaxSAT) solving are assumed [8]. In what follows, we will assume formulas to be propositional. A conjunction of literals is referred to as *term* while a disjunction of literals is referred to as *clause*; also note that a literal is either a Boolean variable or its negation. Whenever convenient, terms and clauses are treated as sets of literals. A formula is said to be in *conjunctive* or *disjunctive normal form* (CNF or DNF, respectively) if it is a conjunction of clauses or disjunction of terms, respectively. Set theory notation will be also used with respect to CNF and DNF formulas when necessary.

A *truth assignment* μ is a mapping from the set of variables to $\{0,1\}$. An assignment is said to satisfy a literal l ($\neg l$, resp.) if it maps variable l to 1 (to 0, resp.). A clause is said to be satisfied by assignment μ if μ satisfies at least one of its literals. If for a CNF formula ϕ there exists an assignment μ that satisfies all clauses of ϕ, formula ϕ is referred to as satisfiable and μ is its *satisfying assignment* (or *model*). In addition, we use the notation $\models$ to denote *entailment*, i.e. $\phi_1 \models \phi_2$ if any model of ϕ_1 is also a model of ϕ_2.

One of the central concepts in rigorous explainable AI (XAI) [32, 66] is of prime implicants as defined below.

Definition 1. *A term π is an* implicant *of formula ϕ if $\pi \models \phi$. An implicant π of ϕ is called* prime *if none of the proper subsets $\pi' \subsetneq \pi$ is an implicant of ϕ.*

In the context of unsatisfiable formulas, the maximum satisfiability (MaxSAT) problem is to find a truth assignment that maximizes the number of satisfied clauses. A number of variants of MaxSAT exist [8, Chapters 23 and 24]. Hereinafter, we are mostly interested in Partial (Unweighted) MaxSAT, which can be formulated as follows. The formula ϕ is represented as a conjunction of *hard* clauses $\mathcal{H}$, which must be satisfied, and *soft* clauses $\mathcal{S}$, which represent a preference to satisfy those clauses, i.e. $\phi = \mathcal{H} \land \mathcal{S}$. Therefore, the Partial MaxSAT problem consists in finding an assignment that satisfies all the hard clauses and maximizes the total number of satisfied soft clauses. In the following, the concepts of minimal unsatisfiable subsets (MUSes) and minimal correction subsets (MCSes) taking into account the hard clauses $\mathcal{H}$ will also be helpful. Concretely, consider unsatisfiable CNF formula $\phi = \mathcal{H} \land \mathcal{S}$ with $\mathcal{H}$ and $\mathcal{S}$ defined as the set of hard and soft clauses, respectively.

Definition 2. *A subset of soft clauses $\mathcal{M} \subseteq \mathcal{S}$ is a* Minimal Unsatisfiable Subset *(MUS) iff $\mathcal{H} \cup \mathcal{M}$ is unsatisfiable and $\forall_{\mathcal{M}' \subsetneq \mathcal{M}}$, $\mathcal{H} \cup \mathcal{M}'$ is satisfiable.*

Definition 3. *A subset of soft clauses $\mathcal{C} \subseteq \mathcal{S}$ is a* Minimal Correction Subset *(MCS) iff $\mathcal{H} \cup \mathcal{F} \setminus \mathcal{C}$ is satisfiable and $\forall_{\mathcal{C}' \subsetneq \mathcal{C}}$, $\mathcal{H} \cup \mathcal{F} \setminus \mathcal{C}'$ is unsatisfiable.*

MUSes and MCSes of a CNF formula are known to be related through the minimal hitting set (MHS) duality [5,9,45,61], which has been recently exploited in a number of practical settings [21, 27, 30, 36, 45] including XAI [31].

2.2 Classification Problems, Decision Lists, and Explanations

This section introduces definitions and notation related with classification problems in ML, but also formal definitions of explanations proposed in recent work [32, 66].

Classification problems. We consider a classification problem, characterized by a set of (categorical) features $\mathcal{F} = \{1, \ldots, m\}$, and by a set of classes $\mathcal{K} = \{c_1, \ldots, c_K\}$. Each feature $j \in \mathcal{F}$ is characterized by a domain D_i. As a result, feature space is defined as $\mathbb{F} = D_1 \times D_2 \times \ldots \times D_m$. A specific point in feature

$$\tau(\mathbf{x}) = \begin{cases} \ominus \text{ if } [2x_1 - x_2 \leq 1] \\ \oplus \text{ if } [2x_1 - x_2 > 1] \end{cases}$$

R_0:	IF	x_1	THEN $\oplus$
R_1:	ELSE IF	x_2	THEN $\oplus$
R_{DEF}:	ELSE		THEN $\ominus$

R_0: IF	x_1	THEN $\oplus$
R_1: IF	x_2	THEN $\oplus$
R_2: IF	$\neg x_1 \land \neg x_2$	THEN $\ominus$

(a) Example linear classifier (b) Example decision list (c) Example decision set

Fig. 1: Example classifiers

space is represented by $\mathbf{v} = (v_1, \ldots, v_m)$. A point $\mathbf{v}$ in feature space denotes an *instance* (or an *example*). Moreover, we use $\mathbf{x} = (x_1, \ldots, x_m)$ to denote an arbitrary point in feature space. In general, when referring to the value of a feature $j \in \mathcal{F}$, we will use a variable x_j, with x_j taking values from D_j. (To keep the notation simple, we opt not to introduce an assignment function, mapping each feature j to some value in D_j.) For simplicity, throughout this paper we will restrict $\mathcal{K}$ to two classes, i.e. $\mathcal{K} = \{\oplus, \ominus\}$. However, most of the ideas described in this document also apply in the more general case of $\mathcal{K}$ with more than two elements; the general case of non-binary classification is also considered in the experimental results presented in Section 4. (In settings where $\mathcal{K} = \{\oplus, \ominus\}$, we will also equate $\oplus$ with 1, and $\ominus$ with 0.)

A classifier implements a *total classification function* $\tau : \mathbb{F} \to \mathcal{K}$. In some settings, e.g. when computing explanations, it will be convenient to represent the classification function as a *decision predicate* $\tau_c : \mathbb{F} \to \{0, 1\}$, parametrized by some fixed class $c \in \mathcal{K}$, and such that $\forall(\mathbf{x} \in \mathbb{F}).\tau_c(\mathbf{x}) \leftrightarrow (\tau(\mathbf{x}) = c)$.

Example 1. To illustrate the definitions above, we consider a very simple linear classifier, defined as follows. Let $\mathcal{F} = \{1, 2\}$, with $D_1 = D_2 = \{0, 1, 2\}$, and let $\mathcal{K} = \{\ominus, \oplus\}$. As a result, feature space is given by $\mathbb{F} = \{0, 1, 2\} \times \{0, 1, 2\}$. Furthermore, the classification function associated with the classifier is shown in Figure 1a. Concretely, the prediction is $\oplus$ if $2x_1 - x_2 > 1$, and it is $\ominus$ otherwise. □

Decision lists (DLs) & decision sets (DSs). A *rule* is of the form "IF antecedent THEN prediction", where the antecedent is of the form $\bigwedge$feature-literals. The interpretation of a rule is that if the antecedent is consistent (i.e. all the literals are true), then the rule *fires* and the prediction is the one associated with the rule. A decision list (DL) [64] is an *ordered* list of rules, whereas a decision set (DS) [14, 39] is an *unordered* list of rules.

Throughout the paper, we will consider ordered sets of rule indices $\mathfrak{R} = \{1, \ldots, R\}$, such that for $i \in \mathfrak{R}$, we will use $\mathfrak{c}$, $\mathfrak{l}$ and $\mathfrak{o}$ to denote, respectively, the class associated with rule i, the set of literals associated with rule i and the order of rule i.

Example 2. Consider another classifier. Let $\mathcal{F} = \{1, 2\}$, with $D_1 = D_2 = \{0, 1\}$, and so $\mathbb{F} = \{0, 1\} \times \{0, 1\}$. The decision list for the classifier is shown in Figure 1b

while an equivalent decision set is shown in Figure 1c. The classification function for the DL can be represented as follows:

$$\tau(\mathbf{x}) = \begin{cases} \oplus \text{ if } [(x_1) \lor (\neg x_1 \land x_2)] \\ \ominus \text{ if } [(\neg x_1 \land \neg x_2)] \end{cases}$$

(Note how the lack of order in DS rules results in a simpler classifier representation $\tau(\mathbf{x})$ for class $\oplus$, e.g. it can be explicitly represented as $x_1 \lor x_2$ since rules R_0 and R_1 are unordered in the decision set of Figure 1c.) □

Note that following the standard convention, we will *always* assume that DLs have a *default rule*, with no literals, that *fires* when for all the preceding rules, the conjunction of literals associated with that rule is inconsistent. An example default rule for the DL shown in Example 2 is marked as $\mathrm{R}_{\mathrm{DEF}}$.

Interpretability & explanations. Interpretability is generally accepted to be a subjective concept, without a formal definition [46]. In this paper we measure interpretability in terms of the overall succinctness of the information provided by an ML model to justify a given prediction. We say that a model is *not* interpretable if for some instance, the justification of a prediction is arbitrarily larger (on the number of features) than a rigorous explanation (which we define next). Moreover, and building on earlier work, we equate explanations with the so-called PI-explanations [3, 19, 32, 66], i.e. subset-minimal sets of feature-value pairs that are sufficient for the prediction. More formally, given an instance $\mathbf{v} \in \mathbb{F}$, with prediction $c \in \mathcal{K}$, i.e. $\tau(\mathbf{v}) = c$, a PI-explanation is a minimal subset $\mathcal{X} \subseteq \mathcal{F}$ such that,

$$\forall(\mathbf{x} \in \mathbb{F}). \bigwedge\nolimits_{j \in \mathcal{X}} (x_j = v_j) \rightarrow (\tau(\mathbf{x}) = c) \tag{1}$$

Another name for a PI-explanation is (a minimal/minimum) *abductive explanation* (AXp) [31, 32]. For simplicity, and depending on the context, we will use *PI-explanation* and the acronym *AXp* interchangeably.

In a similar vein, we consider *contrastive explanations* (CXps) [31, 54]. Contrastive explanation can be defined as a (subset-)minimal set of feature-value pairs ($\mathcal{Y} \subseteq \mathcal{F}$) that suffice to changing the prediction if they are allowed to take *some arbitrary* value from their domain. Formally and as suggested in [31], a CXp is defined as a minimal subset $\mathcal{Y} \subseteq \mathcal{F}$ such that,

$$\exists(\mathbf{x} \in \mathbb{F}). \bigwedge\nolimits_{j \notin \mathcal{Y}} (x_j = v_j) \land (\tau(\mathbf{x}) \neq c) \tag{2}$$

(It is possible and simple to adapt the definition to target a specific class $c' \neq c$.) Moreover, building on the seminal work of Reiter [61], recent work demonstrated a minimal hitting set relationship between AXps and CXps [31], namely each AXp is a minimal hitting set (MHS) of the set of CXps and vice-versa.

For computing both kinds of explanations (AXps and CXps), we will work with sets of features, aiming at finding minimal subsets. It will also be helpful to

describe explanations (concretely AXps) as sets of literals. As a result, starting from an instance $\mathbf{v}$, we create a set of literals $I_{\mathbf{v}} = \{(x_j, v_j) | j \in \mathcal{F}\}$. When clear from the context, we will just use I to denote the literals of an instance.

An AXp $\mathcal{X} \subseteq \mathcal{F}$ can also be viewed as a conjunction ρ of a subset of the literals $I_{\mathbf{v}}$ induced by the instance $\mathbf{v}$ that is *sufficient* for the prediction. Moreover, given a conjunction of literals ρ, we will associate a predicate $\rho : \mathbb{F} \to \{0, 1\}$ (with the symbol duplication deliberately aiming at simplifying the notation) to represent the values taken by the conjunction of literals for each point $\mathbf{x}$ in feature space. As a result, we use $\rho \models \tau_c$ to denote that ρ is sufficient for the prediction, i.e.

$$\forall(\mathbf{x} \in \mathbb{F}).\rho(\mathbf{x}) \rightarrow \tau_c(\mathbf{x}) \tag{3}$$

We can also associate a conjunction of literals η with each CXp, such that the literals in η are *not* the literals specified by the CXp, and such that the following condition holds,

$$\exists(\mathbf{x} \in \mathbb{F}).\eta(\mathbf{x}) \land \neg\tau_c(\mathbf{x}) \tag{4}$$

It should be noted that since a CXp is a minimal set of features, each η is a maximal set of literals such that there exists at least one point in feature space such that the ML model predicts a class other than c.

Example 3. For the linear classifier of Example 1, let $\mathbf{v} = (2, 0)$, with prediction $\oplus$. In this case, the (only) AXp is $\mathcal{X} = \{1\}$, indicating that, as long as $x_1 = 2$, the value of the prediction is $\oplus$, independently of the value of x_2. Moreover, the AXp can also be represented by $\rho \triangleq (x_1 = 2)$. For this very simple example, $\mathcal{Y} = \{1\}$ is also a CXp. Indeed, if we allow feature 1 to take a value other than 2, then the assignment $\mathbf{v}' = (0, 0)$ will change the prediction. (More complex examples of CXps are studied later in the paper.) □

Example 4. For the decision list of Example 2, let $\mathbf{v} = (0, 1)$, with prediction $\oplus$. In this case, the (only) AXp is $\mathcal{X} = \{2\}$, indicating that, as long as $x_2 = 1$, the value of the prediction is $\oplus$, independently of the value of x_1. Moreover, the AXp can also be represented by $\rho \triangleq (x_2 = 1)$. In this case, a CXp is also $\mathcal{Y} = \{2\}$. For example, the point in feature space $\mathbf{v} = (0, 0)$ will cause the prediction to change to $\ominus$. □

Example 5. To illustrate the hitting set duality relationship between AXps and CXps established in [31], we consider a simple classifier represented as a decision list (DL) of three rules (including the default rule). Let $\mathcal{F} = \{1, 2, 3, 4, 5\}$, $D_i = \{0, 1, 2\}$, with $i = 1, \ldots, 5$, and $\mathcal{K} = \{\ominus, \oplus\}$. Let the decision list be:

R_0:	IF	$x_1 = 1 \land x_2 = 1$	THEN $\ominus$
R_1:	ELSE IF	$x_3 \neq 1$	THEN $\oplus$
$\mathrm{R}_{\mathrm{DEF}}$:	ELSE		THEN $\ominus$

We consider the instance $\mathbf{v} = (1, 1, 1, 1, 1)$, which results in prediction $\ominus$. It is straightforward to see that, as long as $x_1 = x_2 = 1$, then the prediction is $\ominus$.

Also, it is less trivial (but still observable) that, as long as $x_3 = 1$, the prediction is guaranteed to be $\ominus$ as well. Moreover, it suffices to change the value of feature 3 and the value of *either* feature 1 or feature 2 to change the prediction to $\oplus$, e.g. set $x_3 = x_1 = 0$ or set $x_3 = x_2 = 2$. As a result, we can conclude that the set of AXps is: $\mathbb{X} = \{\{1, 2\}, \{3\}\}$, and the set of CXps is: $\mathbb{Y} = \{\{1, 3\}, \{2, 3\}\}$. Furthermore, from the minimal hitting set duality relationship between AXps and CXP's [31], the sets in $\mathbb{X}$ are MHSes of the sets in $\mathbb{Y}$ and vice-versa. (Clearly, we could follow the definitions and reach the same conclusions.) □

3 Explaining Decision Lists

It is easy to see that just like DTs [37], DLs can also exhibit redundancy in the literals used, and so the computation of PI-explanations can be instrumental to conveying short explanations to a human decision maker.

Example 6. Consider a possible DL shown below for the function $f(x_1, \ldots, x_4) = (x_1 \land x_2) \lor (x_3 \land x_4)$. (This DL is constructed by applying a "direct translation" of all the paths of the DT shown in [37, Figure 1b] from left to right into rules followed by appending a default rule predicting class $f = 1$.)

R_0:	IF	$x_1 = 0 \land x_3 = 0$	THEN $f = 0$
R_1:	ELSE IF	$x_1 = 0 \land x_3 = 1 \land x_4 = 0$	THEN $f = 0$
R_2:	ELSE IF	$x_1 = 0 \land x_3 = 1 \land x_4 = 1$	THEN $f = 1$
R_3:	ELSE IF	$x_1 = 1 \land x_2 = 0 \land x_3 = 0$	THEN $f = 0$
R_4:	ELSE IF	$x_1 = 1 \land x_2 = 0 \land x_3 = 1 \land x_4 = 0$	THEN $f = 0$
R_5:	ELSE IF	$x_1 = 1 \land x_2 = 0 \land x_3 = 1 \land x_4 = 1$	THEN $f = 1$
R_6:	ELSE IF	$x_1 = 1 \land x_2 = 1$	THEN $f = 1$
R_{DEF}:	ELSE		THEN $f = 1$

Consider a data instance $\mathbf{v} = (1, 0, 1, 1)$ and observe that rule R_5 fires the prediction $f = 1$. Although rule R_5 has four literals, an AXp for instance $\mathbf{v}$ is $(x_3 = 1) \land (x_4 = 1)$. Similarly, in practice one may expect examples of DLs s.t. AXps will be significanlty smaller than the rules that fire the corresponding predictions.

This observation is confirmed by the experimental results in Section 4, in that explanations can play an important role in understanding the predictions made by DLs. □

3.1 DL Explainability

Perhaps surprisingly, whereas DTs can be explained in polynomial time, DLs cannot. This section proves a number of theoretical results related to explainability of DLs. Here we will be using the *knowledge compilation* (KC) map [20], which studied a wealth of queries on knowledge representation languages. We consider the concrete setting of classification, i.e. a language L denotes a classifier τ and a target prediction c. Let us briefly define the queries of interest [20]:

1. **Satisfiability (SAT)**: *if there exists a polynomial-time algorithm for deciding the satisfiability of* $\tau(\mathbf{x}) = c$*, i.e. to decide in polynomial time whether there exists* $\mathbf{x} \in \mathbb{F}$ *such that* $\tau(\mathbf{x}) = c$*. In the case of DLs, this problem will be referred to as* DLSAT*.*
2. **Implicant test (IM)**: *if there exists a polynomial-time algorithm that decides whether a conjunction of literals* ρ *is such that* $\rho \models \tau_c$*, i.e.* $\forall(\mathbf{x} \in \mathbb{F}).\rho(\mathbf{x}) \rightarrow \tau_c(\mathbf{x})$*. In the case of DLs, this problem will be referred to as* DLIM*.*

Similarly, we can define DNFSAT (which is trivially in P) and DNFIM (which is well-known to be in P only if P = NP [20]).

Proposition 1. *DLSAT is* NP*-complete.*

Proof. It is easy to see that the DLSAT is in NP. We simply guess an assignment to the features and check whether the prediction is the expected one according to the DL. To prove NP-hardness, the reduction of CNFSAT to DLSAT is organized as follows:

1. Consider a CNF formula ϕ with clauses $c_1, c_2, \ldots, c_m$.
2. Let the variables in ϕ denote the features (w.l.o.g. assume the features to be Boolean).
3. Consider the negation of each clause $\neg c_i$ which represents a conjunction of literals $\bigwedge_{l_j \in c_i} \neg l_j$.
4. For each $\neg c_i$, create a rule π_i with antecedent $\bigwedge_{l_j \in c_i} \neg l_j$ and prediction $\ominus$.
5. Create a default rule with prediction $\oplus$.
6. Hence, formula ϕ is satisfiable if and only if there is an assignment to the features which results in prediction $\oplus$.

The prediction is $\ominus$ if some clause c_i is falsified, i.e. $\neg c_i$ is satisfied (and hence rule π_i fires). Otherwise, if all clauses are satisfied, and so all $\neg c_i$ are falsified, then the prediction is $\oplus$. □

Proposition 2. *No polynomial-time algorithm exists for DLIM unless* P = NP*.*

Proof. We reduce DNFIM (i.e. IM for DNF) to DLIM, given that IM for DNF is well-known to be solvable in polynomial time only if P = NP [20]. Let ψ denote a DNF, with k terms, i.e. $\psi = t_1 \vee \ldots \vee t_k$, and let p denote a conjunction of literals. IM for DNF is to decide whether p is an implicant of ψ, i.e. $p \models \psi$. The reduction of DNFIM to DLIM is organized as follows:

1. For each conjunction of literals t_i in ψ, create a rule with antecedent given by t_i, i.e. $\pi_i = t_i$, and prediction $\ominus$.
2. The $(k{+}1)^{\text{th}}$ rule is created as follows: the antecendent is p and the prediction is $\oplus$.
3. Finally, we add a default rule with prediction $\ominus$.

As a result, the prediction will be $\oplus$ if and only if $p \wedge \bigwedge_{i \in [k]}(\neg t_i)$ is satisfied, and so $p \not\models \psi$, in which case p is not an implicant of ψ. □

Given the above results, we can conclude the following.

Proposition 3. *There is no polynomial-time algorithm for finding an AXp of a decision list unless* P = NP*.*

Proof (sketch). If there was a polynomial-time algorithm for finding an AXp for a DL then we would be able to solve IM for DL in polynomial time. This would in turn imply that IM for DNF is solvable in polynomial time. □

A Word on Decision Sets. Although decision sets are unordered (in contrast to DLs), this fact does not simplify the computation of PI-explanations. (In what follows, we assume that a DS implements a total classification function, which is not the case in general due to the issue of overlap [35] — otherwise, PI-explanations would be ill-defined.)

Proposition 4. *Finding an AXp for a DS is hard for* $\mathrm{D^P}$.

Proof (sketch). It is known [35] that decision sets can be associated with DNF formulas. It is also known [70] that finding a prime implicant (PI) of a DNF D given a satisfying assignment $\mathbf{v}$ is complete for $\mathrm{D^P}$. Given the aforementioned connection between DSs and DNFs, we show here that the above problem can be reduced to finding a PI-explanation of a DS.

Let the terms in the DNF D become the rules for prediction $\oplus$ in the corresponding DS. Also, let the default rule of the DS predict $\ominus$. Hence, a set of literals ρ (contained in the literals induced by $\mathbf{v}$) is a PI of the DNF D iff ρ is a PI-explanation for the DS prediction $\oplus$ given $\mathbf{v}$. □

Remark 1. In the case of decision sets, it is also simple to observe that deciding whether a set of literals ρ is an AXp is in $\mathrm{D^P}$. For that, one needs to prove first that the set of literals ρ entails prediction $\oplus$; this problem is clearly in coNP. Additionally, one also needs to prove subset-minimality of ρ, i.e. that removing any single literal from ρ results in a subset of literals that does not entail the prediction $\oplus$. (We can consider $|\rho|$ sets of literals, each of which removes a literal from ρ to get a set of literals ρ_k, and check that there are $|\rho|$ assignments such for each ρ_k we get a different prediction $\ominus$.) The latter problem is in NP. Therefore and given Proposition 4, we can establish $\mathrm{D^P}$-completeness of the decision version of finding a PI-explanation in the case of DSs.

DLs vs. DTs and vs. DSs. The results of this section are somewhat surprising in terms of comparing DTs with DLs and DSs. On the one hand, satisfiability query is trivially in P for DTs and DSs, but it is NP-complete for DLs. On the other hand, AXps can be computed in polynomial time for DTs [37], but a polynomial-time algorithm for computing AXps for DLs and DSs would imply $\mathrm{P} = \mathrm{NP}$.

3.2 Explaining Arbitrary DLs with SAT

When explaining decision lists, one can use the work on computing rigorous abductive [32,66] and contrastive explanations [31] for ML models. This section describes a novel propositional encoding for DL classifiers that can be exploited by the generic approach of [31,32].

Let $\mathbf{v}$ denote a point in feature space with prediction $c \in \mathcal{K}$. Moreover, let the rule that fires on $\mathbf{v}$ be $i \in \mathfrak{R}$. Note that for an arbitrary rule $k \in \mathfrak{R}$ to fire, the following constraint must hold true:

$$\bigwedge_{\substack{r_j \in \mathfrak{R} \\ \mathfrak{o}(j) < \mathfrak{o}(k)}} \neg(\mathfrak{l}(j)) \land \mathfrak{l}(k) \tag{5}$$

Constraint (5) encodes the fact that the literals in all the rules preceding rule k must not fire and the rule k must fire. (Recall that $\mathfrak{l}(i)$ represents the set of literals of rule i). This constraint is straightforward to clausify, i.e. convert to CNF. Moreover, let $\varphi(i)$ denote the set of clauses resulting from clausification of the constraint (5) for rule i to fire.

Given a set of literals ρ, ρ is an implicant of the decision function associated with the DL (i.e. ρ is an AXp) for the instance $\mathbf{v}$ and the corresponding prediction $\mathfrak{c}(i)$ if:

$$\rho \models \bigvee_{\substack{j \in \mathfrak{R} \\ \mathfrak{c}(j) = \mathfrak{c}(i)}} \varphi(j) \tag{6}$$

i.e. for any point $\mathbf{x}$ in feature space, if $\rho(\mathbf{x})$ holds true, then one of the rules predicting the same class $\mathfrak{c}(i)$ as rule i must hold true as well. Constraints (5) and (6) comprise the propositional encoding that can be used in the framework of [32] to compute one AXp for the prediction made by a decision list for a given input instance. Note that computing such an AXp ρ is typically done by reducing the initial set of literals $I_{\mathbf{v}}$, which clearly entails the right-hand side of (6), i.e. $I_{\mathbf{v}} \models \bigvee_{j \in \mathfrak{R}, \mathfrak{c}(j) = \mathfrak{c}(i)} \varphi(j)$. Also note that in practice it is convenient to negate this tautology and instead deal with its negation, which is obviously unsatisfiable. Following [31,32], this enables one to apply the well-developed apparatus for computing one AXp (resp. CXp) as an MUS (resp. MCS) of the negated formula [6,7,26,36,38,42,45,48,50,52,53], but also for enumerating a given number of all AXps (resp. CXps) through MUS (resp. MCS) enumeration [41,43,44,55,60].

Example 7. As mentioned above, when computing an AXp in the form of (6), it is convenient to negate the tautology $I_{\mathbf{v}} \models \bigvee_{j \in \mathfrak{R}, \mathfrak{c}(j) = \mathfrak{c}(i)} \varphi(j)$ and instead work with unsatisfiable formula

$$I_{\mathbf{v}} \land \bigwedge_{\substack{j \in \mathfrak{R} \\ \mathfrak{c}(j) = \mathfrak{c}(i)}} \neg\varphi(j)$$

Here, the left part $I_{\mathbf{v}}$ of the conjunction serves as the set $\mathcal{S}$ of unit-size soft clauses, each represented by a literal assigning a value to a feature. This way AXps and CXps can be found as minimal subsets of $\mathcal{S}$ (i.e. MUSes or MCSes, respectively), subject to the hard clauses $\mathcal{H} \triangleq \bigwedge_{j \in \mathfrak{R}, \mathfrak{c}(j) = \mathfrak{c}(i)} \neg\varphi(j)$. Also observe that the negation $\neg\varphi(k)$ (recall that $\varphi(k)$ enforces rule k to fire) constitutes the disjunction

$$\neg\mathfrak{l}(k) \lor \bigvee_{\substack{r_j \in \mathfrak{R} \\ \mathfrak{o}(j) < \mathfrak{o}(k)}} \mathfrak{l}(j)$$

which enforces that either rule k does not fire or one of the preceding rules fires. Also, to enforce that the default rule does not fire, we can simply require one of the non-default rules of the DL to fire. Finally, note that the hard clauses $\mathcal{H}$ encode the fact of *misclassification*, which is clearly impossible when the input instance $I_{\mathbf{v}}$ is given as the soft clauses $\mathcal{S}$, thus making formula $\mathcal{H} \land \mathcal{S}$ unsatisfiable.

Now, consider the DL from Example 6 and recall that rule R_5 fires prediction $f = 1$ for the instance $\mathbf{v} = (1, 0, 1, 1)$. As prediction $f = 1$ is represented by rules R_2, R_5, R_6, R_{DEF}, our hard clauses $\mathcal{H}$ must enforce that none of them fires. Given the above, the hard clauses $\mathcal{H}$ are formed by

$$\mathcal{H} = \left\{ \begin{array}{ll} \neg\varphi(2) \triangleq \left[\neg\mathfrak{l}(2) \lor \bigvee_{j=0}^{1} \mathfrak{l}(j)\right]; & \neg\varphi(5) \triangleq \left[\neg\mathfrak{l}(5) \lor \bigvee_{j=0}^{4} \mathfrak{l}(j)\right]; \\ \neg\varphi(6) \triangleq \left[\neg\mathfrak{l}(6) \lor \bigvee_{j=0}^{5} \mathfrak{l}(j)\right]; & \neg\varphi_{\text{DEF}} \triangleq \left[\bigvee_{j=0}^{6} \mathfrak{l}(j)\right] \end{array} \right\}$$

Here, CNF encoding of terms $\mathfrak{l}(j)$ is omitted as it is trivial to obtain. □

As can be observed in Example 7, the propositional encoding described in this section targets simplicity and for this reason it exhibits redundancy, e.g. expressions $\bigvee_{j=0}^{k-1} \mathfrak{l}(j)$ in the representation of $\neg\varphi(k)$ are repeated for every $k' > k$. As shown in Section 4, the performance results suggest that the proposed encoding scales well on DLs of realistic size. Nevertheless, a number of improvements can be envisioned, which add more structure to the encoding, but with the cost of using additional auxiliary variables. Our initial experiments suggest no significant gains were obtained with a more complex propositional encoding.

4 Experimental Results

This section aims at assessing the proposed SAT-based approach to computing and enumerating rigorous abductive explanations (AXps) [32,66] as well as contrastive explanations (CXps) [31] for decision list models. First, the approach will be tested from the perspective of raw performance, followed by additional information on the comparative number of AXps and CXps as well as their length.

Experimental Setup. The experiments were performed on a MacBook Pro laptop running macOS Big Sur 11.2.3. Therefore, each individual process was run on a Quad-Core Intel Core i5-8259U 2.30 GHz processor with 16 GByte of memory. The memory limit was set to 4 GByte while the time limit used was set to 1800 seconds, for each individual process to run.

Prototype Implementation. A prototype implementation [4] of the proposed approach was developed as a Python script instrumenting incremental calls to the Glucose 3 SAT solver [4] using the PySAT toolkit [28]. The implementation targets the computation of one explanation (either an AXp or a CXp) and enumeration of a given number of those, with a possibility to enumerate all.

[4] The prototype is available at `https://github.com/alexeyignatiev/xdl-tool`.

It is known [31] that a CXp can be computed as an MCS for the encoding formula discussed above and hence CXp enumeration is implemented in the prototype as LBX-based MCS enumeration [53]. Similarly, AXp corresponds to an MUS of the formula and, as a result, AXp enumeration is done using the MARCO-like MUS enumeration approach [41,43,60] due to the hitting set duality between AXps and CXps [31]. Concretely, the MARCO-like explainer is organized as two interconnecting oracles: (i) a SAT oracle checking (un)satisfiability of a selected set of clauses of the formula, and (ii) a minimal hitting set (MHS) oracle, which computes minimal hitting sets of a current collection of MCSes of the formula obtained so far. The MHS oracle was implemented on top of the RC2 MaxSAT solver exploited incrementally [29]. Each iteration of the MARCO-like explainer computes either an AXp or a CXp. The former are reported and blocked (by adding a single clause to the MHS oracle) while the latter are used later as the sets *to hit*. The explainer stops as soon as there are no more minimal hitting set identified by the MHS oracle. As a result, the MARCO-like explainer produces both AXps and CXps upon the end of execution. Note that thanks to the use of MaxSAT-based MHS oracle, AXps computed this way are irredundant, i.e. subset-minimal, and do not have to be reduced further while CXps do need to be reduced by a dedicated reduction procedure (see below). Also note that the MARCO-like approach can also be used in a *dual way*, i.e. targeting CXp enumeration and computing AXps as a by-product. This mode of operation of the explainer has also been implemented in the developed prototype.

It is also important to mention that all the three modes of operation make incremental use of the underlying SAT oracles. As such, the LBX-like CXp enumeration computes an explanation, blocks it by adding a single clause and proceeds to the next CXp. Furthermore, once all explanations for a given data instance are enumerated, all the previously added blocking clauses are *disabled* and the enumeration process starts again for a new data instance. This is done with the use of unique *selector* variables introduced for each data instance. On the contrary, the MARCO-like approaches accumulate and block all explanations on the MHS oracle side. This enables one to keep the same SAT oracle on the checking side of the approach while restarting the MHS oracle from scratch, i.e. with an empty collection of sets to hit, for each new data instance.

Finally, the following heuristics are used. LBX-like computation of a single CXp makes use of the *Clause D* (CLD) heuristic [50]. Computation of a single AXp is done as a simple deletion-based linear search procedure [51], strengthened by exhaustive enumeration of unit-size MCSes used to bootstrap the MHS oracle. Although a more sophisticated algorithm QuickXPlain [38] has been also implemented, it turned out to be outperformed by the aforementioned simpler alternative in this concrete setting.

Benchmarks and Methodology. Experimental evaluation was performed on a subset of datasets selected from a few publicly available sources. In particular, these include datasets from UCI Machine Learning Repository [69] and Penn Machine Learning Benchmarks [58] as well as datasets previously studied in the context of ML explainability [63] and fairness [23, 24]. The number of selected

datasets is 72. We applied the approach of 5-fold cross validation, i.e. each dataset was randomly split into 5 chunks of instances; each of these chunks served as test data while the remaining 4 chunks were used to train the classifiers. As a result, each dataset (out of 72) resulted in 5 individual pairs of training and test datasets represented by 80% and 20% of data instances. Therefore, the total number of training datasets considered in the evaluation is 360.

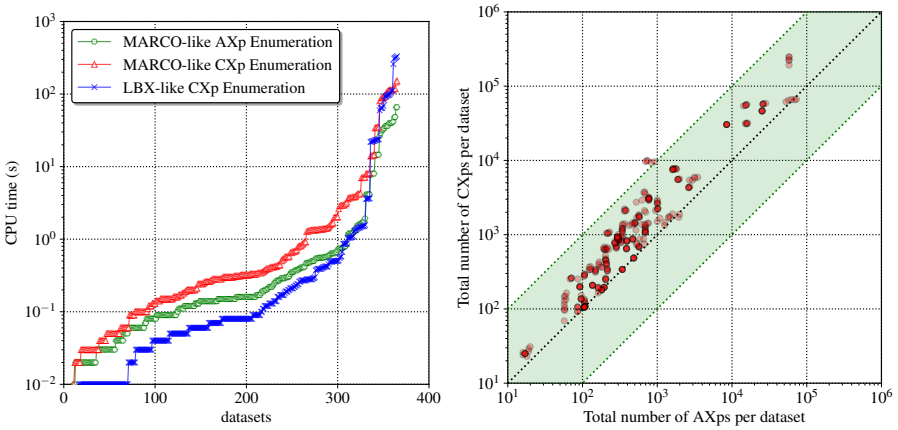
Given a training dataset, i.e. represented by 4 chunks of the original data, a decision list model was trained with the use of the well-known heuristic algorithm CN2 [14, 15][5], the implementation of which was taken from the well-known Python toolkit Orange[6]. The time spent on training the models was ignored. Next, the prototype explainer was run in one of the three modes described above, to enumerate *all* explanations (either AXps, or CXps) for each of the instances of the original 100% data. Also and as mentioned above, the explainer was given 1800 seconds for each of the 360 datasets/models.

Note that the number of rules in the decision list models constructed by CN2 for the target datasets varied from 6 to 2055. Also, the total number of non-class, i.e. solely antecedent, literals used in the models varied from 6 to 6754. Finally, propositional formulas encoding the explanation problems for these models had from 7 to 15340 variables and from 9 to 3932987 clauses. It is important to mention that all data was treated as categorical and hence the propositional formulas given to the encoder incorporated cardinality constraints enforcing that a feature can take exactly one value; in the experiments, these constraints were encoded into CNF using the pairwise encoding [59]. Although left untested, other cardinality encodings would result in smaller formulas — the pairwise encoding was selected intentionally in order to produce larger formulas and so to test scalability of the proposed SAT-based approach.

Raw Performance. Figure 2a depicts a cactus plot showing the raw performance of the explainer working in the three selected modes of operation. (Note that the CPU time axis is scaled logarithmically.) As can be observed, all the algorithms are able to finish successful computation of all the target explanations for all the data instances of each of the 360 benchmark datasets within the given time limit. Surprisingly, the best performing configuration overall turns out to be MARCO-based AXp enumeration. MARCO-based CXp enumeration is a bit slower. Recall that both MARCO-based modes end up enumerating the same sets of explanations including AXps (CXps, resp.) and dual CXps (dual AXps, resp.). Also, recall that the only major difference between the two configurations is the type of the target explanations that are provided by the MHS oracle while the dual explanations have to be reduced by a dedicated reduction procedure. Therefore, the performance difference shown suggests that in practice it may be

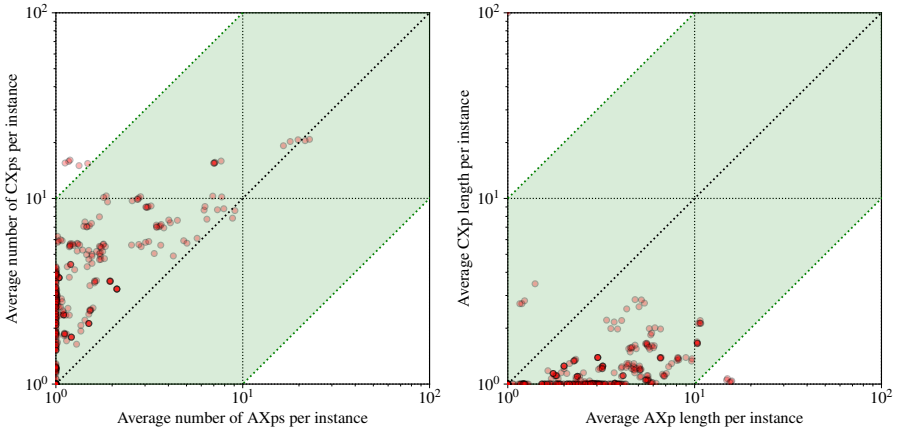
[5] Recent alternative approaches to *sparse* decision lists [1, 2, 65] have also been considered but were eventually discarded for two reasons: (1) they can only deal with binary data and (2) they produce sparse decision lists containing a couple of rules and a few literals in total — i.e. these methods do not provide models that would be of interest for our work.

[6] https://orangedatamining.com/



(a) Raw performance comparison (b) Total number AXps and CXps of per dataset

Fig. 2: Performance of the three operation modes and the total number of explanations per dataset they enumerate.



(a) Average number of explanations per instance (b) Average explanation size per instance

Fig. 3: Average number of AXps and CXps per data instance and their average size.

more beneficial to target AXps and so to reduce dual CXps than doing the opposite (which is not really surprising given that the former correspond to MUS extraction while the latter correspond to MCS extraction). Finally, it should be mentioned that although LBX-like CXp explanation works the most efficiently for most of the benchmarks, in some cases it is outperformed by the competitors, which may be explained by the need to incrementally block a significant number of previously computed solutions (recall that, on the contrary, the MARCO-like configurations restart the MHS oracle from scratch for every new data instance).

AXps vs CXps. As can be seen in Figure 2b, the total number of AXps per dataset tends to be lower than the total number of CXps. Concretely, the number of AXps per dataset varies from 16 to 72838 while the number of CXps per dataset varies from 23 to 248825. (Observe that the time to compute one explanation is negligible.) These data are in line with the results previously obtained in [31] when explaining a different kind of ML model (namely, XGBoost models [13]) with a different reasoning engine (namely, Z3 SMT solver [56]). Unsurprisingly, the average number of CXps per data instance is also higher than the average number of AXps, as shown in Figure 3a. In general, the average number of CXps per instance varies from 1 to 20.8 while the average number of AXps goes from 1 to 22.7. However and as one can observe in the scatter plot Figure 3a, for the lion's share of data instances there is a single AXp while there are many more CXps. Note that the picture is the opposite for the average explanation length (measured as the number of literals remaining in the explanation). In particular, CXps are shorter than AXps and the average length of a CXp per data instance does not exceed 2.8 while the average length of AXp varies from 1 to 15.8 (which in fact may provide another insight into the underperforming MARCO-like CXp enumeration). Observe that these data also confirms the results previously reported in [31].

Final Remarks. A few conclusions can be made with respect to the experimental results shown above. First, all the explainer configurations scale well and are able to enumerate all explanations for all data instances incrementally, even for DL models with thousands of rules and literals encoded into CNF formulas with millions of clauses. Second, MARCO-like AXp enumeration outperforms both LBX-like and MARCO-like CXp enumeration. Third, the number of CXps per dataset and per instance tends to be higher than the number of AXps. And finally, AXps are on average much larger than CXps.

5 Conclusions

This paper investigates the computation of rigorous (or PI-) explanations for DLs. The paper first argues that, similar to DTs [37], DLs may also not be interpretable. (This observation is also validated by the experimental results.) Furthermore, the paper proves that in contrast to the case of DTs, finding one PI-explanation for DLs (and also for DSs) cannot be in P unless P = NP. As a result, one possible solution for finding AXps and CXps is to encode the problem to propositional logic, and find one or enumerate more than one explanation(s) using SAT oracles. The experimental results demonstrate that SAT-based approaches are effective at finding explanations (both AXps and CXps) of DLs. The experimental results also confirm that a MARCO-like algorithm is effective at enumerating explanations of DLs.

The results in this paper suggest a number of future research topics. The application of SAT to explaining DLs motivates the investigation of which other ML models can be explained with SAT solvers, and for which explanations can be computed efficiently.

References

1. Angelino, E., Larus-Stone, N., Alabi, D., Seltzer, M.I., Rudin, C.: Learning certifiably optimal rule lists. In: KDD. pp. 35–44 (2017)
2. Angelino, E., Larus-Stone, N., Alabi, D., Seltzer, M.I., Rudin, C.: Learning certifiably optimal rule lists for categorical data. J. Mach. Learn. Res. **18**, 234:1–234:78 (2017), http://jmlr.org/papers/v18/17-716.html
3. Audemard, G., Koriche, F., Marquis, P.: On tractable XAI queries based on compiled representations. In: KR. pp. 838–849 (2020)
4. Audemard, G., Lagniez, J., Simon, L.: Improving glucose for incremental SAT solving with assumptions: Application to MUS extraction. In: SAT. pp. 309–317 (2013)
5. Bailey, J., Stuckey, P.J.: Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization. In: PADL. pp. 174–186 (2005)
6. Belov, A., Lynce, I., Marques-Silva, J.: Towards efficient MUS extraction. AI Commun. **25**(2), 97–116 (2012)
7. Belov, A., Marques-Silva, J.: Accelerating MUS extraction with recursive model rotation. In: FMCAD. pp. 37–40 (2011)
8. Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.): Frontiers in Artificial Intelligence and Applications, vol. 336. IOS Press (2021)
9. Birnbaum, E., Lozinskii, E.L.: Consistent subsets of inconsistent systems: structure and behaviour. J. Exp. Theor. Artif. Intell. **15**(1), 25–46 (2003)
10. Bouckaert, R.R., Frank, E., Hall, M.A., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: WEKA - experiences with a java open-source project. J. Mach. Learn. Res. **11**, 2533–2541 (2010), http://portal.acm.org/citation.cfm?id=1953016
11. Camburu, O., Giunchiglia, E., Foerster, J., Lukasiewicz, T., Blunsom, P.: Can I trust the explainer? verifying post-hoc explanatory methods. CoRR **abs/1910.02065** (2019), http://arxiv.org/abs/1910.02065
12. Chen, C., Rudin, C.: An optimization approach to learning falling rule lists. In: AISTATS. pp. 604–612 (2018)
13. Chen, T., Guestrin, C.: XGBoost: A scalable tree boosting system. In: KDD. pp. 785–794 (2016)
14. Clark, P., Boswell, R.: Rule induction with CN2: some recent improvements. In: EWSL. pp. 151–163 (1991)
15. Clark, P., Niblett, T.: The CN2 induction algorithm. Machine Learning **3**, 261–283 (1989)
16. Cohen, W.W.: Efficient pruning methods for separate-and-conquer rule learning systems. In: Bajcsy, R. (ed.) Proceedings of the 13th International Joint Conference on Artificial Intelligence. Chambéry, France, August 28 - September 3, 1993. pp. 988–994. Morgan Kaufmann (1993)
17. Cohen, W.W.: Fast effective rule induction. In: ICML. pp. 115–123 (1995)
18. Cohen, W.W., Singer, Y.: A simple, fast, and effictive rule learner. In: AAAI. pp. 335–342 (1999)
19. Darwiche, A., Hirth, A.: On the reasons behind decisions. In: ECAI. pp. 712–720 (2020). https://doi.org/10.3233/FAIA200158
20. Darwiche, A., Marquis, P.: A knowledge compilation map. J. Artif. Intell. Res. **17**, 229–264 (2002)
21. Davies, J., Bacchus, F.: Solving MAXSAT by solving a sequence of simpler SAT instances. In: CP. pp. 225–239 (2011)

22. Demsar, J., Curk, T., Erjavec, A., Gorup, C., Hocevar, T., Milutinovic, M., Mozina, M., Polajnar, M., Toplak, M., Staric, A., Stajdohar, M., Umek, L., Zagar, L., Zbontar, J., Zitnik, M., Zupan, B.: Orange: data mining toolbox in python. J. Mach. Learn. Res. **14**(1), 2349–2353 (2013), http://dl.acm.org/citation.cfm?id=2567736, https://orangedatamining.com/
23. Auditing black-box predictive models. https://blog.fastforwardlabs.com/2017/03/09/fairml-auditing-black-box-predictive-models.html (2016)
24. Friedler, S., Scheidegger, C., Venkatasubramanian, S.: On algorithmic fairness, discrimination and disparate impact (2015)
25. Ignatiev, A.: Towards trustable explainable AI. In: IJCAI. pp. 5154–5158 (2020)
26. Ignatiev, A., Janota, M., Marques-Silva, J.: Quantified maximum satisfiability. Constraints An Int. J. **21**(2), 277–302 (2016)
27. Ignatiev, A., Morgado, A., Marques-Silva, J.: Propositional abduction with implicit hitting sets. In: ECAI. pp. 1327–1335 (2016)
28. Ignatiev, A., Morgado, A., Marques-Silva, J.: PySAT: A Python toolkit for prototyping with SAT oracles. In: SAT. pp. 428–437 (2018)
29. Ignatiev, A., Morgado, A., Marques-Silva, J.: RC2: an efficient MaxSAT solver. J. Satisf. Boolean Model. Comput. **11**(1), 53–64 (2019)
30. Ignatiev, A., Morgado, A., Weissenbacher, G., Marques-Silva, J.: Model-based diagnosis with multiple observations. In: IJCAI. pp. 1108–1115 (2019)
31. Ignatiev, A., Narodytska, N., Asher, N., Marques-Silva, J.: From contrastive to abductive explanations and back again. In: AI*IA (2020), preliminary version available from https://arxiv.org/abs/2012.11067
32. Ignatiev, A., Narodytska, N., Marques-Silva, J.: Abduction-based explanations for machine learning models. In: AAAI. pp. 1511–1519 (2019)
33. Ignatiev, A., Narodytska, N., Marques-Silva, J.: On relating explanations and adversarial examples. In: NeurIPS. pp. 15857–15867 (2019)
34. Ignatiev, A., Narodytska, N., Marques-Silva, J.: On validating, repairing and refining heuristic ML explanations. CoRR **abs/1907.02509** (2019), http://arxiv.org/abs/1907.02509
35. Ignatiev, A., Pereira, F., Narodytska, N., Marques-Silva, J.: A sat-based approach to learn explainable decision sets. In: IJCAR. pp. 627–645 (2018)
36. Ignatiev, A., Previti, A., Liffiton, M.H., Marques-Silva, J.: Smallest MUS extraction with minimal hitting set dualization. In: CP. pp. 173–182 (2015)
37. Izza, Y., Ignatiev, A., Marques-Silva, J.: On explaining decision trees. CoRR **abs/2010.11034** (2020)
38. Junker, U.: QUICKXPLAIN: preferred explanations and relaxations for over-constrained problems. In: AAAI. pp. 167–172 (2004)
39. Lakkaraju, H., Bach, S.H., Leskovec, J.: Interpretable decision sets: A joint framework for description and prediction. In: KDD. pp. 1675–1684 (2016)
40. Lakkaraju, H., Bastani, O.: "how do I fool you?": Manipulating user trust via misleading black box explanations. In: AIES. pp. 79–85 (2020)
41. Liffiton, M.H., Malik, A.: Enumerating infeasibility: Finding multiple MUSes quickly. In: CPAIOR. pp. 160–175 (2013)
42. Liffiton, M.H., Mneimneh, M.N., Lynce, I., Andraus, Z.S., Marques-Silva, J., Sakallah, K.A.: A branch and bound algorithm for extracting smallest minimal unsatisfiable subformulas. Constraints An Int. J. **14**(4), 415–442 (2009)
43. Liffiton, M.H., Previti, A., Malik, A., Marques-Silva, J.: Fast, flexible MUS enumeration. Constraints An Int. J. **21**(2), 223–250 (2016)
44. Liffiton, M.H., Sakallah, K.A.: On finding all minimally unsatisfiable subformulas. In: SAT. pp. 173–186 (2005)

45. Liffiton, M.H., Sakallah, K.A.: Algorithms for computing minimal unsatisfiable subsets of constraints. J. Autom. Reasoning **40**(1), 1–33 (2008)
46. Lipton, Z.C.: The mythos of model interpretability. Commun. ACM **61**(10), 36–43 (2018)
47. Lundberg, S.M., Lee, S.: A unified approach to interpreting model predictions. In: NeurIPS. pp. 4765–4774 (2017)
48. Lynce, I., Marques-Silva, J.: On computing minimum unsatisfiable cores. In: SAT (2004)
49. Marques-Silva, J., Gerspacher, T., Cooper, M.C., Ignatiev, A., Narodytska, N.: Explaining naive bayes and other linear classifiers with polynomial time and delay. In: NeurIPS (2020)
50. Marques-Silva, J., Heras, F., Janota, M., Previti, A., Belov, A.: On computing minimal correction subsets. In: IJCAI. pp. 615–622 (2013)
51. Marques-Silva, J., Lynce, I.: On improving MUS extraction algorithms. In: SAT. pp. 159–173 (2011)
52. Mencia, C., Ignatiev, A., Previti, A., Marques-Silva, J.: MCS extraction with sublinear oracle queries. In: SAT. pp. 342–360 (2016)
53. Mencia, C., Previti, A., Marques-Silva, J.: Literal-based MCS extraction. In: IJCAI. pp. 1973–1979 (2015)
54. Miller, T.: Explanation in artificial intelligence: Insights from the social sciences. Artif. Intell. **267**, 1–38 (2019)
55. Morgado, A., Liffiton, M.H., Marques-Silva, J.: MaxSAT-based MCS enumeration. In: HVC. pp. 86–101 (2012)
56. de Moura, L.M., Bjørner, N.: Z3: an efficient SMT solver. In: TACAS. pp. 337–340 (2008)
57. Narodytska, N., Shrotri, A.A., Meel, K.S., Ignatiev, A., Marques-Silva, J.: Assessing heuristic machine learning explanations with model counting. In: SAT. pp. 267–278 (2019)
58. Penn Machine Learning Benchmarks. https://github.com/EpistasisLab/penn-ml-benchmarks
59. Prestwich, S.D.: CNF encodings. In: Handbook of Satisfiability: Second Edition, Frontiers in Artificial Intelligence and Applications, vol. 336, pp. 75–100. IOS Press (2021)
60. Previti, A., Marques-Silva, J.: Partial MUS enumeration. In: AAAI (2013)
61. Reiter, R.: A theory of diagnosis from first principles. Artif. Intell. **32**(1), 57–95 (1987)
62. Ribeiro, M.T., Singh, S., Guestrin, C.: "why should I trust you?": Explaining the predictions of any classifier. In: KDD. pp. 1135–1144 (2016)
63. Ribeiro, M.T., Singh, S., Guestrin, C.: Anchors: High-precision model-agnostic explanations. In: AAAI. pp. 1527–1535 (2018)
64. Rivest, R.L.: Learning decision lists. Mach. Learn. **2**(3), 229–246 (1987). https://doi.org/10.1007/BF00058680, https://doi.org/10.1007/BF00058680
65. Rudin, C., Ertekin, S.: Learning customized and optimized lists of rules with mathematical programming. Math. Program. Comput. **10**(4), 659–702 (2018). https://doi.org/10.1007/s12532-018-0143-8, https://doi.org/10.1007/s12532-018-0143-8
66. Shih, A., Choi, A., Darwiche, A.: A symbolic approach to explaining bayesian network classifiers. In: IJCAI. pp. 5103–5111 (2018)
67. Shih, A., Choi, A., Darwiche, A.: Compiling bayesian network classifiers into decision graphs. In: AAAI. pp. 7966–7974 (2019)

68. Slack, D., Hilgard, S., Jia, E., Singh, S., Lakkaraju, H.: Fooling LIME and SHAP: adversarial attacks on post hoc explanation methods. In: AIES. pp. 180–186 (2020)
69. UCI Machine Learning Repository. https://archive.ics.uci.edu/ml
70. Umans, C., Villa, T., Sangiovanni-Vincentelli, A.L.: Complexity of two-level logic minimization. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **25**(7), 1230–1246 (2006)
71. Wang, F., Rudin, C.: Falling rule lists. In: AISTATS (2015)
72. Yang, F., Yang, Z., Cohen, W.W.: Differentiable learning of logical rules for knowledge base reasoning. In: NeurIPS. pp. 2319–2328 (2017)
73. Yang, H., Rudin, C., Seltzer, M.I.: Scalable bayesian rule lists. In: ICML. pp. 3921–3930 (2017)