# On Trustworthy Rule-Based Models and Explanations

Mohamed Siala[1] (✉), Jordi Planes[2], and Joao Marques-Silva[3]

[1] LAAS-CNRS, Université de Toulouse, CNRS, INSA Toulouse, France
{msiala}@laas.fr
[2] Universitat de Lleida jordi.planes@udl.cat
[3] ICREA & University of Lleida jpms@icrea.cat

**Abstract.** A task of interest in machine learning (ML) is that of ascribing explanations to the predictions made by ML models. Furthermore, in domains deemed high risk, the rigor of explanations is paramount. Indeed, incorrect explanations can and will mislead human decision makers. As a result, and even if interpretability is acknowledged as an elusive concept, so-called interpretable models are employed ubiquitously in high-risk uses of ML and data mining (DM). This is the case for rule-based ML models, which encompass decision trees, diagrams, sets and lists. This paper relates explanations with well-known undesired facets of rule-based ML models, which include negative overlap and several forms of redundancy. The paper develops algorithms for the analysis of these undesired facets of rule-based systems, and concludes that well-known and widely used tools for learning rule-based ML models will induce rule sets that exhibit one or more negative facets.

**Keywords:** Explainability · Interpretability · Rule-based models · Formal Methods.

## 1 Introduction

Explainable Artificial Intelligence (XAI) is a mainstay of trustworthy AI [1, 20, 7, 13, 42]. Furthermore, in domains that are deemed of high risk, explanations should be trustable [40, 41, 24, 15]. The importance of explanations and the need to trust those explanations motivated work on so-called interpretable models [40, 41], even though it is generally accepted that a rigorous definition of interpretability is elusive at best [29]. Rule-based models, which encompass decision trees [6], diagrams [21, 14], sets [8, 28, 44] and lists [39, 44], epitomize interpretable models.

Work on the induction of rule-based models can be traced at least to the 1970s [43, 25], in the concrete case of decision trees.[4] Decision trees are widely used in practice and often exemplify interpretable models [40, 41]. The perceived

[4] Although extremely popular in ML and DM, decision trees found earlier uses in other domains, e.g. https://en.wikipedia.org/wiki/Phylogenetic_tree and https://en.wikipedia.org/wiki/Decision_tree.

importance of interpretability has recently motivated the development of algorithms for learning optimal decision trees [11, 22]. Decision sets (or rule sets) find a wide range of uses in different domains [17, 18, 36, 23, 35, 2]. As with decision trees, there has been recent interest in learning optimal decision sets [28]. Decision lists also find many practical uses, but claims about their interpretability are harder to justify [31]. As a result, this paper studies decision sets, but also decision trees when viewed as a special case of decision sets.

At present, some of the best-known ML toolkits implement one or more methods of induction of rule-based models [34, 36, 12]. Nevertheless, it has been argued [31] that rule-based methods, although easier to fathom by human-decision makers, still require explanations to be computed. (Otherwise, human decision-makers would be expected to manually solve NP-hard function problems [31].) Therefore, a key question is: *for rule-based models, when can explanations be computed trivially, such that a human decision-maker can manually produce an explanation?*

This paper shows that rigorous explanations can be found manually whenever some undesired facets of decision sets are nonexistent. Concretely, the paper relates easy-to-compute explanations with the non-existence of *negative overlap*, i.e. the existence of cases where two or more rules can fire that predict different values. Furthermore, the non-existence of redundant literals in rules is shown to be a necessary condition for minimality of explanations.

Given this state of affairs, the paper then investigates whether existing ML toolkits are able to learn rule-based models that avoid the aforementioned negative facets. As the results demonstrate, this is not the case. In addition, the paper investigates whether model-agnostic methods targeting feature selection (i.e. that produce rules as explanations) are capable of preventing negative overlap (i.e. the most worrisome negative facet). Unfortunately, as the results show, this is also not the case with the well-known explainer Anchor [38].

**Contributions.** The paper studies decision sets,[5] concretely the problem of *negative overlap*, i.e. when two rules that predict different classes fire, but also the existence of local or global redundancies of literals in rules. The paper develops algorithms for deciding the existence of negative overlap, but also for deciding local and global redundancy. Furthermore, the results in the paper take into account possible constraints on the inputs. The paper then relates these negative facets of decision sets with the ability of human decision-makers to manually produce rigorous explanations, namely abductive explanations. In addition, the experiments confirm that implemented rule-learning algorithms in well-known toolkits exhibit the negative facets of decision sets, thus complicating (complexity-wise) the computation of rigorous explanations.

**Organization.** The paper is organized as follows. Section 2 introduces the notation and definitions used throughout the paper. Section 3 briefly comments on related work. Section 4 details the paper's main contributions. Section 5 reports on the experimental results. Finally, Section 6 concludes the paper.

[5] Decision trees are a special case of a decision set, and so we also present experiments on decision trees. However, we opt not to address decision lists due to the intrinsic difficulties with their explanation [31].

## 2 Background

The notation and definitions used throughout the paper are adapted from past works [28, 4, 27].

**Propositional Logic and Generalizations [4].** Let $X = \{x_1, \ldots, x_n\}$ be a set of Boolean variables. A literal is a Boolean variable or its negation. A clause $C$ is a disjunction of literals and a cube $L$ is a conjunction of literals. We use the notation $l_i \in C$ (respectively $l_i \in L$) if $C = l_1 \vee \ldots \vee l_k$ (respectively $L = l_1 \wedge \ldots \wedge l_k$). A conjunctive normal form (CNF) formula $F$ is a conjunction of clauses. That is, $F = C_1 \wedge \ldots \wedge C_k$ where $C_j$ is a clause. In this case, we use the notation $C_j \in F$. Note by definition that a clause/cube is a CNF. An assignment $v = (v_1, \ldots v_n)$ is a point in $\{0,1\}^n$. If $F = C_1 \wedge \ldots \wedge C_k$ is a CNF, $v \models F$ iff $\forall C_j \in F, \exists x_i \in C_j$ such that $v_i = 1$ or $\exists\, \neg x_i \in C_j$ such that $v_i = 0$. If $\exists\, v \in \{0,1\}^n$ such that $v \models F$ then $F$ is said satisfiable, otherwise unsatisfiable. If $F_1$ and $F_2$ are two CNF formulas, $F_1 \models F_2$ iff $v \models F_1 \implies v \models F_2$. Note that $F_1 \models F_2$ iff $F_1 \wedge \neg F_2$ is unsatisfiable. Given a CNF formula $F$, the satisfiability problem (SAT) asks if $F$ is satisfiable. SAT solvers are highly deployed in practice to answer SAT related queries, such as finding satisfying assignments or proving unsatisfiability [4]. Furthermore, extensions of propositional to more expressive logics can be handled by considering Satisfiability Modulo Theories (SMT) [4].

**Machine Learning.** We consider rule-based models for classification and regression that can be represented as a set of unordered rules. Let $\mathcal{F} = \{1, \ldots m\}$ be a set of features where each feature $i$ takes values from a domain $D_i$. The feature space is the Cartesian product of the domains $\mathbb{F} = D_i \times \ldots \times D_m$. The outcome space (i.e., classes for classification and numerical values for regression) is denoted by $\mathcal{V}$. A dataset is a set $\{(x, o) \mid x \in \mathbb{F} \wedge o \in \mathcal{V}\}$, and where $x = (x_1, \ldots, x_m)$. A literal represents a condition on the values of a feature. We use $\mathbb{L}$ to represent the universe of literals. A background knowledge $\mathcal{B}$ is a propositional formula over literals from $\mathbb{L}$ that specifies the conditions that any arbitrary point in feature space must comply with. In other words, a point in feature space $x$ is *valid* iff $x \models \mathcal{B}$. We assume in the rest of the paper that $\mathcal{B}$ is given as a CNF. For example, consider a dataset representing individuals and the two literals $l_1 := \mathit{employed}$, $l_2 := \mathit{salary} > 50k$. The background knowledge $\mathcal{B}$ can contain the clause $l_1 \vee \neg l_2$ to model the fact that an unemployed individual cannot have a salary greater than $50k$. Note that $\mathcal{B}$ can be a tautology (for instance when no condition is given). In this case, any arbitrary point in feature space is a valid. A user can also miss certain constraints she is not aware of. Let $\lambda \notin \mathcal{V}$ be a dummy value. A supervised ML (classification or regression) model $\kappa$ is a mapping from $\mathbb{F}$ to $\{\lambda\} \cup \mathcal{V}$ such that $\kappa(x) = \lambda$ iff $x \not\models \mathcal{B}$.

A rule $R_i$ is a pair $(L_i, o_i)$ such that $L_i$ is a conjunction of literals (i.e., cube) from $\mathcal{L} \subseteq \mathbb{L}$ and $o_i \in \mathcal{V}$. $R_i$ fires on $x \in \mathbb{F}$ iff $x \models L_i$. With a slight abuse of notation we shall sometimes use $L_i$ as the subset of $\mathcal{L}$ formed by the literals in $L_i$. A decision set $\mathcal{M}$ is a set of rules $\mathcal{M} = \{R_1, \ldots, R_r, R_{r+1}\}$ such that $\forall i \leq r, L_i \neq \emptyset$ and $L_{r+1} = \emptyset$. $R_{r+1}$ is called the default rule. We denote $\Delta(o)$

the set $\{R_i | o_i = o\}$. $\mathcal{M}$ is used as an ML model $\kappa_{\mathcal{M}}$ as follows:

$$\kappa_{\mathcal{M}}(x) = \begin{cases} \lambda \notin \mathcal{V} & \text{if } x \not\models \mathcal{B} \\ o_{r+1} & \text{if no rule fires on } x \\ o & \text{if } \{o\} = \{o_i \mid R_i \text{ fires on } x\} \\ \text{Tie-breaking strategy} & \text{otherwise} \end{cases}$$

Note that decision trees (DTs), decision diagrams (DDs), random forests (RFs) and boosted trees (BTs), can be seen as decision sets where each path represents a rule. Clearly, in such models the default rule never fires. In the case of DTs and DDs, each input fires exactly one rule (since it follows exactly one path). Thus, no tie-breaking strategy is needed. This is not the case with RFs and BTs since each input fires one rule on each tree. Therefore, a tie-breaking strategy is needed.

We extend the notion of cover and overlap from [28] by considering the background knowledge $\mathcal{B}$ and the input space.

**Definition 1 (Cover).** *Given $X \subseteq \mathbb{F}$ and background knowledge $\mathcal{B}$, $Cover(X, B, L) = \{x \mid x \in X \land x \models \mathcal{B} \land x \models L\}$.*

**Definition 2 (Overlap).** *Given a background knowledge $\mathcal{B}$, two rules $R_i$ and $R_j$ such that $i, j \leq r$ overlap in $X \subseteq \mathbb{F}$ iff $Cover(X, B, L_i) \cap Cover(X, B, L_j) \neq \emptyset$.*

We say that $R_i$ and $R_j$ positively (respectively negatively) overlap if they overlap and $o_i = o_j$ (respectively $o_i \neq o_j$). We use the notation $R_i \ominus R_j$ if $R_i$ and $R_j$ negatively overlap. Observe that DTs and DDs exhibit no overlap since each input is captured by exactly one rule. This is not the case for RFs and BTs, since each input fires exactly one rule from each tree. Thus, overlaps may occur only between rules from different trees.

**Formal Explanations [27, 10].** Most approaches to explainability target at instance, i.e. a pair $(x, c)$ with $x \in \mathbb{F}$ and $c \in \mathcal{V}$. We use $\kappa$ throughout the paper to denote a machine learning model. Given an instance $(v, c)$, with $c = \kappa(v)$, a weak abductive explanation (WAXp) is a subset $\mathcal{X}$ of the features $\mathcal{F}$ which, if assigned the values dictated by $v$, is sufficient for the classifier to output prediction $c = \kappa(v)$ [27, 10]:

$$\forall (x \in \mathbb{F}). \left[ \bigwedge\nolimits_{i \in \mathcal{X}} (x_i = v_i) \rightarrow (\kappa(x) = c) \right] \tag{1}$$

A subset-minimal WAXp is an *abductive explanation* (AXp). Recent work demonstrated the need for explaining interpretable models, including decision trees [27] and lists [31]. To the best of our knowledge, past work did not investigate formal explanations for decision sets.

Furthermore, the definition of WAxp (see (1)) can be generalized to account for literals involving other relational operators [27] (e.g. relational operators taken from $\{\in, \geq, >, <, \leq\}$). In addition, constraints on the inputs [19, 3] can be accounted for by conjoining a set of constraints $\mathcal{C}_{\mathcal{B}}$. For example, these constraints allow capturing the background knowledge introduced earlier in this

section. Concretely, we write that $\mathcal{C}_{\mathcal{B}}(x)$ holds true iff $x$ respects the background knowledge, i.e. $x \models \mathcal{B}$.

## 3 Related Work

The learning of rule-based models has been the subject of research since the 1970s [43, 25]. The importance of the topic, especially given their widely accepted interpretability, has motivated recent work on learning decision sets [36, 35, 2] and (optimal) trees [11]. These earlier works were motivated by the accepted belief that decision trees, sets and lists are interpretable [5, 40, 41]. Accounts of methods for learning decision sets and lists include [17, 18].

Motivated by the elusive nature of interpretability's definition [29], recent work [31] uncovered practical difficulties in computing and/or using so-called interpretable models as explanations. For example, it has been shown that paths in decision trees can be arbitrarily redundant (on the number of features) when compared with an AXp [27]. Similarly, the computation of an AXp for a decision list equates with solving an NP-hard problem [31], i.e. something that is in general beyond the capabilities of a human decision-maker. Nevertheless, past work did not address formal explanations for decision sets, arguably because of the existence of negative overlap.

Although the paper assesses rule-based methods using formal explanations, XAI is better-known by the use of model-agnostic methods [1, 20, 7, 32, 13, 42]. Well-known examples include LIME [37], SHAP [30] and Anchors [38]. Since so-called interpretable models have been proposed for high-risk uses of ML, we focus on rigorous (i.e. formal) explanations.

The main results of this paper, namely the direct relationship between easy-to-compute explanations and the non-existence of well-known negative facets of rule-based models, are novel. The observation that rule-based models, obtained with well-known toolkits, exhibit those negative facets, is also a novel result, to the best of our knowledge.

## 4 Overlap and Redundancy

In this section, we let $\mathcal{B}$ be a background knowledge and $\mathcal{M} = \{R_1, \ldots, R_r, R_{r+1}\}$ be a decision set where $R_{r+1}$ is the default rule such that each rule $R_{i \leq r}$ fires on at least one valid input (w.r.t. $\mathcal{B}$). As mentioned in the introduction, we provide a formal framework to address the following questions: (i) How can we generate all (negative) overlap?; (ii) Is rule $R_i$ redundant in $\mathcal{M}$?; and (iii) Is literal $l$ redundant in a given rule?.

We use Example 1 throughout the paper to illustrate the different concepts.

*Example 1.* $\mathcal{B}$ is background knowledge that encodes the following constraints (in a CNF): $(salary > 0) \leftrightarrow (age \geq 18)$; $(size = 140) \rightarrow (size > 120)$; $(weight > 90) \rightarrow (weight \geq 85)$; and $(weight \geq 85) \rightarrow (weight > 80)$. The decision set contains the following rules:

- $R_1 = ((salary > 0) \land (size \neq 140) \land (age > 10) \land (color = blue) \land (weight > 80),\ 1)$

**Algorithm 1** Negative Overlap Pairs

1: **Function:** *Pairs*
2: **Input:** $\mathbb{F}, O, \mathcal{M} = \{R_1, \dots R_r\}, \mathcal{B}$
3: **Output:** $\Pi = \{(i,j) \mid R_i \ominus R_j\}$
4: $\Pi = \emptyset$
5: $\Psi = GetList(o_1, o_2, \dots o_r)$
6: $g = |\Psi|$
7: **for** $a$ in $1, \dots g-1$ **do**
8: **for** $b$ in $a+1, \dots g$ **do**
9: **for** $R_i$ in $\Delta(\Psi(a))$ **do**
10: **for** $R_j$ in $\Delta(\Psi(b))$ **do**
11: **if** $\mathcal{B} \wedge L_i \wedge L_j$ is SATISFIABLE **then**
12: $\Pi \leftarrow \Pi \cup \{(i,j)\}$
13: **end if**
14: **end for**
15: **end for**
16: **end for**
17: **end for**
18: **Return** $\Pi$

- $R_2 = ((salary > 0) \wedge (size = 140),\ 1)$
- $R_3 = ((salary > 0) \wedge (weight > 90),\ 1)$
- $R_4 = ((size > 120) \wedge (weight < 85),\ 0)$

### 4.1 Overlap

We start by giving a sufficient and necessary condition to check if two rules negatively overlap.

**Lemma 1 (Overlap Check).** *Two rules $R_i$ and $R_j$ overlap iff $\mathcal{B} \wedge L_i \wedge L_j$ is satisfiable.*

*Proof.* $\mathcal{B} \wedge L_i \wedge L_j$ is satisfiable iff $\exists x \in \mathbb{F}, x \models \mathcal{B} \wedge L_i$ and $x \models \mathcal{B} \wedge L_j$. This is equivalent to $\exists x \in \mathbb{F}, \{x\} \in Cover(\mathbb{F}, B, L_i) \cap Cover(\mathbb{F}, B, L_j)$. The latter means that $R_i$ and $R_j$ overlap. □

In Example 1, one can use Lemma 1 to show that $R_3$ and $R_4$ do not overlap, in contrast to $R_1$ and $R_4$, which do.

We consider now the question of generating all negative overlap. Algorithm 1 finds all pairs of rules that exhibit a negative overlap. We use $GetList(o_1, o_2, \dots o_r)$ as a function that computes a list that contains the distinct values in $\{o_1, \dots o_r\}$.

Algorithm1 terminates because each pair of rules will be visited at most once. The correctness of Algorithm1 follows from the fact that each pair $(R_i, R_j)$ such that $o_i \neq o_j$ is visited exactly once in Line 12. The complexity of Algorithm 1 is $O(|\mathcal{M}|^2 \times f(\mathcal{M}))$ where $f(M)$ is the worst complexity of $\mathcal{B} \wedge L_i \wedge L_j$ for an arbitrary pair of rules $(R_i, R_j)$. This observation follows from the fact that computing *GetList* can be naturally be done in $O(|\mathcal{M}|)$ and the fact that the satisfiability check in Line 12 is called at most once for each pair $(R_i, R_j)$.

Finally, one might ask whether the default rule can be triggered. Proposition 1 shows that this can be achieved with one SAT call.

**Proposition 1 (Default Rule Application).** *The default rule is triggered iff* $\mathcal{B} \wedge \neg L_1 \ldots \wedge \neg L_r$ *is satisfiable.*

*Proof (Sketch).* No rule fires on a solution to $\mathcal{B} \wedge \neg L_1 \ldots \wedge \neg L_r$. □

### 4.2 Redundancy

In order to study rule and literal redundancy, we provide a formal definition of decision sets equivalence. We denote by $S_{\mathcal{M}}(o) = \cup_{R_i \in \Delta(o)} \{x \in \mathbb{F} \mid x \models \mathcal{B} \wedge L_i\}$.

**Definition 3 (Decision Set Equivalence).** *Let* $\mathcal{M}_1$ *and* $\mathcal{M}_2$ *be two decision sets defined over the same feature space* $\mathbb{F}$ *and output* $\mathcal{V}$ *and having the same default rule.* $\mathcal{M}_1$ *is equivalent to* $\mathcal{M}_2$ *iff* $\forall o \in \mathcal{V}, S_{\mathcal{M}_1}(o) = S_{\mathcal{M}_2}(o)$.

The following lemma is an immediate consequence of Definition 3.

**Lemma 2 (Lemma Decision Set Equivalence).** *Let* $\mathcal{M}_1$ *and* $\mathcal{M}_2$ *be two equivalent decision sets that exhibit no negative overlap and let* $\mathcal{B}$ *be a background knowledge. Then* $\forall x \models \mathcal{B}, \kappa_{\mathcal{M}_1}(x) = \kappa_{\mathcal{M}_2}(x)$.

We introduce the notion of rule redundancy to capture the fact that removing a given rule from a decision set leads to an equivalent decision set.

**Definition 4 (Rule Redundancy).** *A rule* $R_i$ *is redundant in* $\mathcal{M}$ *iff* $\mathcal{M} \setminus R_i$ *is equivalent to* $\mathcal{M}$

Let $G_i = \Delta(o_i) \setminus \{R_i\} = \{R_{i_1}, \ldots, R_{i_z}\}$ where $R_{i_m} = (L_{i_m}, o_{i_m})$.

**Proposition 2 (Rule Redundancy Check).** *A rule* $R_i$ *is redundant in* $\mathcal{M}$ *iff* $\mathcal{B} \wedge L_i \models L_{i_1} \vee \ldots \vee L_{i_z}$.

*Proof.* Let $\mathcal{M}* = \mathcal{M} \setminus R_i$. Clearly $R_i$ is redundant in $\mathcal{M}$ iff $S_{\mathcal{M}}(o_i) = S_{\mathcal{M}^*}(o_i)$. In other words, iff $\cup_{R_j \in \Delta(o_i)} \{x \in \mathbb{F} \mid x \models \mathcal{B} \wedge L_j\} = \cup_{R_j \in \Delta(o_i) \setminus R_i} \{x \in \mathbb{F} \mid x \models \mathcal{B} \wedge L_j\}$. The latter is true iff $\mathcal{B} \wedge L_i \models L_{i_1} \vee \ldots \vee L_{i_z}$. □

Following Proposition 2, one can check if a rule is redundant with one SAT oracle since $\mathcal{B} \wedge L_i \models L_{i_1} \vee \ldots \vee L_{i_z}$ iff $\mathcal{B} \wedge L_i \wedge \neg L_{i_1} \wedge \ldots \wedge \neg L_{i_z}$ is unsatisfiable. For instance, in Example 1, this allows to show that $R_3$ is redundant.

One can also build an equivalent decision set with no redundant rules by checking and removing redundant rules iteratively. Note that the order in which the redundant rules are removed matters as it might return a different decision set at each execution.

We assume in the rest of this section that no rule is redundant. Suppose that $L_i$ contains at least two literals and that $l \in L_i$. We denote by $\mathcal{M}_l^i = \mathcal{M} \cup (L_i \setminus l, o_i) \setminus R_i$ the decision set identical to $\mathcal{M}$ except that $l$ is removed from $L_i$. We give a formal definition of literal redundancy.

**Definition 5 (Literal Redundancy).** *A literal* $l$ *is redundant in* $L_i$ *iff* $l \in L_i$ *and* $\mathcal{M}_l^i$ *is equivalent to* $\mathcal{M}$.

Informally speaking, a literal is redundant in $L_i$ iff its removal from $L_i$ leads to an equivalent decision set. In the following we prove that there are only two cases of redundancies that we call local and global redundancies, and we show sufficient and necessary conditions to find (and remove) them. When using $L_i$, we suppose that it contains at least two literals.

We denote by $L_i^{\bar{l}} = L_i \cup \{\neg l\} \setminus \{l\}$. We define the following sets to address literal redundancy: $\Omega_i = \cup_{R_j \in G_i} \{x \in \mathbb{F} \mid x \models \mathcal{B} \wedge L_j\}$, $\Theta_i = \{x \in \mathbb{F} \mid x \models \mathcal{B} \wedge L_i \setminus \{l\}\}$, $\Xi_i = \{x \in \mathbb{F} \mid x \models \mathcal{B} \wedge L_i\}$, $\Upsilon_i = \{x \in \mathbb{F} \mid x \models \mathcal{B} \wedge L_i^{\bar{l}}\}$. By construction, we have:

- $\Theta_i = \Xi_i \cup \Upsilon_i$
- $S_{\mathcal{M}}(o_i) = \Omega_i \cup \Xi_i$
- $S_{\mathcal{M}_l^i}(o_i) = \Omega_i \cup \Theta_i = \Omega_i \cup \Xi_i \cup \Upsilon_i$

**Proposition 3 (Literal Redundancy (1)).** *A literal $l$ is redundant in $L_i$ iff $l \in L_i$ and $\Omega_i \cup \Xi_i = \Omega_i \cup \Theta_i = \Omega_i \cup \Xi_i \cup \Upsilon_i$*

*Proof.* Observe first that $\mathcal{M}_l^i$ is equivalent to $\mathcal{M}$ iff $S_{\mathcal{M}}(o_i) = S_{\mathcal{M}_l^i}(o_i)$. Therefore, $l$ is redundant in $L_i$ iff $\Omega_i \cup \Xi_i = \Omega_i \cup \Theta_i = \Omega_i \cup \Xi_i \cup \Upsilon_i$. □

**Lemma 3 (Local Redundancy).** *If $l \in L_i$ and $\mathcal{B} \wedge L_i \setminus \{l\} \models l$ then $l$ is redundant in $L_i$. This is called local redundancy.*

*Proof.* If $\mathcal{B} \wedge L_i \setminus \{l\} \models l$ then $\Xi_i = \Theta_i$ and thus $S_{\mathcal{M}_l^i}(o_i) = \Omega_i \cup \Theta_i = \Omega_i \cup \Xi_i = S_{\mathcal{M}}(o_i)$. Therefore, by Proposition 3, $l$ is redundant in $L_i$. □

In Example 1, $(age > 10)$ is locally redundant in $R_1$.

Recall that $G_i = \Delta(o_i) \setminus \{R_i\} = \{R_{i_1}, \ldots, R_{i_z}\}$ and $L_i^{\bar{l}} = L_i \cup \{\neg l\} \setminus \{l\}$.

**Lemma 4 (Global Redundancy).** *If $l$ is not locally redundant in $L_i$ and $\mathcal{B} \wedge L_i^{\bar{l}} \models L_{i_1} \vee \ldots, \vee L_{i_z}$, then $l$ is redundant in $L_i$. This is called global redundancy.*

*Proof.* If $\mathcal{B} \wedge L_i^{\bar{l}} \models L_{i_1} \vee \ldots, \vee L_{i_z}$ then $\Upsilon_i \subseteq \Omega_i$. Thus, since $\Theta_i = \Xi_i \cup \Upsilon_i$, we have $S_{\mathcal{M}_l^i}(o_i) = \Omega_i \cup \Theta_i = \Omega_i \cup \Xi_i \cup \Upsilon_i = \Omega_i \cup \Xi_i = S_{\mathcal{M}}(o_i)$. Therefore, by Proposition 3, $l$ is redundant in $L_i$. □

In Example 1, $(size \neq 140)$ is globally redundant in $R_1$.

**Theorem 1 (Literal Redundancy (2)).** *A literal $l \in L_i$ is redundant iff it is locally redundant or globally redundant.*

*Proof.* $\Longrightarrow$: If $l$ is redundant, then by Proposition 3 we have $\Omega_i \cup \Xi_i = \Omega_i \cup \Xi_i \cup \Upsilon_i$. Observe that $\Upsilon_i \cap \Xi_i = \emptyset$. This is because if $x \in \Upsilon_i \cap \Xi_i$, then $x \models B \wedge L_i \wedge L_i^{\bar{l}}$ which is false because $L_i \wedge L_i^{\bar{l}}$ contains $l$ and $\neg l$. Therefore, there are only two cases for $\Omega_i \cup \Xi_i = \Omega_i \cup \Xi_i \cup \Upsilon_i$ to hold. Either $\Upsilon_i = \emptyset$ or $\Upsilon_i \neq \emptyset$ and $\Upsilon_i \in \Omega_i$. The first case is true iff $\mathcal{B} \wedge L_i \setminus \{l\} \models l$, that is, $l$ is locally redundant. The second case is true iff $\mathcal{B} \wedge L_i^{\bar{l}} \models L_{i_1} \vee \ldots, \vee L_{i_z}$, that is, $l$ is globally redundant
$\Longleftarrow$: trivial. □

**Corollary 1 (Assessing Literal Redundancy).** *A literal $l \in L_i$ is redundant iff one of the following conditions holds:*

1. ***Local redundancy**:*

$$\mathcal{B} \wedge (L_i \setminus \{l\}) \wedge \neg l \text{ is unsatisfiable.}$$

2. ***Global redundancy**: (1) does not hold, and*

$$\mathcal{B} \wedge L_i^{\bar{l}} \wedge \neg L_{i_1} \wedge \cdots \wedge \neg L_{i_z} \text{ is unsatisfiable.}$$

*Proof.* Immediate from Theorem 1 and Lemmas 3 and 4. □

Corollary 1 can be used to iteratively remove redundant literals, thus building decision sets with no rules/literal redundancies. It should be noted that different removal orders might lead to different decision sets.

*Example 2.* Suppose that $\mathcal{B} = (b \vee w) \wedge (\neg d \vee f)$ and $\mathcal{M} = \{R_1, R_2, R_3\}$ where $R_1 : (L_1 = a \wedge b, o_1)$, $R_2 : (L_2 = a \wedge w, o_1)$, and $R_3 : (L_3 = c \wedge d \wedge f, o_2)$.

- $\mathcal{B} \wedge L_3 \setminus \{f\} \models f$. Therefore, $f$ is locally redundant in $L_3$.
- $L_1^{\bar{b}} = a \wedge \neg b$, and $\mathcal{B} \wedge L_1^{\bar{b}} = (b \vee w) \wedge a \wedge \neg b \equiv a \wedge \neg b \wedge w$. Thus $\mathcal{B} \wedge L_1^{\bar{b}} \models R_2$ and therefore $b$ is globally redundant in $L_1$.

**Relation to Abductive Explanations.**

**Proposition 4.** *Suppose that $L_k \subseteq \{x_i = v_i^j \mid i \in [1, m], v_i^j \in D_i\}$. If $R_k = (L_k, o_k)$ fires on $x$, and there is no negative overlap involving $R_k$, then the features used in $L_k$ represent a WAXp.*

*Proof.* By construction. □

**Proposition 5.** *Suppose that $L_k \subseteq \{x_i = v_i^j \mid i \in [1, m], v_i^j \in D_i\}$. If $R_k = (L_k, o_k)$ fires on $v$, there is no negative overlap, and $L_k$ contains no (global or local) redundant literal, then the features from $L_k$ represent a AXp.* □

*Proof.* Suppose by contradiction that the features from $L_k$ do not define an AXp. Then there is a literal $l \in L_k$ such that $\forall x \in \mathbb{F}$ such that $x \models \mathcal{B}$, if $x \models L_k \setminus \{l\}$, then $\kappa_{\mathcal{M}(x)} = o_k$. In this case, $\mathcal{M}_l^k$ (i.e., the decision set identical to $\mathcal{M}$ except for $L_k$ which is replaced with $L_k \setminus \{l\}$) is equivalent to $\mathcal{M}$. Then, by Definition 5, $l$ is redundant, thus the contradiction. □

Observe that, if the conditions of Proposition 5 hold, then the literals in $L_k$ represent an AXp, and so can be identified manually by a human decision-maker. Otherwise, as proved in earlier work for the concrete case of decision lists [31], finding an AXp is computationally hard.

## 5 Experiments

We evaluate the different desired properties on different use cases including decision sets, decision trees, and anchor explanations. All SAT calls are performed using the PySAT toolkit[6] with its default configuration [26]. All experiments run on AppleM1 Pro that has 32G memory and 8 cores.

**Prediction Models & Datasets.** In order to make our evaluation as broad and as unbiased as possible, we selected datasets from the UCI machine learning repository [7] with the parameters: $\mathcal{P} = (Task, Min, Max, Nb, Types)$ on each use case (whenever relevant) where:

- $Task \subseteq \{classification, regression\}$ is the prediction task
- $Min$ (respectively $Max$) is the minimum (respectively $maximum$) size of the dataset.
- $Nb$ is the minimum number of inputs of each class present in the dataset in case of classification.
- $Types \subseteq \{numerical,\ binaly\}$ is the type of features.

We describe the different prediction models along with their tailored setting.

- **Orange (v3)**[8]: a library to learn decision sets for classification. The datasets are selected using the parameters $\mathcal{P} = (\{classification\}, 100, 10^6, 20, \{binary, numerical\})$.
- **Boomer [36]**[9]: A library for learning gradient boosted multi-label classification rules. We use the default Boomer datasets[10].
- **scikit-learn (v1.6.1)**[11] and **Interpretable AI (IAI)**[12] to learn decision trees (DTs) for classification and regression. scikit-learn learns trees in a greedy way with no guarantee of optimality whereas IAI learns optimal decision trees. The parameters used for the datasets are $\mathcal{P} = (\{classification, regression\}, 100, 4 * 10^6, 20, \{binary, numerical\})$.

**Background Knowledge.** In our empirical study, the Boolean variables that are used in the different decision sets represent a domain relation of the form $(x_f \bowtie v_f)$ where $\bowtie \in \{=, >, \geq, \leq, <\}$ for some $f \in \mathbb{F}$ and $v_f \in D_f$. We implemented a general purpose procedure to generate a background knowledge $\mathcal{B}$ for each use case that maintains domain coherence. For instance, if $(length > 30)$ and $(length = 17)$ appear in a decision set, then $\mathcal{B}$ contains the clause $\neg(length = 17) \vee \neg(length > 30)$.

Given a set of rules, for each feature $f$, we first compute a list, called $Val_f$, that contains all distinct values from the domain of $f$ that are used in the decision set (or Anchor explanations). $Val_f$ is increasingly ordered if the values are numerical. We also collect the set of unary relations used for $f$, denoted

[6] https://pysathq.github.io/.
[7] https://archive.ics.uci.edu
[8] https://orangedatamining.com
[9] https://github.com/mrapp-ke/MLRL-Boomer
[10] https://github.com/mrapp-ke/Boomer-Datasets
[11] https://scikit-learn.org/stable/
[12] https://www.interpretable.ai/

**Algorithm 2** Domain Constraints As a Background Knowledge

**Function:** Build $\mathcal{B}$
**Input:** $Relations_1, \ldots, Relations_m, Val_1, \ldots, Val_m$
**Output:** A background knowlegde $\mathcal{B}$ as a CNF
$\mathcal{B} = \emptyset$
**for** $f \in \{1, \ldots, m\}$ **do**
  **if** $`=' \in Relations_f$ **then**
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{(f = Val_f[i]) \implies \neg(f = Val_f[j]) \mid i < j \in [1, |Val_f|]\}$
  **end if**
  **if** $`>' \in Relations_f$ **then**
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{(f > Val_f[i+1]) \implies (f > Val_f[i]) \mid i \in [1, |Val_f| - 1]\}$
  **end if**
  **if** $`\geq' \in Relations_f$ **then**
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{(f \geq Val_f[i+1]) \implies (f \geq Val_f[i]) \mid i \in [1, |Val_f| - 1] :\}$
  **end if**
  **if** $\{`=', `\geq'\} \subseteq Relations_f$ **then**
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{(x = Val_f[i]) \implies (x \geq Val_f[i]) \mid i \in [1, |Val_f|]\}$
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{(x = Val_f[i]) \implies \neg(x \geq Val_f[i+1]) \mid i \in [1, |Val_f| - 1]\}$
  **end if**
  **if** $\{`=', `>'\} \subseteq Relations_f$ **then**
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{(x = Val_f[i]) \implies \neg(x > Val_f[i]) \mid i \in [1, |Val_f|]\}$
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{(x = Val_f[i+1]) \implies (x > Val_f[i]) \mid i \in [1, |Val_f| - 1]\}$
  **end if**
  **if** $\{`\geq', `>'\} \subseteq Relations_f$ **then**
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{(x > Val_f[i]) \implies (x \geq Val_f[i]) \mid i \in [1, |Val_f|]\}$
  **end if**
  **if** $\{`=', `\geq', `>'\} \subseteq Relations_f$ **then**
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{(x \geq Val_f[i]) \implies (x = Val_f[i]) \vee (x > Val_f[i]) \mid i \in [1, |Val_f|]\}$
  **end if**
**end for**
**Return** $\mathcal{B}$

by $Relations_f$, which can be any subset of $\{=, >, \geq, \leq, <\}$. The background knowledge $\mathcal{B}$ is then constructed using Algorithm 2 as a CNF. Note that we need only the three operators $>, \geq$, and $=$, since $(x > v)$ is equivalent to $(x \geq v - 1)$ and $(x < v)$ is equivalent to $(x \leq v - 1)$. Algorithm 2 follows standard procedures for encoding finite domains into SAT [33].

**Learning Setting.** For Orange, sickit-learn, and IAI, a grid search is used to select the best values for the maximum rule length, the minimum covered examples per rule, among others. Each dataset used with Orange, sickit-learn, and IAI is split into 80% for training and 20% for testing. Boomer's learning parameters are the default ones except for the maximum number of rules that we fix to 100 with one label classification. The detailed grid search parameters are given in Table 1. Cross validation is performed with 5 folds for all experiments using stratified sampling and each execution is randomly repeated 4 times.

**Experimental Pipeline.** All decision sets that have only one output are discarded. For each decision set, we first remove duplicate rules and rules that

**Table 1.** Grid Search Parameters.

| | Orange | Sklearn Class. | Sklearn Reg. | IAI Class. | IAI Reg. |
|---|---|---|---|---|---|
| Beam Width | 10,30 | - | - | - | - |
| Min Covered | 5,15 | - | - | - | - |
| Max Rule Length | 3,5 | - | - | - | - |
| Criterion | - | gini, entropy | sqr err, friedman mse | - | mse |
| Max Depth | - | 3,5,7,9 | 3,5,7,9,11 | 3,5,7 | 3,5,7 |
| Min Sample Leaf | - | 5,15,25 | 5,15,25 | - | - |
| Min Bucket | - | - - | - | 5,15 | 5,15 |

never fire. After this preprocessing, we run Algorithm 1 to find all overlap. Next, we look for redundant rules then remove them. Finally, we compute all locally/globally redundant literals. We use a timeout of one hour on each decision set to find all pairs negative overlap and rule/literal redundancies.

**Decision Set Statistics.**

- Train: Training accuracy (classification) or Training MSE (regression)
- Test: Testing accuracy (classification) or Training MSE (regression)
- NR: Number of rules
- NP: Cardinality of $\{o_i \mid \mathcal{M} = \cup_i (L_i, o_i)\}$
- TO: CPU time (s) to find all negative overlap
- TB: CPU time (s) to generate the background knowledge
- TC: CPU time (s) to find all redundant rules
- TR: CPU time (s) to find all redundant literals
- BS: Size of the background knowledge
- RS: Sum of the sizes of the rules
- RM: Maximum rule size
- NO: Number of negative overlap
- PO $= \frac{\text{NO}}{\text{Total}}$: Percentage of negative overlap where Total is the total number of pairs of rules associated to different predictions

**Model Statistics.** We report for each prediction model the following:

- DS: The total number of decision sets
- EX: The total number of decision sets that timed out
- IR: Number of instances that admit at least one redundant rule
- PR: Percentage of redundant rules for instances that admit at least one redundant rule
- IL: Number of instances that admit at least one locally redundant literal
- PL: Percentage of locally redundant literals for instances that admit at least one locally redundant literal
- IG: Number of instances that admit at least one globally redundant literal
- PG: Percentage of globally redundant literals for instances that admit at least one globally redundant literal

In the rest of the section, we focus on the most important observations.

**Table 2.** Summary of the Results. Floats are converted to integers.

| | DS | EX | NR | NP | TO | TB | TC | TR | BS | RS | RM | IR | IL | PL | IG | PG |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| sklearn classification | 196 | 21 | 35 | 3 | 0 | 0 | 0 | 94 | 23 | 233 | 5 | 0 | 123 | 7 | 175 | 33 |
| sklearn regression | 28 | 8 | 70 | 69 | 0 | 0 | 8 | 879 | 85 | 541 | 7 | 0 | 20 | 18 | 0 | 0 |
| IAI classification | 177 | 0 | 17 | 4 | 0 | 0 | 0 | 19 | 13 | 96 | 4 | 0 | 82 | 3 | 135 | 10 |
| IAI regression | 28 | 0 | 56 | 53 | 0 | 0 | 3 | 333 | 77 | 367 | 5 | 0 | 25 | 15 | 2 | 0 |
| Orange | 127 | 12 | 175 | 2 | 76 | 0 | 2 | 10 | 12139 | 405 | 3 | 16 | 1 | 0 | 13 | 0 |
| Boomer | 180 | 16 | 97 | 49 | 0 | 0 | 0 | 21 | 30 | 180 | 11 | 42 | 6 | 0 | 20 | 0 |

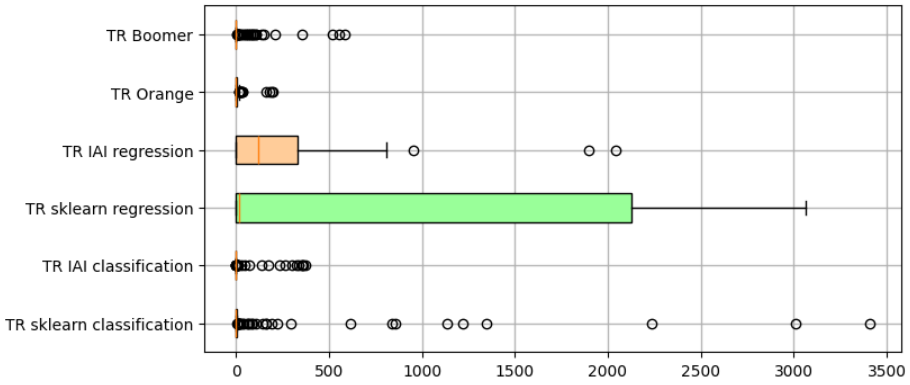



**Fig. 1.** Box Plot of TR. The X-axis is the time (s).

**Summary.** Table 2 gives the full statistics for each learning model[13]. Instance-related statistics are averaged for each model. Decision sets that are worse than random guess are ignored. Instance statistics are averaged for each prediction model. Only the results of the experiments that did not reach the timeout are reported. The time to generate the background knowledge (TB) is often less than a second. The time to find redundant rules (TC) is often few seconds, except for some decision sets where it took about a minute. The runtime to find all literal redundancies (TR), however, is much longer. To observe this more accurately, we present in Figure 1 its box plot across all models. This is expected because every literal is checked for redundancy by application of Corollary 1.

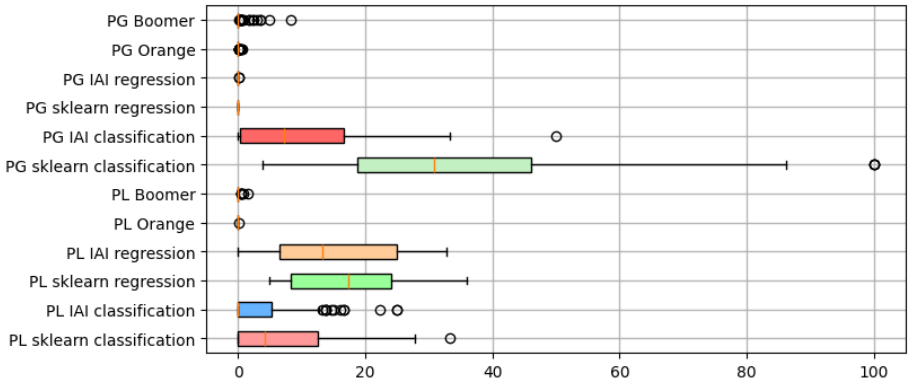
### 5.1 Rule and Literal Redundancy

We are interested in this section in the evaluation of the presence of redundant rules and locally/globally redundant literals, their correlations with other characteristics, as well as the efficiency of our approach.

**Redundancy.** We note first that rule redundancy does not occur often as we can see in column IR in Table 2 except for Boomer. Figure 2 represents a box

[13] The detailed results can be found at `https://siala.github.io/data/2025-ecml/`

**Fig. 2.** Local/Global Redundancies. The X-axis represents the values of PL/PG.




plot of the percentage of local (respectively global) redundancies PL (respectively PG) for all learning models. Orange and Boomer barely exhibit literal redundancies (see columns IL and IG in Table 2). Regression models did not show any global literal redundancy except for 2 cases with IAI regression trees. This is expected because for a literal to be globally redundant, there should be at least two rules predicting the exact same value, which is rare in regression. Classification trees, however, exhibit a noticeable presence of global redundancy ('PG IAI classification' and 'PG sklearn classification'). Figure 2 shows a significant presence of local redundancy in all tree models. We note that for each prediction task (regression, classification), IAI trees have fewer local/global redundancies than sklearn trees (in terms of the median and the maximum values). This suggests that optimal trees tend to reduce redundancy.

**Correlations.** We looked into different correlations between local/global redundancy and other statistics. We report the results only for models where at least 30% of its decision trees/sets exhibit local/global redundancy. There was a moderate negative correlation of global redundancy with the number of prediction outcomes (i.e., size of $\mathcal{V}$) with scikit-learn and IAI classification trees. Figure 3 shows the most important correlations of local redundancy with the statistics mentioned earlier. For instance, on the x-axis, with NR we show the correlation of the local redundancy values found by each model with the number of rules. Clearly local redundancy with scikit-learn regression trees highly correlates with NR, NP, BS, RM. IAI regression trees has the same tendency.

### 5.2 Negative Overlap

We evaluate the presence of negative overlap on Orange and Boomer and their relationship with relevant statistics. Boomer timed out on 4 datasets (emotions, image, scene, yeast) after the one hour time limit. The results are summarized in Table 3 for instances that did not timeout. The most important observation is the high percentage of negative overlap (column PO). Indeed, with Boomer

**Fig. 3.** Pearson Correlations of Local Redundancies (PL)

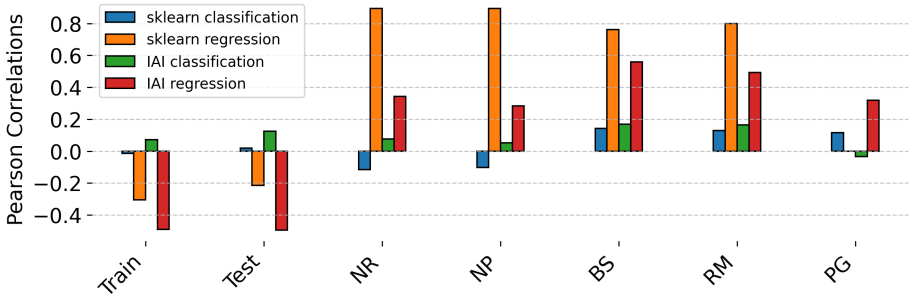



**Table 3.** Summary of the Negative Overlap Results. Floats are converted to Integers.

| | DS | EX | Train | Test | NR | NP | IR | PR | TO | TR | BS | RS | RM | PO |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Orange | 127 | 12 | 70 | 71 | 175 | 2 | 16 | 0 | 76 | 10 | 12139 | 405 | 3 | 50 |
| Boomer | 180 | 16 | 95 | 95 | 97 | 49 | 42 | 1 | 0 | 21 | 30 | 180 | 11 | 99 |

decision sets, almost every pair of rules with different predictions overlap. Such an observation is worth reporting to the user. The results are less spectacular for orange with an average close to 50% but still worth noting. The runtime to find all negative overlap per instance is not negligible.

**Negative Overlap in Boomer.** As the results in this section confirm (see Table 3), Boomer [36] exhibits extensive negative overlap. This is to be expected. In contrast with the approach outlined in this paper, where negative overlap is targeted as a reason for non-interpretability, Boomer exploits boosting (and as a result negative overlap) to build high-accuracy rule ensembles. The theoretical and practical advantages of boosting are well-known [16, 9], namely to allow the learning of strong classifiers. As argued in this paper, a downside of negative overlap (and so of rule ensembles) is that finding explanations becomes a computationally-hard challenge. Our experiments are reported for completeness, and confirm the previous remarks.

### 5.3 Application to Anchor Explanations

Anchors are well-known model-agnostic explanations representing local, "sufficient" conditions for predictions [38]. The question we ask here addresses precisely one of the open questions in [38]: How to find potentially conflicting anchors? To answer this question, we generate anchors for different inputs, then apply our approach to find negative overlap between anchors.

We reproduced the exact experiments in [38] with the three datasets: *adult* for predicting whether a person makes $> 50K$ annually; *rcdv* for predicting recidivism for individuals released from prison; and *lending* for predicting whether a loan on the Lending Club website will turn out bad. For each dataset, four models are used for prediction: boosted trees with xgboost, random forest, logistic

**Table 4.** Anchor Experiments

| Learner | Dataset | Train | Test | NR | TO | NO | PO | RM |
|---|---|---|---|---|---|---|---|---|
| xgboost | recidivism | 92.39 | 74.33 | 333 | 0 | 87 | 0.31 | 17 |
| randomforest | recidivism | 93.52 | 75.46 | 321 | 0 | 65 | 0.25 | 17 |
| logistic | recidivism | 62.59 | 60.00 | 196 | 0 | 735 | 7.81 | 12 |
| nn | recidivism | 87.47 | 71.49 | 341 | 1 | 150 | 0.52 | 17 |
| xgboost | lending | 90.10 | 82.89 | 260 | 0 | 384 | 2.47 | 15 |
| randomforest | lending | 91.25 | 83.60 | 278 | 0 | 207 | 1.18 | 15 |
| logistic | lending | 82.56 | 83.51 | 50 | 0 | 54 | 9.38 | 14 |
| nn | lending | 88.00 | 82.54 | 159 | 0 | 66 | 1.07 | 16 |
| xgboost | adult | 90.35 | 84.26 | 565 | 8 | 3195 | 4.03 | 14 |
| randomforest | adult | 93.52 | 85.60 | 558 | 7 | 2534 | 3.27 | 13 |
| logistic | adult | 83.00 | 82.98 | 378 | 3 | 2788 | 7.86 | 13 |
| nn | adult | 92.47 | 83.62 | 597 | 11 | 3212 | 3.61 | 14 |

regression, and neural networks. Each model is built using the exact configuration in the original paper [38]. For each dataset and each model, we generate all anchors of the validation set and look for all negative overlap.

Table 4 presents the results for each dataset and each model. As we can see, negative overlap in Anchor explanations is present in all use cases. Often, anchors of random forests exhibit the lowest percentage of negative overlap, whereas those of logistic regression have the highest percentage. We also observe that the best (and respectively, worst) models in terms of prediction quality tend to have the lowest (respectively, highest) percentages of negative overlap. These observations suggest that the quality of Anchor explanations depends on the prediction quality of the learner/model.

## 6 Conclusions

This paper investigates the occurrence of negative facets of decision sets, namely negative overlap and (global or local) literal redundancy. Dedicated algorithms for their identification are proposed. Furthermore, the paper reveals the tight relationship between decision sets for which manual explanations can be devised, and the non-existence of the aforementioned negative facets. A first set of experiments confirms that these negative facets occur ubiquitously in existing implementations of decision sets, thus rendering unrealistic the manual identification of explanations. A second set of experiments confirms that the explanations obtained with the well-known explainer Anchors will also exhibit the same negative facets.

**Acknowledgments.** Mohamed Siala would like to thank INSA Toulouse for funding his research visit to the University of Lleida. This work was supported in part by the MCIN/AEI/10.13039/501100011033/FEDER, UE under the project PID2022-139835NB-C22. This work was supported in part by the Spanish Government under

grant PID 2023-152814OB-I00. The authors at University of Lleida would like to thank the Catalan Government for the quality accreditation given to their research group GREiA (2021 SGR 1615).

## References

1. Adadi, A., Berrada, M.: Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). IEEE Access **6**, 52138–52160 (2018)
2. Atzmueller, M., Fürnkranz, J., Kliegr, T., Schmid, U.: Explainable and interpretable machine learning and data mining. Data Min. Knowl. Discov. **38**(5), 2571–2595 (2024)
3. Audemard, G., Lagniez, J., Marquis, P., Szczepanski, N.: Deriving provably correct explanations for decision trees: The impact of domain theories. In: IJCAI. pp. 3688–3696 (2024)
4. Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability - Second Edition, Frontiers in Artificial Intelligence and Applications, vol. 336. IOS Press (2021)
5. Breiman, L.: Statistical modeling: The two cultures. Statistical science **16**(3), 199–231 (2001)
6. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth (1984)
7. Carvalho, D.V., Pereira, E.M., Cardoso, J.S.: Machine learning interpretability: A survey on methods and metrics. Electronics **8**(8), 832 (2019)
8. Clark, P., Niblett, T.: The CN2 induction algorithm. Machine Learning **3**(4), 261–283 (1989)
9. Collins, M., Schapire, R.E., Singer, Y.: Logistic regression, AdaBoost and Bregman distances. Mach. Learn. **48**(1-3), 253–285 (2002)
10. Darwiche, A.: Logic for explainable AI. In: LICS. pp. 1–11 (2023)
11. Demirovic, E., Lukina, A., Hebrard, E., Chan, J., Bailey, J., Leckie, C., Ramamohanarao, K., Stuckey, P.J.: MurTree: optimal decision trees via dynamic programming and search. J. Mach. Learn. Res. **23**, 26:1–26:47 (2022)
12. Demsar, J., et al.: Orange: data mining toolbox in python. J. Mach. Learn. Res. **14**(1), 2349–2353 (2013)
13. Dwivedi, R., et al.: Explainable AI (XAI): core ideas, techniques, and solutions. ACM Comput. Surv. **55**(9), 194:1–194:33 (2023)
14. Florio, A.M., Martins, P., Schiffer, M., Serra, T., Vidal, T.: Optimal decision diagrams for classification. pp. 7577–7585. AAAI Press (2023)
15. Freitas, A.A.: Comprehensible classification models: a position paper. SIGKDD Explor. Newsl. **15**(1), 1–10 (Mar 2014)
16. Freund, Y.: Boosting a weak learning algorithm by majority. Inf. Comput. **121**(2), 256–285 (1995)
17. Fürnkranz, J., Gamberger, D., Lavrac, N.: Foundations of Rule Learning. Cognitive Technologies, Springer (2012)
18. Fürnkranz, J., Kliegr, T.: A brief overview of rule learning. In: RuleML. pp. 54–69 (2015)
19. Gorji, N., Rubin, S.: Sufficient reasons for classifier decisions in the presence of domain constraints. In: AAAI. pp. 5660–5667 (2022)
20. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. ACM Comput. Surv. **51**(5), 93:1–93:42 (2019)

21. Hu, H., Huguet, M., Siala, M.: Optimizing binary decision diagrams with maxsat for classification. In: AAAI. pp. 3767–3775. AAAI Press (2022)
22. Hu, H., Siala, M., Hebrard, E., Huguet, M.: Learning optimal decision trees with maxsat and its integration in adaboost. In: IJCAI. pp. 1170–1176 (2020)
23. Hüllermeier, E., Fürnkranz, J., Mencía, E.L., Nguyen, V., Rapp, M.: Rule-based multi-label classification: Challenges and opportunities. In: RuleML. pp. 3–19 (2020)
24. Huysmans, J., Dejaeger, K., Mues, C., Vanthienen, J., Baesens, B.: An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. Decision Support Systems **51**(1), 141–154 (2011)
25. Hyafil, L., Rivest, R.L.: Constructing optimal binary decision trees is NP-complete. Inf. Process. Lett. **5**(1), 15–17 (1976)
26. Ignatiev, A., Tan, Z.L., Karamanos, C.: Towards universally accessible SAT technology. In: SAT. pp. 4:1–4:11 (2024)
27. Izza, Y., Ignatiev, A., Marques-Silva, J.: On tackling explanation redundancy in decision trees. J. Artif. Intell. Res. **75**, 261–321 (2022)
28. Lakkaraju, H., Bach, S.H., Leskovec, J.: Interpretable decision sets: A joint framework for description and prediction. In: KDD. pp. 1675–1684 (2016)
29. Lipton, Z.C.: The mythos of model interpretability. Commun. ACM **61**(10), 36–43 (2018)
30. Lundberg, S.M., Lee, S.: A unified approach to interpreting model predictions. In: NeurIPS. pp. 4765–4774 (2017)
31. Marques-Silva, J., Ignatiev, A.: No silver bullet: interpretable ML models must be explained. Frontiers Artif. Intell. **6** (2023)
32. Minh, D., Wang, H.X., Li, Y.F., Nguyen, T.N.: Explainable artificial intelligence: a comprehensive review. Artif. Intell. Rev. **55**(5), 3503–3568 (2022)
33. Ohrimenko, O., Stuckey, P.J., Codish, M.: Propagation via lazy clause generation. Constraints An Int. J. **14**(3), 357–391 (2009)
34. Pedregosa, F., et al.: Scikit-learn: Machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
35. Rapp, M., Fürnkranz, J., Hüllermeier, E.: On the efficient implementation of classification rule learning. Adv. Data Anal. Classif. **18**(4), 851–892 (2024)
36. Rapp, M., Mencía, E.L., Fürnkranz, J., Nguyen, V., Hüllermeier, E.: Learning gradient boosted multi-label classification rules. In: ECML. pp. 124–140 (2020)
37. Ribeiro, M.T., Singh, S., Guestrin, C.: "why should I trust you?": Explaining the predictions of any classifier. In: KDD. pp. 1135–1144 (2016)
38. Ribeiro, M.T., Singh, S., Guestrin, C.: Anchors: High-precision model-agnostic explanations. In: AAAI. pp. 1527–1535 (2018)
39. Rivest, R.L.: Learning decision lists. Mach. Learn. **2**(3), 229–246 (1987)
40. Rudin, C.: Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature machine intelligence **1**(5), 206–215 (2019)
41. Rudin, C., Chen, C., Chen, Z., Huang, H., Semenova, L., Zhong, C.: Interpretable machine learning: Fundamental principles and 10 grand challenges. Statistic Surveys **16**, 1–85 (2022)
42. Schwalbe, G., Finzel, B.: A comprehensive taxonomy for explainable artificial intelligence: a systematic survey of surveys on methods and concepts. Data Min. Knowl. Discov. **38**(5), 3043–3101 (2024)
43. Shwayder, K.: Conversion of limited-entry decision tables to computer programs - A proposed modification to Pollack's algorithm. Commun. ACM **14**(2), 69–73 (1971)
44. Yu, J., Ignatiev, A., Stuckey, P.J., Bodic, P.L.: Learning optimal decision sets and lists with SAT. J. Artif. Intell. Res. **72**, 1251–1279 (2021)