Eliminating The Impossible, Whatever Remains Must Be True

On Extracting and Applying Background Knowledge In The Context Of Formal Explanations

Jinqiang Yu[1,4], **Alexey Ignatiev**[1], **Peter J. Stuckey**[1,4], **Nina Narodytska**[2], **Joao Marques-Silva**[3]

[1] Monash University, Melbourne, Australia
[2] VMWare Research, Palo Alto, USA
[3] IRIT, CNRS, Toulouse, France
[4] ARC Training Centre in OPTIMA, Melbourne, Australia
jinqiang.yu@monash.edu, alexey.ignatiev@monash.edu, peter.stuckey@monash.edu,
nnarodytska@vmware.com, joao.marques-silva@irit.fr

Abstract

The rise of AI methods to make predictions and decisions has led to a pressing need for more explainable artificial intelligence (XAI) methods. One common approach for XAI is to produce a post-hoc explanation, explaining why a black box ML model made a certain prediction. Formal approaches to post-hoc explanations provide succinct reasons for *why* a prediction was made, as well as *why not* another prediction was made. But these approaches assume that features are independent and uniformly distributed. While this means that "why" explanations are correct, they may be longer than required. It also means the "why not" explanations may be suspect as the counterexamples they rely on may not be meaningful. In this paper, we show how one can apply background knowledge to give more succinct "why" formal explanations, that are presumably easier to interpret by humans, and give more accurate "why not" explanations. In addition, we show how to use existing rule induction techniques to efficiently extract background information from a dataset, and also how to report which background information was used to make an explanation, allowing a human to examine it if they doubt the correctness of the explanation.

1 Introduction

Recent years have witnessed rapid advances in Artificial Intelligence (AI) and Machine Learning (ML) algorithms revolutionizing all aspects of human lives (LeCun, Bengio, and Hinton 2015; ACM 2018). An ever growing range of practical applications of AI and ML, on the one hand, and a number of critical issues observed in modern AI systems (e.g. decision bias (Angwin et al. 2016) and brittleness (Szegedy et al. 2014)), on the other hand, gave rise to the quickly advancing area of theory and practice of Explainable AI (XAI).

Several major approaches to XAI have been proposed in the recent past. Besides tackling XAI through computing *interpretable* ML models directly (Rudin 2019), or through the use of interpretable models for approximating complex *black-box* ML models (Ribeiro, Singh, and Guestrin 2016), the most prominent approach to XAI is to compute *post-hoc explanations* to ML predictions on demand (Lundberg and Lee 2017; Ribeiro, Singh, and Guestrin 2018). Prior work distinguishes post-hoc (*abductive*) explanations answering a *"why?"* question and (*contrastive*) explanations targeting a *"why not?"* question (Miller 2019). Heuristic approaches to post-hoc explainability are known to suffer from a number of fundamental explanation quality issues (Narodytska et al. 2019; Ignatiev, Narodytska, and Marques-Silva 2019c; Camburu et al. 2019; Ignatiev 2020), including the existence of out-of-distribution attacks (Slack et al. 2020). A promising alternative is formal explainability where explanations are computed as prime implicants of the decision function associated with ML predictions (Shih, Choi, and Darwiche 2018). Formal explanations have also been related with abductive reasoning (Ignatiev, Narodytska, and Marques-Silva 2019a,b).

Although provably correct and minimal, formal explanations have a few limitations. In order to provide provable correctness guarantees that a subset of features is sufficient for an ML prediction, formal approaches have to take into account the complete feature space assuming that the features are independent and uniformly distributed (Wäldchen et al. 2021). This makes a formal reasoner check all the combinations of feature values, including those that realistically can *never appear* in practice. This issue is caused by the inability of modern (both formal and heuristic[1]) explanation approaches to account for background knowledge associated with the problem domain of the target dataset. It results in formal explanations being unnecessarily long, which makes them hard for a human decision maker to interpret.

Motivated by this limitation, our work focuses on computing both abductive and contrastive formal explanations making use of background knowledge, and makes the following contributions: First, given a training dataset, an efficient generic approach to extracting background knowledge in the form of highly accurate *if-then* rules is proposed. Following recent work on using constraints in compilation-based formal explainability (Gorji and Rubin 2022), accurate background knowledge is argued to be the key to good quality explanations. The approach builds on a recent formal method for learning decision sets (Ignatiev et al. 2021) and is able to extract reasonably short rules representing relations between various features of the target dataset. Also, as our approach is designed to enumerate 100% accurate rules, its performance is shown to be on par with a modern implementation

[1]The lack of background knowledge support pertains to heuristic approaches as well by making them error-prone (Slack et al. 2020; Shrotri et al. 2022).

of the well-known Apriori and Eclat association rule mining algorithms (Agrawal and Srikant 1994; Zaki et al. 1997). Second, a novel approach to computing formal explanations taking into account background knowledge is proposed, *independent* of the nature of the background knowledge; the only requirement imposed is that the knowledge must be represented as a conjunction of constraints. Third, we prove theoretically that the use of background knowledge positively affects the quality of both abductive and contrastive explanations, thus, helping to build trust in the underlying AI systems. Fourth, we develop an effective way to discover which background knowledge rules are used in extracting an explanation. This enables a human decision maker to examine whether or not the rules used are meaningful, which further facilitates human trust in the explanations computed. Fifth and finally, motivated by the results of (Ignatiev, Narodytska, and Marques-Silva 2019c; Ignatiev 2020), we argue that background knowledge helps one assess the correctness of heuristic ML explainers (Ribeiro, Singh, and Guestrin 2016; Lundberg and Lee 2017; Ribeiro, Singh, and Guestrin 2018) more accurately since it blocks impossible combinations of feature values. Namely, we show that the estimated correctness of SHAP, LIME, and Anchor may improve significantly when background knowledge is available.

2 Preliminaries

SAT, MaxSAT, and SMT. Definitions standard in *propositional satisfiability* (SAT) and *maximum satisfiability* (MaxSAT) solving are assumed (Biere et al. 2021). SAT and MaxSAT formulas are assumed to be propositional. A propositional formula φ is considered to be in *conjunctive normal form* (CNF) if it is a conjunction (logical *"and"*) of clauses, where a *clause* is a disjunction (logical *"or"*) of literals, and a *literal* is either a Boolean variable b or its *negation* $\neg b$. Whenever convenient, a clause is treated as a set of literals. A *truth assignment* μ is a mapping from the set of variables in φ to $\{0, 1\}$. A clause is *satisfied* by truth assignment μ if one of its literals is assigned value 1 by μ; otherwise, the clause is said to be *falsified*. If all clauses of formula φ are satisfied by assignment μ then μ also satisfies φ; otherwise, φ is falsified by μ. A formula φ is said to be *satisfiable* if there is an assignment μ that satisfies φ; otherwise, φ is *unsatisfiable*.

In the context of unsatisfiable formulas, the maximum satisfiability problem is to find a truth assignment that maximizes the number of satisfied clauses. Hereinafter, we will make use of a variant of MaxSAT called Partial (Unweighted) MaxSAT (Biere et al. 2021, Chapters 23 and 24). The formula φ in Partial (Unweighted) MaxSAT is a conjunction of *hard* clauses $\mathcal{H}$, which must be satisfied, and *soft* clauses $\mathcal{S}$, which represent a preference to satisfy them, i.e. $\varphi = \mathcal{H} \wedge \mathcal{S}$. The Partial Unweighted MaxSAT problem aims at finding a truth assignment that satisfies all the hard clauses while maximizing the total number of satisfied soft clauses.

Note that we consider a family of ML classifiers such that their decision making process can be represented logically as a propositional formula. This is needed for applying formal reasoning about ML model behavior, as well as for representing background knowledge extracted. Finally, a logical representation of boosted tree models will require us to apply an extension of propositional logic to decidable fragments of first-order logic (FOL). Namely, we will assume the use of *satisfiability modulo theories* (SMT) in the theory of linear arithmetic over reals, i.e. the concept of a clause will be lifted to *linear constraints* over real variables. Optimization problems for SMT can be defined analogously to MaxSAT.

Table 1: Several examples extracted from *adult* dataset.

Education	Status	Occupation	Relationship	Sex	Hours/w	Target
HighSchool	Married	Sales	Husband	Male	40 to 45	$\geq$ 50k
Bachelors	Married	Sales	Wife	Female	$\leq$ 40	$\geq$ 50k
Masters	Married	Professional	Wife	Female	$\geq$ 45	$\geq$ 50k
Masters	Married	Professional	Wife	Female	$\leq$ 40	$\geq$ 50k
Dropout	Separated	Service	Not-in-family	Male	$\leq$ 40	$<$ 50k
Dropout	Never-Married	Blue-Collar	Unmarried	Male	$\geq$ 45	$\geq$ 50k

Classification Problems. Classification problems consider a set of classes $\mathcal{K} = \{c_1, c_2, \ldots, c_k\}$, and a set of features $\mathcal{F} = \{1, \ldots, m\}$. The value of each feature $i \in \mathcal{F}$ is taken from a domain $\mathcal{D}_i$, which can be integer, real-valued or Boolean. Therefore, the complete feature space is defined as $\mathbb{F} \triangleq \prod_{i=1}^{m} \mathcal{D}_i$. A concrete point in feature space is represented by $\mathbf{v} = (v_1, \ldots, v_m) \in \mathbb{F}$, where each $v_i \in \mathbf{v}$ is a constant taken by feature $i \in \mathcal{F}$. An *instance* or *example* is denoted by a specific point $\mathbf{v} \in \mathbb{F}$ in feature space and its corresponding class $c \in \mathcal{K}$, i.e. a pair $(\mathbf{v}, c)$ represents an instance. Moreover, the notation $\mathbf{x} = (x_1, \ldots, x_m)$ denotes an arbitrary point in feature space, where each $x_i \in \mathbf{x}$ is a variable taking values from its corresponding domain $\mathcal{D}_i$ and representing feature $i \in \mathcal{F}$.

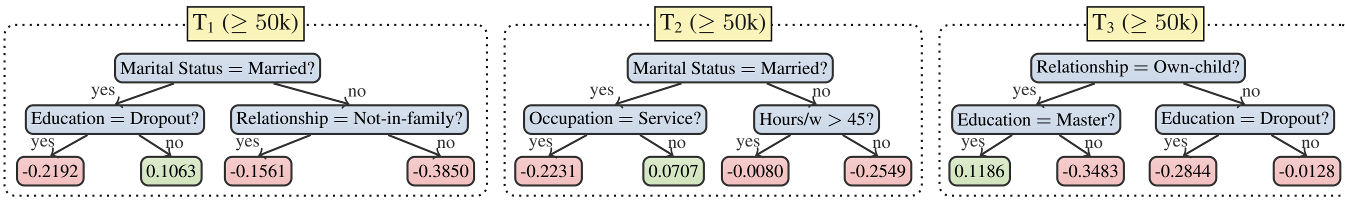
A classifier defines a classification function $\tau : \mathbb{F} \rightarrow \mathcal{K}$. Whenever convenient, a classification function τ and an associated class c are represented by a *decision predicate* $\tau_c : \mathbb{F} \rightarrow \{0, 1\}$. A decision predicate τ_c is given a specific class $c \in \mathcal{K}$, such that $\forall(\mathbf{x} \in \mathbb{F}).\tau_c(\mathbf{x}) \leftrightarrow (\tau(\mathbf{x}) = c)$. There are many ways to learn classifiers for a given dataset. In this paper, we consider: *decision lists* (DLs) (Rivest 1987; Clark and Niblett 1989), *boosted trees* (BTs) (Friedman 2001; Chen and Guestrin 2016), and *binarized neural networks* (BNNs) (Hubara et al. 2016).

Example 1. *Consider the data shown in Table 1. It represents a snapshot of instances taken from a simplified version*[2] *of the* adult *dataset (Kohavi 1996). Figure 1 illustrates DL and BT models trained for this dataset. Observe that for instance* $\mathbf{v} = \{$*Education* $=$ *HighSchool*, *Status* $=$ *Married, Occupation* $=$ *Sales, Relationship* $=$ *Husband*, *Sex* $=$ *Male*, *Hours/w* $=$ *40 to 45*$\}$ *from Table 1, rule* R_2 *in the DL in Figure 1a predicts* $\geq 50k$. *Similarly, the sum of the weights* (0.1063, 0.0707 *and* -0.0128 *in the* 3 *trees, respectively) for prediction* $\geq 50k$ *is positive* (0.1642)

[2]For simplicity, the running example used throughout the text will correspond to a *simplified* version of the *adult* dataset (Kohavi 1996), where some of the features are dropped. Note that the experimental results shown below deal with the *original* datasets.

R_0:	IF	Education = Dropout	THEN	Target $< 50k$
R_1:	ELSE IF	Occupation = Service	THEN	Target $< 50k$
R_2:	ELSE IF	Status = Married $\wedge$ Relationship = Husband	THEN	Target $\geq 50k$
R_3:	ELSE IF	Status = Married $\wedge$ Relationship = Wife	THEN	Target $\geq 50k$
R_{DEF}:	ELSE		THEN	Target $< 50k$

(a) Decision list.



(b) Boosted tree (Chen and Guestrin 2016) consisting of 3 trees with the depth of each tree at most 2.

Figure 1: Example DL and BT models trained on the well-known *adult* classification dataset.

in the BT in Figure 1b, and so the BT model also predicts $\geq 50k$ *for the aforementioned instance* $\mathbf{v}$.

Interpretability and Explanations. Interpretability is not formally defined since it is a subjective concept (Lipton 2018). In this paper, we define interpretability as the conciseness of the computed explanations for an ML model to justify a provided prediction. The definition of explanation for an ML model is built on earlier work (Shih, Choi, and Darwiche 2018; Ignatiev, Narodytska, and Marques-Silva 2019a; Darwiche and Hirth 2020; Audemard, Koriche, and Marquis 2020; Marques-Silva and Ignatiev 2022), where explanations are equated with *abductive explanations* (AXps), which are subset-minimal sets of features sufficing to explain the prediction given by an ML model. Concretely, given an instance $\mathbf{v} \in \mathbb{F}$ and a computed prediction $c \in \mathcal{K}$, i.e. $\tau(\mathbf{v}) = c$, an AXp is a subset-minimal set of features $\mathcal{X} \subseteq \mathcal{F}$, such that

$$\forall(\mathbf{x} \in \mathbb{F}). \bigwedge\nolimits_{i \in \mathcal{X}} (x_i = v_i) \rightarrow (\tau(\mathbf{x}) = c) \tag{1}$$

Abductive explanations are also prime implicants of the decision predicate τ_c and hence a *prime implicant* (PI) explanation is another name for an AXp.

Example 2. *Consider the models in Figure 1 and instance* $\mathbf{v}$ *from Example 1. By examining the DL model, specifying Education = HighSchool, Status = Married, Occupation = Sales, and Relationship = Husband guarantees that any compatible instance is classified by* R_2 *independent of the values of other features, i.e. Sex and Hours/w. Similarly, the prediction of an instance is guaranteed to be* $\geq 50k$ *in Figure 1b as long as the feature values above are used, since the sum of weights is promised to be* $0.1063 + 0.0707 + -0.0128 = 0.1642$ *for class* $\geq 50k$. *Therefore, the (only) AXp* $\mathcal{X}$ *for the prediction of* $\mathbf{v}$ *is* {*Education, Status, Occupation, Relationship*} *in both models.*

We also consider *contrastive explanations* (CXps) defined as subset-minimal sets of features that are necessary to change the prediction if the features of a CXp are allowed to take arbitrary values from their domains. Formally and following (Ignatiev et al. 2020), a CXp for prediction $\tau(\mathbf{v}) = c$ is defined as a minimal subset $\mathcal{Y} \subseteq \mathcal{F}$ such that

$$\exists(\mathbf{x} \in \mathbb{F}). \bigwedge\nolimits_{i \notin \mathcal{Y}} (x_i = v_i) \wedge (\tau(\mathbf{x}) \neq c) \tag{2}$$

Example 3. *Consider the setup of Example 2. Given either model,* $\mathcal{Y} = \{$*Occupation*$\}$ *is a CXp for instance* $\mathbf{v}$ *because the prediction for* $\mathbf{v}$ *can be changed if feature 'Occupation' is allowed to take another value, e.g. if the value is changed to 'Service'. Similarly, changing the value of feature 'Occupation' to 'Service' triggers that the weights in the 3 trees become* 0.1063, -0.2231 *and* -0.0128. *Therefore, the total weight is* -0.0982, *i.e. the prediction is changed. By further examining the two models, other subsets of features can be identified as CXps for* $\mathbf{v}$. *The set of CXps is* $\mathbb{Y} = \{\{$*Education*$\}, \{$*Status*$\}, \{$*Occupation*$\}, \{$*Relationship*$\}\}$, *while the set of AXps demonstrated in Example 2 is* $\mathbb{X} = \{\{$*Education, Status, Occupation, Relationship*$\}\}$.

Recent work, which builds on the seminal work of Reiter (Reiter 1987), establishes a minimal hitting set (MHS) duality relationship between AXps and CXps (Ignatiev et al. 2020). In other words, each CXp *minimally hit* every AXp, and vice-versa. The explanations enumeration algorithms used in this paper employ this fact.

Example 4. *Observe how the minimal hitting set duality holds for the set of abductive explanations* $\mathbb{X}$ *and the set of contrastive explanations* $\mathbb{Y}$ *shown in Example 3. The only abductive explanation minimally hits all the contrastive explanations and vice versa.*

There is a growing body of recent work on formal explanations (Marques-Silva et al. 2020, 2021; Izza and Marques-Silva 2021; Ignatiev and Marques-Silva 2021; Arenas et al. 2021; Wäldchen et al. 2021; Darwiche and Marquis 2021; Malfa et al. 2021; Boumazouza et al. 2021; Blanc, Lange, and Tan 2021; Izza, Ignatiev, and Marques-Silva 2022; Gorji and Rubin 2022; Ignatiev et al. 2022; Huang et al. 2022; Marques-Silva and Ignatiev 2022; Amgoud and Ben-Naim 2022; Ferreira et al. 2022; Arenas et al. 2022).

3 Extracting Background Knowledge

Recent work (Gorji and Rubin 2022) argues that background knowledge is helpful in the context of formal explanations. The idea is that, if identified, background knowledge may help forbid some of the combinations of feature values that would otherwise have to be taken into account by a formal reasoner, thus, slowing the reasoner down and making the explanations unnecessarily long. But the question of how such knowledge can be obtained in an automated way remains open.

Example 5. *Assume that Table 1 represents* trustable *information. The following two rules can be extracted:*

- IF *Relationship* = *Husband* THEN *Status* = *Married*
- IF *Relationship* = *Wife* THEN *Status* = *Married*

These rules may be used to discard feature Status *when computing explanations as long as* Relationship *equals either* Husband *or* Wife *because of the implications identified.*

Here we describe the MaxSAT-based approach to automatically extract background knowledge, which represents hidden relations between features of a dataset if the dataset is assumed to be trustable. It builds on the recent two-stage approach (Ignatiev et al. 2021) to learning smallest size decision sets (Kamath et al. 1992; Lakkaraju, Bach, and Leskovec 2016; Ignatiev et al. 2018; Malioutov and Meel 2018; Ghosh and Meel 2019; Yu et al. 2020, 2021). Concretely, we apply the first stage of (Ignatiev et al. 2021) which enumerates individual decision rules given a dataset, using MaxSAT.

Without diving into the details, the idea of (Ignatiev et al. 2021) is as follows. Given training data $\mathcal{E}$ and target class $c \in \mathcal{K}$, a MaxSAT solver is invoked multiple times, each producing a unique subset-minimal (irreducible) rule in the form of *"IF* antecedent *THEN* prediction c*"*, where the antecedent is a set of feature values. The MaxSAT solver is fed with various CNF constraints and an objective function targeting rule size minimization. The approach also detects and blocks *symmetric rules*, i.e. those that do not contribute new information to the rule-based representation of class $c \in \mathcal{K}$.

We can modify the MaxSAT approach outlined above to learning background knowledge in the form of decision rules, i.e. identifying the dependency of a feature $i \in \mathcal{F}$ on other features $j \in \mathcal{F} \setminus \{i\}$. For this, we need to discard the prediction column from the dataset $\mathcal{E}$ and instead focus on a feature $i \in \mathcal{F}$, consider some of its values $v_{ij} \in \mathcal{D}_i$ and "pretend" to compute decision rules for a "fake class" $x_i = v_{ij}$. Thanks to the properties of the approach of (Ignatiev et al. 2021), all the rules computed are guaranteed to be subset-minimal and to respect training data $\mathcal{E}$. Once all the rules for feature $i \in \mathcal{F}$ and value $v_{ij} \in \mathcal{D}_i$ are computed, the same exercise can be repeated for all the values in $\mathcal{D}_i \setminus \{v_{ij}\}$ but, more importantly, all the other features.

Example 6. *Consider again Table 1. The two rules shown in Example 5 are computed by our rule learning approach if we focus on feature* Status*. The following two rules can be extracted when feature* Relationship *is focused on instead:*

- IF *Status* = *Married* ∧ *Sex* = *M.* THEN *Rel.* = *Husband*
- IF *Status* = *Married* ∧ *Sex* = *F.* THEN *Rel.* = *Wife*

As can be observed below, both Example 5 and Example 6 may be used to shorten explanations under certain conditions (see Section 4).

Duplicate Rules. As mentioned above, all rules generated with the MaxSAT approach of (Ignatiev et al. 2021) are guaranteed to be subset-minimal. Furthermore, none of the rules enumerated is symmetric with another rule if considered in the *if-then* form. However, when the rules are treated as clauses, i.e. a disjunction of Boolean literals, some rules may duplicate the other. Indeed, recall that a rule of size $k \leq |\mathcal{F}|$ is of the form $(f_1 \wedge \ldots \wedge f_{k-1}) \rightarrow f_k$ where each f_i represents a literal $(x_i = v_{i_{j_i}})$, $i \in \mathcal{F}$ and $v_{i_{j_i}} \in \mathcal{D}_i$. Clearly, this same proposition can be equivalently represented as a clause $(\neg f_1 \vee \ldots \vee \neg f_{k-1} \vee f_k)$. Observe that the same clause can be used to represent another rule $(f_1 \wedge \ldots \wedge f_{k-2} \wedge \neg f_k) \rightarrow \neg f_{k-1}$, which can thus be seen as symmetric in the *clausal form*. This way, a clause of size k represents k possible rules. However, due to symmetry, it suffices to compute only one of them and block all the "duplicates" by adding its clausal representation to the MaxSAT solver. This novel symmetry breaking mechanism significantly improves the scalability of our approach.

Example 7. *Consider a rule* {*IF Status* = *Married* ∧ *Sex* = *Male THEN Relationship* = *Husband*} *computed when compiling feature-value Relationship* = *Husband. This rule is represented as a clause*

(*Status* ≠ *Married* ∨ *Sex* ≠ *M.* ∨ *Relationship* = *Husband*)

There are two duplicates in other contexts:

- IF *Status* = *Married* ∧ *Rel.* ≠ *Husband* THEN *Sex* = *F.*
- IF *Sex* = *M.* ∧ *Rel.* ≠ *Husband* THEN *Status* ≠ *Married*

Extraction limit. Even if we remove duplicate rules, there can still be many rules to enumerate for an entire dataset. Many of them will never, or only rarely, contribute to reducing the size of explanations of the classifier. Extracting these *low value* rules is unnecessary in the rule extracting process. In practice, we noticed that some rules (e.g. long rules or rules having a low support) never contribute to explanation reduction. Hence, we apply an *extraction limit* to prevent exhaustive rule enumeration, which enables us to focus only on *most useful* rules. Here, extraction limit can be a restriction of a user's choice, e.g. a total extraction runtime, a limit on the number of rules, rule support or size, etc.

A high-level view on the overall rule extraction approach is provided in Algorithm 1. Initially, the class column from the original dataset $\mathcal{E}$ is dropped and the features $\mathcal{F}$ in $\mathcal{E}$ are acquired. For each feature $i \in \mathcal{F}$, the algorithm enumerates the decision rules targeting i until the extraction limit is met or no more rules can be found. The rules previously learned are blocked in the clausal form to avoid computing their duplicates. Finally, the algorithm returns the rules extracted.

Our approach computes only rules that are perfectly consistent with the *known* data, which makes sense if the data is extensive and trustworthy. In practical settings, however, some of the data are unknown, i.e. the rules computed may be inconsistent with unseen parts of the feature space $\mathbb{F}$. If testing and validation data are available, then the rules can be tested against them. We can then exclude the rules that are not *sufficiently* accurate wrt. test and/or validation data.

Algorithm 1: Rule Extraction

Input: Dataset $\mathcal{E}$, extraction limit λ
Output: Rules φ

$\mathcal{E}_f, \mathcal{F} \leftarrow \texttt{DropClass}(\mathcal{E}), \texttt{ExtractFeatures}(\mathcal{E})$
$\varphi, B \leftarrow \emptyset, \emptyset$ *# to extract and block rules, resp.*
for $i \in \mathcal{F}$ **do**
  for rule $\in \texttt{EnumerateRules}(\mathcal{E}_f, i, B)$ **do**
    if $\texttt{limit}$(rule, λ) is $true$ **then**
      break
    $\varphi \leftarrow \varphi \cup$ rule
  $B \leftarrow \varphi$
return φ

4 Knowledge-Assisted Explanations

In this section, we show how to apply background knowledge as additional constraints when computing a single formal abductive or contrastive explanation for an ML model prediction but also when enumerating them. We also show how to identify the rules that have been used when extracting formal explanations, which comes in handy when trustable explanations are of concern.

We assume the obtained background knowledge can be represented as a formula φ. Under that assumption, (Gorji and Rubin 2022) proposes to compute AXps for positive predictions of a Boolean classifier $\tau : \mathbb{F} \rightarrow \{0, 1\}$ taking into account constraints φ. Observe that formula φ can be seen as representing a predicate $\varphi : \mathbb{F} \rightarrow \{0, 1\}$, the truth value of which, i.e. $\varphi(\mathbf{x})$, can be tested for an instance $\mathbf{v} \in \mathbb{F}$. The approach of (Gorji and Rubin 2022) relies on *compiling* a Boolean classifier $\tau(\mathbf{x})$ into a *tractable* representation (Shih, Choi, and Darwiche 2018) and proposes to compute an AXp $\mathcal{X} \subseteq \mathcal{F}$ for prediction $\tau(\mathbf{v}) = 1$, $\mathbf{v} \in \mathbb{F}$, subject to constraints φ as a prime implicant of $[\varphi(\mathbf{x}) \rightarrow \tau(\mathbf{x}) = 1]$.

Observe that we can generalize this idea to the context of computing formal abductive and contrastive explanations for *any classifier* that admits a logical representation suitable for making reasoning oracle calls wrt. formulas (1) and (2). Namely, given a prediction $\tau(\mathbf{x}) = c$, $\mathbf{v} \in \mathbb{F}$, $c \in \mathcal{K}$, an abductive explanation $\mathcal{X} \subseteq \mathcal{F}$ subject to background knowledge φ is such that:

$$\forall(\mathbf{x} \in \mathbb{F}). \bigwedge\nolimits_{j \in \mathcal{X}} (x_j = v_j) \rightarrow [\varphi(\mathbf{x}) \rightarrow (\tau(\mathbf{x}) = c)] \quad (3)$$

More importantly, the same can be done with respect to contrastive explanations. Given a prediction $\tau(\mathbf{x}) = c$, $\mathbf{v} \in \mathbb{F}$, $c \in \mathcal{K}$, a contrastive explanation $\mathcal{Y} \subseteq \mathcal{F}$ subject to background knowledge φ is such that the following holds:

$$\exists(\mathbf{x} \in \mathbb{F}). \bigwedge\nolimits_{i \notin \mathcal{Y}} (x_i = v_i) \wedge [\varphi(\mathbf{x}) \wedge (\tau(\mathbf{x}) \neq c)] \quad (4)$$

Note that (3) and (4) are the negation of each other, i.e. a subset of features $\mathcal{Y} \subseteq \mathcal{F}$ *is* a CXp for prediction $\tau(\mathbf{x}) = c$ iff $\mathcal{X} = \mathcal{F} \setminus \mathcal{Y}$ *is not* an AXp. This means when dealing with either AXps or CXps, one can reason about (un)satisfiability of formula $\bigwedge_{i \in \mathcal{Z}} (x_i = v_i) \wedge [\varphi(\mathbf{x}) \wedge (\tau(\mathbf{x}) \neq c)]$ with $\mathcal{Z}$ being either $\mathcal{X}$ or $\mathcal{F} \setminus \mathcal{Y}$ depending on the kind of target explanation. Therefore, if background knowledge φ is a *conjunction* of constraints, e.g. rules, we can integrate them in the existing formal explainability setup of (Ignatiev, Narodytska, and Marques-Silva 2019a) with no additional overhead.

Following (Ignatiev et al. 2020) and applying the same arguments, an immediate observation to make is that in the presence of background knowledge, the minimal hitting set duality between AXps and CXps holds:

Proposition 1. *Let $\mathbf{v} \in \mathbb{F}$ be an instance such that $\tau(\mathbf{v}) = c$, $c \in \mathcal{K}$, and background knowledge φ is compatible with $\mathbf{v}$. Then any AXp $\mathcal{X}$ for prediction $\tau(\mathbf{v}) = c$ minimally hits any CXp for this prediction, and vice versa.*

Proposition 1 enables us to apply algorithms originally studied in the context of over-constrained systems (Bailey and Stuckey 2005; Liffiton and Sakallah 2008; Belov, Lynce, and Marques-Silva 2012; Marques-Silva et al. 2013; Mencia, Previti, and Marques-Silva 2015; Ignatiev et al. 2015; Liffiton et al. 2016; Bendík, Cerná, and Benes 2018) to explore all AXps and CXps for ML predictions. In particular, the existing explanation extraction and enumeration algorithms (Ignatiev, Narodytska, and Marques-Silva 2019a; Ignatiev et al. 2020) can be readily applied by taking into account background knowledge, as shown in (3) and (4).

Gorji et al. (Gorji and Rubin 2022) proved that subset-minimal AXps computed subject to additional constraints for Boolean classifiers tend to be smaller than their unconstrained "counterparts". The rationale is that when additional constraints are imposed, some of the features $i \in \mathcal{F}$ may be dropped from an AXp because the equalities $x_i = v_i$ falsify the constraints, i.e. they represent data instances that are *not permitted* by the constraints. Based on their result, the following generalization can be proved to hold:

Proposition 2. *Consider $\mathbf{v} \in \mathbb{F}$ such that $\tau(\mathbf{v}) = c$, $c \in \mathcal{K}$, and background knowledge φ is compatible with $\mathbf{v}$. Then for any subset-minimal AXp $\mathcal{X} \subseteq \mathcal{F}$ for prediction $\tau(\mathbf{v}) = c$, there is a subset-minimal AXp $\mathcal{X}' \subseteq \mathcal{F}$ for $\tau(\mathbf{v}) = c$ subject to background knowledge φ such that $\mathcal{X}' \subseteq \mathcal{X}$.*

Proof. First, observe that if (1) holds for a set $\mathcal{S}$ then (3) holds for $\mathcal{S}$ too. Let $\mathcal{X}$ be a subset-minimal AXp for $\tau(\mathbf{v}) = c$ with no knowledge of φ, i.e. (1) holds for $\mathcal{X}$. Thanks to the observation above, (3) also holds for $\mathcal{X}$. To make it subset-minimal subject to φ, we can apply linear search feature traversal (similar to the AXp extraction algorithm (Ignatiev, Narodytska, and Marques-Silva 2019a)) checking if any of the features of $\mathcal{X}$ can be dropped s.t. (3) still holds. The result subset-minimal set of features $\mathcal{X}'$ is the target AXp subject to knowledge φ. □

Remark 1. *Note that the opposite, i.e. that given AXp $\mathcal{X}'$ subject to background knowledge φ, there must exist a subset-minimal AXp $\mathcal{X} \supseteq \mathcal{X}'$ without background knowledge φ, in general does not hold. To illustrate a counterexample, consider a fully Boolean classifier $\tau : \{0, 1\}^3 \rightarrow \{0, 1\}$ on features $\mathcal{F} = \{a, b, c\}$, which returns 1 iff $(a + b + c) \geq 2$. Consider instance $\mathbf{v} = (1, 1, 0)$ classified as 1. Given knowledge $\varphi = (\neg c \rightarrow a) \wedge (\neg c \rightarrow b)$, a valid subset-minimal AXp is $\mathcal{X}' = \{c\}$. However, when discarding knowledge φ, the only subset-minimal AXp for $\mathbf{v}$ is $\mathcal{X} = \{a, b\} \not\supseteq \mathcal{X}'$.*

R_0:	IF	Status = Married	THEN	Target $\geq$ 50k
R_1:	ELSE IF	Sex = Male $\land$ Relationship $\neq$ Husband	THEN	Target $<$ 50k
R_{DEF}:	ELSE		THEN	Target $\geq$ 50k

Figure 2: A DL for selected examples of *adult* dataset.

Example 8. *Consider the DL in Figure 2. Given an instance* $\mathbf{v} = \{$*Education = Dropout, Status = Separated, Occupation = Service, Relationship = Not-in-Family, Sex = Male, Hours/w =* $\leq 40\}$*, the prediction enforced by* R_1 *is* $\leq 50k$ *and the AXp is* $\mathcal{X} = \{$*Status, Relationship, Sex*$\}$*. Let a single constraint* φ *be* $\{$*Sex* $=$ *Male* $\land$ *Relationship* $=$ *Not-in-Family* $\rightarrow$ *Status* $=$ *Separated*$\}$*. Feature 'Status' can be dropped because the constraint* φ *ensures it to be set to the "right value" if the other two features are set as required, and hence* R_0 *is guaranteed not to fire. Thus, we can compute a smaller AXp* $\mathcal{X}' = \{$ *Relationship, Sex* $\}$*.*

While using background knowledge φ pays off in terms of interpretability of abductive explanations, this cannot be said wrt. contrastive explanations. Surprisingly and as the following result proves, background knowledge can only contribute to increase the size of contrastive explanations.

Proposition 3. *Consider* $\mathbf{v} \in \mathbb{F}$ *such that* $\tau(\mathbf{v}) = c$*,* $c \in \mathcal{K}$*, and background knowledge* φ *is compatible with* $\mathbf{v}$*. Then for any subset-minimal CXp* $\mathcal{Y}' \subseteq \mathcal{F}$ *for prediction* $\tau(\mathbf{v}) = c$ *subject to knowledge* φ*, there is subset-minimal CXp* $\mathcal{Y} \subseteq \mathcal{F}$ *is a CXp for prediction* $\tau(\mathbf{v}) = c$ *such that* $\mathcal{Y}' \supseteq \mathcal{Y}$*.*

Proof. First, observe that if (4) holds for a set $\mathcal{S}$ then (2) holds for $\mathcal{S}$ too. Let $\mathcal{Y}'$ be a subset-minimal CXp subject to background knowledge φ, i.e. (4) holds for $\mathcal{Y}'$. By the observation made above, (2) also holds for $\mathcal{Y}'$. Now, by applying linear search dropping features of $\mathcal{Y}'$ and checking (2) (Ignatiev, Narodytska, and Marques-Silva 2019a), one can get a subset-minimal $\mathcal{Y} \subseteq \mathcal{Y}'$ wrt. (2), i.e. $\mathcal{Y}$ is a subset-minimal CXp. □

Remark 2. *Note that the reverse direction: given a CXp* $\mathcal{Y}$ *generated without using background knowledge, there must exists a CXp* $\mathcal{Y}' \supseteq \mathcal{Y}$ *using background knowledge, does not hold. Consider a classifier on Boolean features* $\mathcal{F} = \{a, b, c\}$ *which returns the parity ODD, EVEN of* $a + b + c$*. Consider background knowledge* $a = b$*. Now* $\mathcal{Y} = \{a\}$ *is a CXp for* $\tau(1, 1, 1) =$ *ODD without using background knowledge supported by instance* $\tau(0, 1, 1) =$ *EVEN. But this does not agree with the background knowledge. The only CXp using the background knowledge is* $\{c\}$*, because* a *and* b *must change together they never affect the parity. However and as our experimental results confirm, in practice these examples do not arise, as we always find a CXp using background knowledge that extends a CXp without background knowledge.*

One may wonder then why background knowledge is useful when computing CXps. The reason is that the CXps generated using background knowledge are *correct* under the assumption that the background knowledge describes the actual relationships between features. On the contrary, CXps generated without using background knowledge are *only correct* under the assumption that every combination of feature values is possible, i.e. all features are independent and their values are uniformly distributed across the feature space, which hardly ever occurs in practice.

Example 9. *Consider the setup of Example 8. Observe that a CXp for the prediction is* $\mathcal{Y} = \{$ *Status* $\}$*. Its correctness relies on the fact that changing Status to Married changes the prediction to* $\geq 50k$*. But given the background knowledge* φ*, this is clearly erroneous. Since the other fixed features in instance* $\mathbf{v}$ *are* $\{$ *Sex* $=$ *Male, Education* $=$ *Dropout, Occupation* $=$ *Service, Relationship* $=$ *Not-in-family, Hours/w* $= \leq 40$ $\}$*, the modification is inconsistent with knowledge* φ*. This demonstrates the weakness of CXps as they rely on the assumption that any tuple of feature values in* $\mathbb{F}$ *is possible. Applying constraint* φ *leads to a larger CXp* $\mathcal{Y}' \triangleq \{$ *Status, Relationship* $\}$*. This clearly does allow the prediction to change and it is compatible with* φ*.*

Algorithm 2: Determine Background Knowledge Used

Input: Classifier τ, instance $\mathbf{v}$, prediction $c = \tau(\mathbf{v})$, constraints φ, AXp $\mathcal{X}'$
Output Used rules: $\varphi_u \subseteq \varphi$

$\varphi_u \leftarrow \varphi$
if `Entails`$(\mathcal{X}', \tau, \mathbf{v}, c, \emptyset)$ **then**
  return $\emptyset$
for $r \in \varphi$ **do**
  if `Entails`$(\mathcal{X}', \tau, \mathbf{v}, c, \varphi_u \setminus \{r\})$ **then**
    $\varphi_u \leftarrow \varphi_u \setminus \{r\}$
return φ_u

4.1 Attributing Responsibilities to Knowledge

Since the computed background knowledge is not always useful, e.g. the extracted rules may not necessarily contribute to smaller AXps, we introduce an approach to discovering which of the rules are used to reduce an explanation. Using this approach, we can observe and measure the effect of the value of extraction limit discussed in Section 3, e.g. the size limit of 5 can be considered as a reasonable extraction limit if the size of most of the *useful rules* is no more than 5. A further usage is that when providing a user with an explanation, we can expose which background knowledge was used to generate the explanation. This enables the user to assess the quality of the rules used and decide whether they trust or disagree with the background knowledge.

Given a knowledge-assisted AXp $\mathcal{X}'$ for prediction $\tau(\mathbf{v}) = c$ and background knowledge φ, Algorithm 2 reports a subset of rules φ *responsible* for the explanation $\mathcal{X}'$. The algorithm makes use of a number of calls to `Entails`, which is meant to be a call to reasoning oracle deciding the validity of formula (3) subject to background knowledge specified as the *final parameter*. First, we check if $\mathcal{X}'$ satisfies the AXp condition (1) with no knowledge given. If this is the case, then no rules are used when computing $\mathcal{X}'$ and so the algorithm returns $\emptyset$. Otherwise, the algorithm proceeds by considering each rule $r \in \varphi$ one by one and checking if condition (3) holds for background knowledge $\varphi \setminus \{r\}$ (see line 5). If it does then rule r can be dropped; otherwise, it is necessary for AXp $\mathcal{X}'$ and is thus kept. This simple linear search procedure ends up identifying a subset-minimal set of rules φ_u that are responsible for abductive explanation $\mathcal{X}'$.

Note that a similar algorithm can be outlined for identifying background knowledge useful when computing contrastive explanations. In that case, instead of making calls to `Entails`, one would need to make calls to a reasoner deciding the validity of formula (4) subject to a varying set of background constraints φ_u.

5 Experimental Results

This section assesses our approach to extracting background knowledge wrt. popular association rule mining algorithms and the quality of the enumerated AXps and CXps with background knowledge for 3 different ML models: DLs, BTs, and BNNs. Finally, this section applies the background knowledge identified for evaluating correctness of the explanations produced by some heuristic ML explainers.

Setup and Prototype Implementation. All the experiments were run on an Intel Xeon 8260 CPU running Ubuntu 20.04.2 LTS, with a memory limit of 8 GByte. A prototype of the proposed approach to extracting background knowledge and computing AXps and CXps applying background knowledge was developed as a set of Python scripts.[3] The implementation of knowledge extraction builds on (Ignatiev et al. 2021) and extensively uses state-of-the-art SAT technology (Ignatiev, Morgado, and Marques-Silva 2018, 2019). Also, the implementation of explanation enumeration for DLs and BNNs makes use of the SAT technology (Ignatiev, Morgado, and Marques-Silva 2018) while for BTs we apply modern SMT solvers (Gario and Micheli 2015).

A few words should be said about the competition considered. First, we compared our knowledge extraction approach to the Apriori and Eclat algorithms (Agrawal and Srikant 1994; Zaki et al. 1997). In our experiments, these algorithms behave almost identically with Eclat solving one more instance; as a result, we use Eclat as the best competitor. When running Eclat, we apply the same setup as used for our approach. Finally, heuristic explainers are represented by LIME (Ribeiro, Singh, and Guestrin 2016), SHAP (Lundberg and Lee 2017), and Anchor (Ribeiro, Singh, and Guestrin 2018) in their default configurations.

Datasets. The benchmarks considered include a selection of datasets publicly available from UCI Machine Learning Repository (Dua and Graff 2017) and Penn Machine Learning Benchmarks (Olson et al. 2017). In total, 24 datasets are selected. Whenever applicable, numeric features in all benchmarks were quantized into 4, 5, or 6 intervals. Therefore, the total number of quantized datasets considered is 62.

Machine Learning Models. We used CN2 (Clark and Niblett 1989) to train the DL models studied. BTs were computed by XGBoost (Ribeiro, Singh, and Guestrin 2016) s.t. each class is represented by 25 trees of depth 3. BNNs were trained by PyTorch (Paszke et al. 2019). Three configurations of hidden layers[4] were used when training BNNs to achieve sufficient test accuracy. As usual, each of the 62 datasets was randomly split into 80% of training and 20% of test data, respectively. The average test accuracy of the DL, BT, and BNN models was 76.47%, 76.17%, and 80.31%.

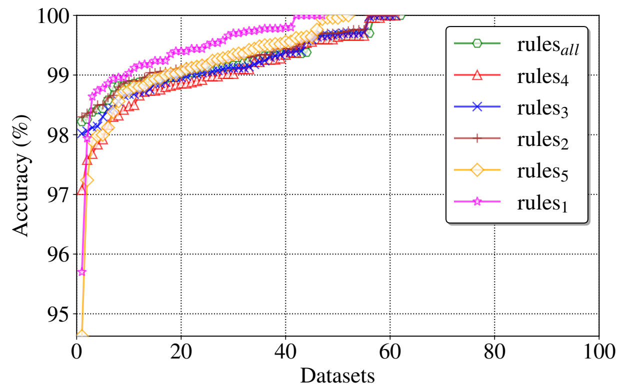
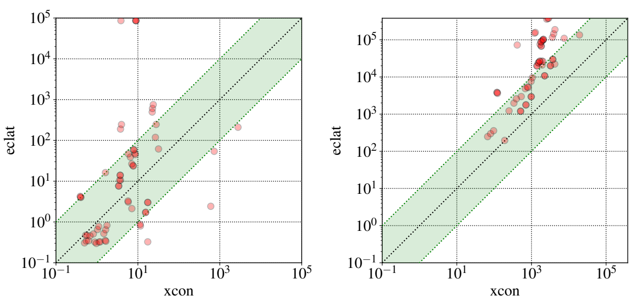


Figure 3: Accuracy of rules extracted by *xcon*.

Table 2: Average accuracy of individual rules over test data.

	rule$_1$	rule$_2$	rule$_3$	rule$_4$	rule$_5$	rule$_{all}$
Accuracy (%)	99.22	99.35	99.29	99.16	99.06	99.08



(a) Runtime comparison. (b) Number of rules comparison.

Figure 4: *Eclat* vs. *xcon* – performance comparison.

5.1 Knowledge Extraction

Knowledge extraction is tested using 5-fold cross validation. The average accuracy of each rule is measured as the proportion of test instances that violate that rule, averaged over the folds. We consider rules of length 1 to 5, and extracting all possible rules (*all*). Figure 3 and Table 2 compare the average accuracy of the background knowledge extracted. As can be observed, the average accuracy of *all rules* and $rules_s$, $s \in \{1, \ldots, 5\}$, exceeds 99%.

Additionally, we compare the overall performance of exhaustive rule extraction against rule extraction with the size

[3]The implementation as well as all datasets and logs of our experiments is available at https://github.com/jinqiang-yu/xcon22.

[4]The 3 configurations are classified as *small*, *medium* and *large*. The size of the hidden layers of these 3 configurations is as follows: large: (64, 32, 24, 2); medium: (32, 16, 8, 2); small: (10, 5, 5, 2).

limit 5. On average, exhaustive (limited, resp.) rule enumeration ends up computing 2116.29 (1964.24, resp.) rules per dataset. According to our results, our approach is quite efficient and for the lion's share of datasets (59 out of 62) both exhaustive and limited enumeration finish within 30 seconds; for the 3 remaining datasets, limited enumeration is a bit faster but both approaches finished rule enumeration within 3000 seconds.

In the remainder of this section, we compare *xcon* against Eclat in terms of the overall performance. For a fair comparison, we set Eclat to extract only rules of confidence 100%, i.e. all the rules extracted are perfectly consistent with the *known* data.

Scalability. Figure 4a demonstrates that *xcon* can extract rules faster or on par with Eclat in the vast majority of the considered datasets. Moreover, Eclat can only extract rules for the train-test 5-fold pairs of 58 (out of 62) datasets, while *xcon* is able to extract rules for all the considered datasets.

Rule Number. Figure 4b depicts the comparison of the number of extracted rules in the 58 datasets solved by both approaches. Observe that Eclat extracts more rules than *xcon* because it uses a less expressive language for the feature literals, i.e. it cannot extract rules containing the *negation* of a feature-value pair. For example, assume *xcon* can extract a rule [IF $x_1 \neq 0$ THEN $x_2 = 1$] given features 1 and 2 and their domains $\mathcal{D}_1 = \mathcal{D}_2 = \{0, 1, 2\}$. In this case, Eclat is unable to extract the above rule – instead, it has to extract two distinct rules to represent the same information, i.e. [IF $x_1 = 1$ THEN $x_2 = 1$] but also [IF $x_1 = 2$ THEN $x_2 = 1$].

5.2 Knowledge-Assisted Explanations

This section evaluates the proposed approach to computing formal explanations for DLs, BTs, and BNNs, where the computed background knowledge was applied. In particular, we evaluate the runtime of explanation enumeration, explanation size, as well as the portion of background knowledge used when computing explanations. Note that here we consider only the rules of size at most 5, which is shown to be a reasonable value in Section 5.3.

For each of the 62 datasets, we selected all *test* instances and enumerated 20 *smallest size* AXps or CXps for each such instance. Hereinafter, $xcon_*$ s.t. $* \in \{dl, bt, bnn\}$ denotes the proposed approach applied for explaining DL, BT, and BNN models, respectively. Furthermore, a superscripted version $xcon^r_*$ is used to denote the configurations that apply background knowledge.

Scalability. The scatter plots in Figures 5a, 5b, 6a, 6b, 7a and 7b depict the comparison of the average runtime of computing a single AXp or CXp for an instance (taken across all the 20 explanations computed) between $xcon_*$ and $xcon^r_*$. Clearly, for all the 3 models, the use of background knowledge significantly improves the performance of AXp extraction (see Figures 5a, 6a, and 7a). At first glance, the performance of CXp extraction deteriorates significantly in the case of DLs (Figure 5b) if compared to the other two models. We should say that this impression is caused by a different scaling used in Figure 5b — observe that CXp extraction is 1–3 orders of magnitude faster for DLs than for the other 2 models, both when applying and not applying background knowledge. Also, this can be explained by the fact that the average CXp size in the case of DLs increases tremendously, which leads to a much larger number of reasoning oracle calls when computing an explanation. For BTs and BNNs, the use of background knowledge neither improves nor degrades the computation of CXps (Figures 6b and 7b), even though an increase of CXp size can be also observed.

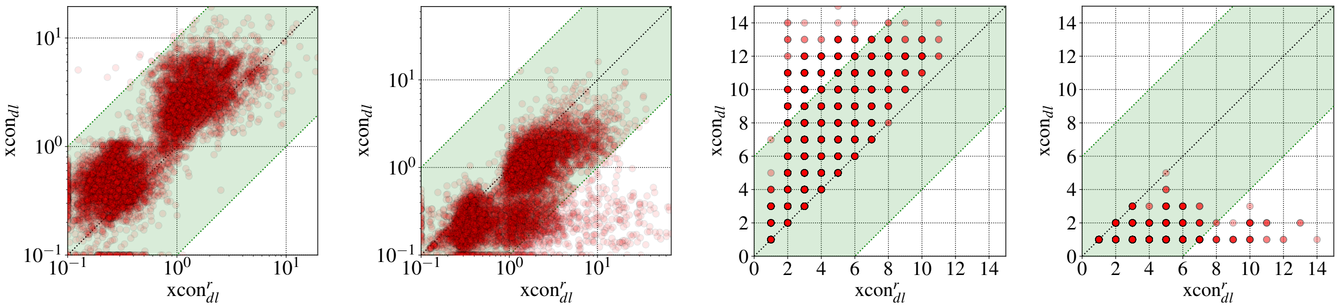
Table 3: Change of average minimum explanation size.

Dataset	Feats	Model	AXp Size		CXp Size	
			Before	After	Before	After
adult	65	DL	7.46	3.65	1.00	1.60
		BT	5.02	2.84	1.10	2.13
		BNN	7.51	3.00	1.40	2.15
compas	16	DL	5.65	3.74	1.01	1.15
		BT	3.91	3.09	1.06	1.15
		BNN	4.40	2.79	1.19	1.30
lending	35	DL	5.30	4.30	1.00	1.41
		BT	1.99	1.80	1.00	2.04
		BNN	4.36	2.49	1.35	1.90
recidivism	29	DL	9.51	5.58	1.00	1.23
		BT	6.04	4.04	1.17	1.67
		BNN	7.01	4.01	1.42	1.82

Explanation Quality. The change of smallest size of AXps and CXps in an instance is shown in Figure 5c, 5d, 6c, 6d, 7c, and 7d. As can be seen in Figure 5c,6c and 7c, our results demonstrate how background knowledge, if present, contributes to AXp size reduction across all models. In particular, in many cases the size of a smallest AXp drops from 14 to 2, from 11 to 2, and from 17 to 2, for DLs, BTs, and BNNs, respectively. In contrast to AXps, Figures 5d, 6d and 7d illustrate that the size of smallest CXps is increased when background knowledge is applied. Namely, in a number of cases the size of a smallest CXp jumps from 1 to 14, from 1 to 15, and from 2 to 13 for DLs, BTs, and BNNs, respectively. These results exemplify how easy it is to flip the prediction when no background knowledge is present illustrating the potential correctness issues for the corresponding CXps.

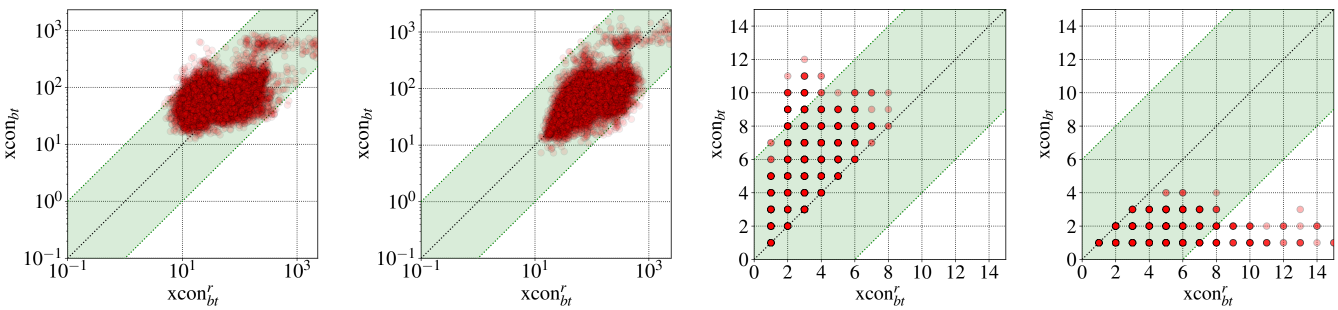
Table 3 details the change of the average size of smallest AXps and CXps computed without or with background knowledge for DLs, BTs and BNNs and for a selection of 4 publicly available datasets: *adult*, *compas*, *lending* and *recidivism*, which were previously studied in the context of heuristic and formal explanaitions. (Here, all numeric features, if any, are quantized into 6 intervals.) Note that Table 3 confirms the general observations made that background knowledge triggers smaller AXps but larger CXps for all the models studied. The average size of smallest AXps in *adult* and *recidivism* drops by around 4 for DLs and BNNs, while the average smallest CXp size slightly increases for the two models. In *compas*, the size of smallest AXps in the three models decreases by 1–2 and the size of smallest CXps subtly increases. The size of smallest AXps in *lending* drops by 1.00 in DLs, 0.19 in BTs, and 1.87 in BNNs.

On Eclat-Assisted Explanations. The experiment above



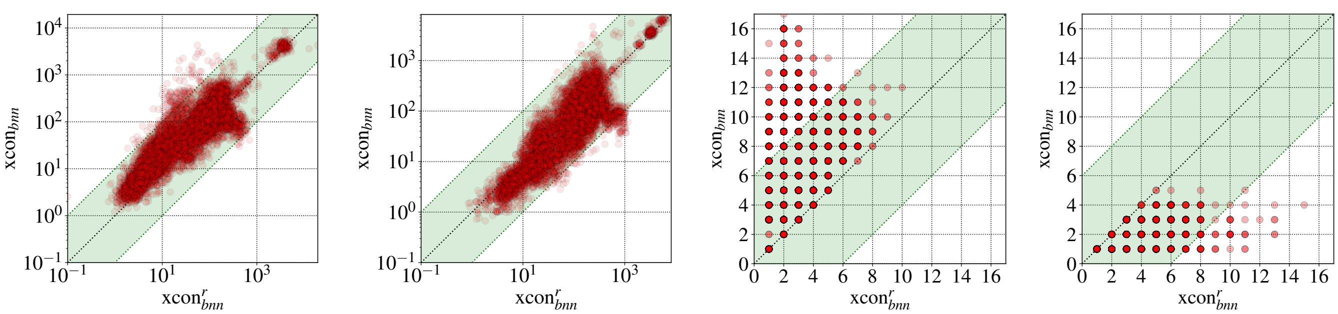
(a) AXp runtime for DLs. (b) CXp runtime for DLs. (c) AXp size for DLs. (d) CXp size for DLs.

Figure 5: Impact of *xcon* rules on runtime (ms) and explanation size for DLs.



(a) AXp runtime for BTs. (b) CXp runtime for BTs. (c) AXp size for BTs. (d) CXp size for BTs.

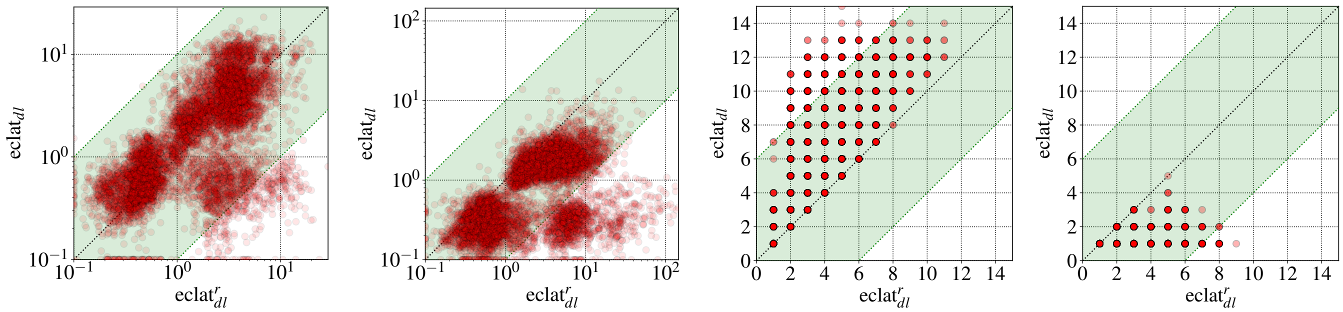
Figure 6: Impact of *xcon* rules on runtime (ms) and explanation size for BTs.



(a) AXp runtime for BNNs. (b) CXp runtime for BNNs. (c) AXp size for BNNs. (d) CXp size for BNNs.

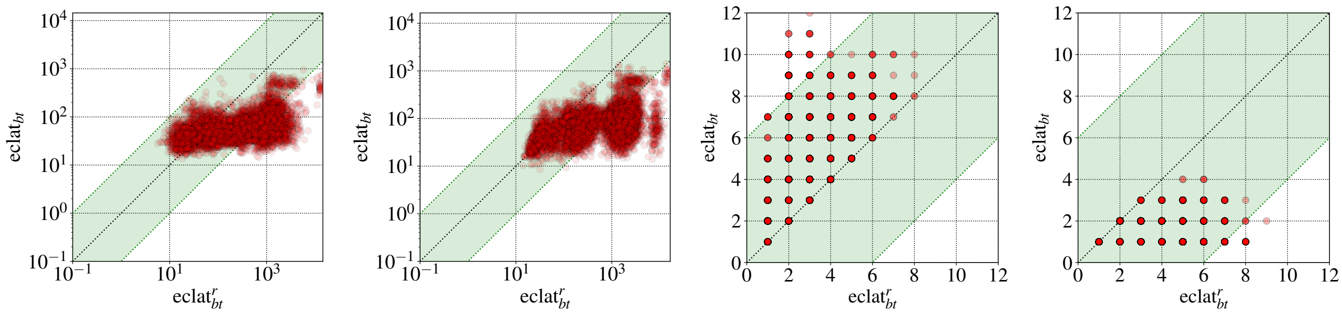
Figure 7: Impact of *xcon* rules on runtime (ms) and explanation size for BNNs.

was repeated for the background knowledge extracted with the use Eclat and its results are detailed below. Namely, Figure 8, Figure 9 and Figure 10 evaluate the proposed approach to computing formal explanations for DLs, BTs, and BNNs, taking into account the rules mined by Eclat. Note that here only 58 datasets tackled by Eclat are considered. Similar to $xcon_*$ above, $eclat_* \in \{dl, bt, bnn\}$ represents the formal explanation approach applied to DL, BT, and BNN models, respectively. Moreover, $eclat^r_*$ is used to denote the configurations that apply background knowledge extracted by Eclat. Scalability-wise and in contrast to the case of *xcon*, where the performance of AXps computation improves and the performance of CXp computation degrades in the presence of background knowledge extracted by the MaxSAT approach, the use of Eclat-provided background knowledge degrades the performance of CXp generation for DLs as well as both AXp and CXp computation for BTs. This can be explained by the larger number of rules extracted by Eclat compared to the MaxSAT approach. In terms of the quality of explanations, observations similar to the case of *xcon* can be made, i.e. background knowledge extracted by Eclat can trigger AXp size reduction across all the 3 models, while the size of CXps increases due to the background knowledge.



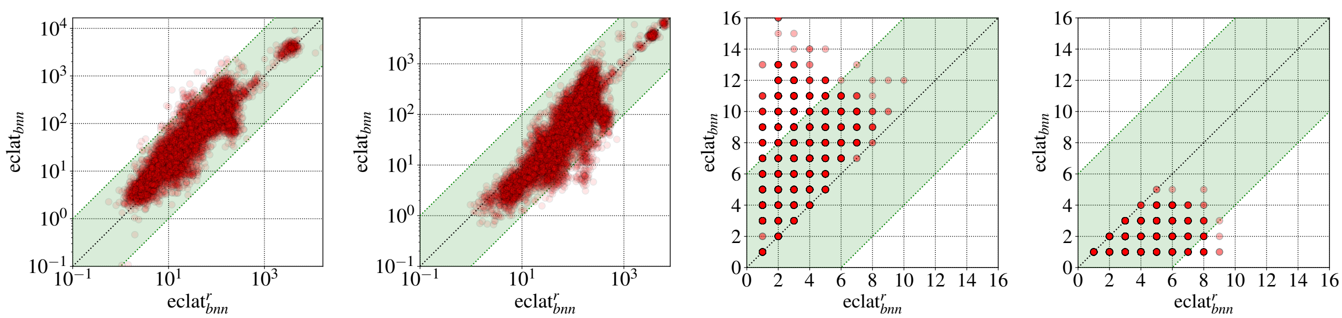
(a) AXp runtime for DLs. (b) CXp runtime for DLs. (c) AXp size for DLs. (d) CXp size for DLs.

Figure 8: Impact of Eclat rules on runtime (ms) and explanation size for DLs.



(a) AXp runtime for BTs. (b) CXp runtime for BTs. (c) AXp size for BTs. (d) CXp size for BTs.

Figure 9: Impact of Eclat rules on runtime (ms) and explanation size for BTs.



(a) AXp runtime for BNNs. (b) CXp runtime for BNNs. (c) AXp size for BNNs. (d) CXp size for BNNs.

Figure 10: Impact of Eclat rules on runtime (ms) and explanation size for BNNs.

5.3 Usefulness of Background Knowledge

To assess rules' contribution into explanation extraction, we applied the setup of Section 5.2, i.e. we enumerated at most 20 smallest size AXps for each test instance.[5] Table 4 presents the evaluation of which rules contribute to AXp size reduction for DLs, BTs, and BNNs for the same selection of datasets studied in Table 3, i.e. *adult*, *compas*, *lending* and *recidivism*. However and in contrast to the previous experiment, the rules here are exhaustively extracted for each of the datasets, i.e. *no extraction limit* is applied. This resulted in extracting rules up to size 7.

[5]This experiment is conducted only for the proposed MaxSAT-based approach for knowledge extraction.

Our experimental results indicate that rules of size greater than 5 are not frequently used when computing AXps. Table 4 shows that the size of more than 98% of the useful rules in the three models for *compas* ranges from 1 to 4. For *adult*, more than 95% of the useful rules comprise 1 to 5 literals. The rules of size 5 are significant in the case of *lending* and *recidivism*, where more than 20% of the useful rules contain 5 literals. However, there are 29.0%, 29.1%, and 26.1% of the useful rules larger than size 5 for *recidivism* in DLs, BT, and BNNs, respectively, while less than 11% of the useful rules contain more than 5 literals for the other 3 datasets.

Table 4: Size distribution of used rules.

Dataset	Feats	Model	Distribution (%) 1	2	3	4	5	6	7
adult	65	DL	10.9	17.2	37.7	21.9	8.4	3.1	0.8
		BT	7.3	10.0	39.5	30.2	10.1	2.4	0.4
		BNN	9.8	11.5	39.6	26.8	9.0	2.7	0.5
compas	16	DL	55.4	17.4	22.3	3.3	0.2	1.4	–
		BT	53.2	29.0	16.1	1.0	0.1	0.6	–
		BNN	41.4	27.3	27.2	2.9	0.9	0.3	–
lending	35	DL	43.4	4.1	3.3	18.7	20.2	9.3	1.1
		BT	41.7	7.6	4.5	13.3	23.2	9.1	0.7
		BNN	36.2	3.6	3.5	21.3	24.6	9.5	1.2
recidivism	29	DL	2.9	1.5	9.1	25.8	28.6	20.3	8.7
		BT	2.1	1.5	8.1	25.8	30.9	20.7	8.4
		BNN	1.6	1.4	7.5	24.1	36.6	18.7	7.4

Table 5: Average runtime per explanation.

Model	Runtime per explanation (ms)						
	xcon_{axp}	xcon^r_{axp}	xcon_{cxp}	xcon^r_{cxp}	LIME	SHAP	Anchor
DL	2	1	1	2	3755	42 555	3800
BT	80	82	97	151	98	6	351
BNN	196	152	179	199	15 607	183 058	11 384

These results support our choice of value 5 as the extraction limit since the size of the vast majority of useful rules is no more than 5.

5.4 Formal vs. Heuristic Explanations

Following (Ignatiev, Narodytska, and Marques-Silva 2019a; Narodytska et al. 2019; Ignatiev 2020), we apply formal explanations to assess the runtime and explanation quality for the heuristic approaches LIME, SHAP, and Anchor. The idea is to show the importance of trustable background knowledge when targeting a more accurate quality assessment.

Scalability. Figure 11 and Table 5 illustrate the runtime of a single explanation extraction for a data instance across the 62 datasets performed by LIME, SHAP, Anchor, $xcon_*$, and $xcon^r_*$. Here, $xcon^r_*$ and $xcon_*$ represent the proposed approach to computing AXps or CXps with/without background knowledge, s.t. $* \in \{axp, cxp\}$. Observe that both $xcon_*$ and $xcon^r_*$ outperform LIME and Anchor for all the 3 models, explaining a data instance in a fraction of a second. LIME and Anchor are 1-2 orders of magnitude slower for DL and BNN models, while LIME outweighs Anchor when generating explanations for BTs. The worst performance for DL and BNN models is demonstrated by SHAP while, surprisingly, SHAP outperforms the other competitors for BTs models.

Correctness. The average correctness of computed explanations[6] is shown in Figure 12 and Table 6. Here, an explanation is said to be correct if it answers a *"why"* question and satisfies (1) (or (3) in the presence of background knowledge) or it answers a *"why not"* question and satisfies (2) (or (4) in the presence of background knowledge). The superscripted notation $lime^r_*$, $shap^r_*$, and $anchor^r_*$ is used to denote

[6]LIME/SHAP assign weights to *all the features*. We use only those whose weight contributes to the decision made based on sign.

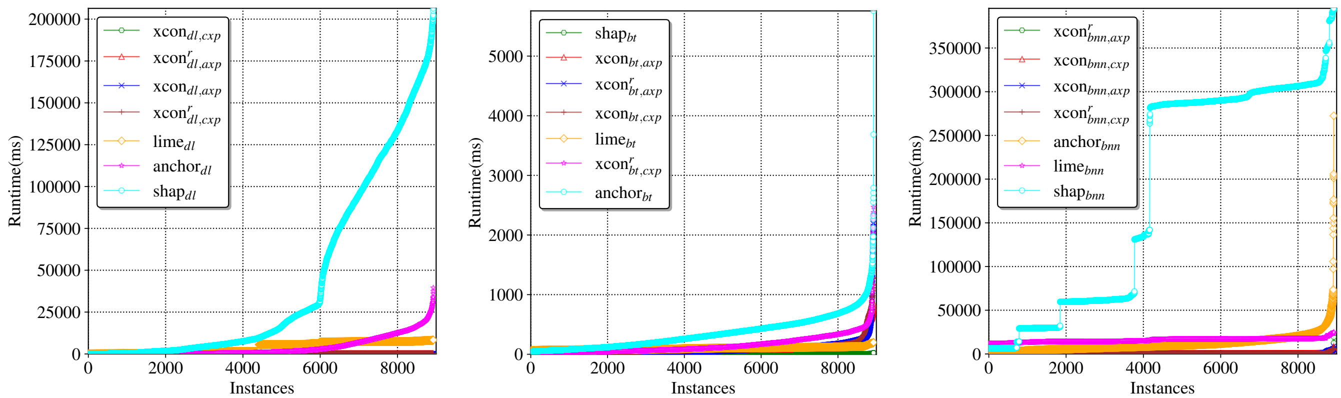
Table 6: Average correctness of LIME, SHAP and Anchor.

Explainer	Correctness (%)					
	Without knowledge			With knowledge		
	DL	BT	BNN	DL	BT	BNN
LIME	6.06	38.26	8.2	31.06	60.63	47.88
SHAP	49.47	72.89	58.89	91.72	93.75	95.0
Anchor	24.03	13.85	6.57	73.85	73.0	70.1

the fact that background knowledge is applied when evaluating correctness of the explanations produced by LIME, SHAP, and Anchor, respectively. Figure 12 and Table 6 show that the average correctness is higher when background knowledge is applied as the number of features required in a minimal correct explanation answering a *"why"* question can drop, which is demonstrated in Section 5.2. However, heuristic approaches are not able to achieve 100% correctness in the majority of the datasets. The best results are demonstrated by SHAP in both Figure 12a and Figure 12b. SHAP's explanations for most of the datasets achieve 40% correctness when no background knowledge applied, while its correctness jumps to 80% for the vast majority of datasets when background knowledge is taken into account. As of LIME and Anchor, without background knowledge, the correctness of most of the explanations is less than 20% for $anchor_{bt}$, $lime_{bnn}$, $anchor^r_{bnn}$, and $lime_{dl}$, but the correctness dramatically increases when background knowledge is applied. Figure 12b demonstrates that with background knowledge the best correctness is achieved by SHAP, followed by Anchor, where the major correctness for SHAP, Anchor, and LIME is more than 80%, 60% and 40%, respectively.

Heuristic explainers consistently demonstrate low correctness when no background knowledge is applied, which confirms the earlier results of (Ignatiev, Narodytska, and Marques-Silva 2019c; Ignatiev 2020). However, the situation changes dramatically when we apply the background knowledge. This is because some of the counterexamples invalidating heuristic explanations are forbidden by the knowledge extracted. Assuming that this knowledge is valid, these correctness results better reflect the reality and so are more trustable.

Explanation quality. Although explanation correctness dramatically increases when background knowledge is used, large size of correct explanations can render them uninterpretable. In this experiment, we evaluate the size of correct explanations computed by LIME, SHAP, and Anchor, and check how far those correct explanations are from their subset-minimal counterparts. Concretely, given a correct heuristic explanation computed either by LIME, or SHAP, or Anchor, we apply the formal approach to reduce it further, with or without background knowledge. Then we contrast the size of correct explanations and their corresponding size-minimal correct explanations for DL, BT, and BNN models. The comparison is detailed in the scatter plots of Figure 13, 14, and 15. As can be observed, a vast majority of correct explanations computed by LIME, SHAP, and Anchor are not minimal. Their size significantly exceeds the size of subset-

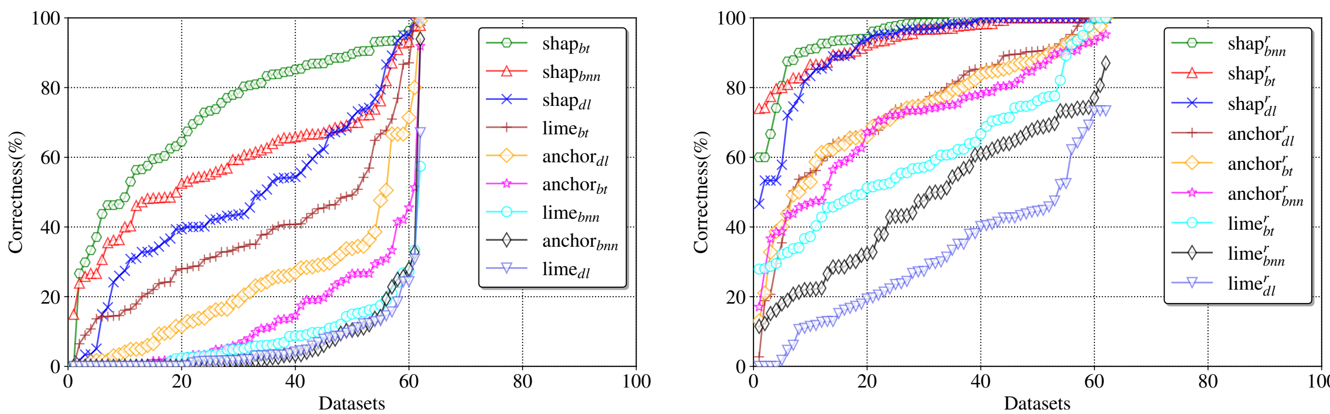


(a) Runtime for DLs

(b) Runtime for BTs

(c) Runtime for BNNs

Figure 11: Runtime (ms) of the considered explainers per explanation for DLs, BTs and BNNs.



(a) Correctness without background knowledge.

(b) Correctness with background knowledge.

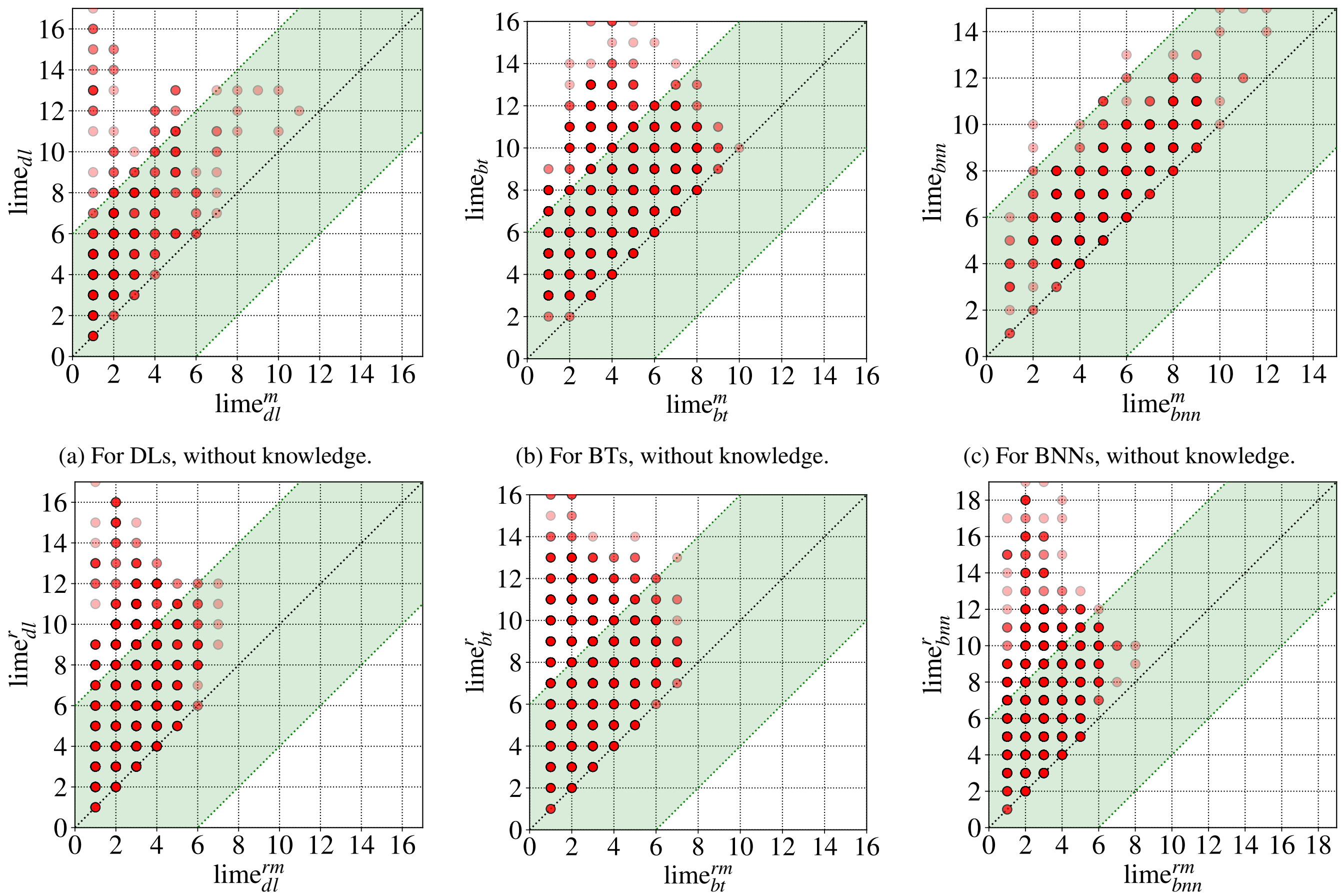
Figure 12: Correctness of heuristic explanations.

minimally reduced explanations. Furthermore, the size difference increases when background knowledge is available, which is in line with our earlier observations regarding the AXp computation.

6 Related Work

Many methods for extracting knowledge from a dataset of rules exist (Hipp, Güntzer, and Nakhaeizadeh 2000; Zhang and Zhang 2002; Agrawal and Srikant 1994; Zaki et al. 1997; Izza et al. 2020; Belaid, Bessiere, and Lazaar 2019). For use as background knowledge, we aim at very high confidence in the rules, ideally they should be *completely* valid for the feature space. While this is impossible to guarantee, the approach we define only generates rules, which are valid for the entire data used for rule generation. We can then use a validation or test set to remove rules that are not supported by the larger data. Traditional rule mining is more interested in rules with high support and less focused on validity, although it can be adapted to this case (see our experimental results above). Although the explanation methods we apply in the presence of background knowledge are agnostic about where it comes from, the motivation for our rule extraction method is twofold: (1) the rules are computed in a clausal form and (2) their high quality is guaranteed by the use of the strict optimization problem formulation.

The most prominent approaches to post-hoc explainability are of heuristic nature (Ribeiro, Singh, and Guestrin 2016; Lundberg and Lee 2017; Ribeiro, Singh, and Guestrin 2018) and based on sampling in the vicinity of the instances being explained. None of these approaches can handle background knowledge. Furthermore, they are susceptible to out-of-distribution attacks (Slack et al. 2020). Approaches to formal explainability are represented by compilation of classifiers into tractable representations (Shih, Choi, and Darwiche 2018) and reasoning-based explanation approaches (Ignatiev, Narodytska, and Marques-Silva 2019a; Marques-Silva and Ignatiev 2022). The closest related work is (Gorji and Rubin 2022). Based on compilation of a binary classifier into a binary decision diagram (BDD), it conjoins concocted background knowledge to give more succinct *"why"* explanations for the classifier. This approach is restricted to much smaller examples than we con-



(a) For DLs, without knowledge.

(b) For BTs, without knowledge.

(c) For BNNs, without knowledge.

(d) For DLs, with background knowledge.

(e) For BTs, with background knowledge.

(f) For BNNs, with background knowledge.

Figure 13: Size of LIME explanations for DLs, BTs and BNNs.

sider here, since the compilation of a classifier into a BDD tends to explode with the feature space. The SAT and SMT based approaches to explanation we use are far more scalable. Finally, we consider a much broader class of classifiers, and also examine *"why not"* explanations and how they can be improved by using background knowledge.

7 Conclusions

Using background knowledge is highly advantageous for producing formal explanations of machine learning models. For abductive explanations (AXps), the use of background knowledge substantially shortens explanations, making them easier to understand, *and* improves the speed of producing explanations. For contrastive explanations (CXps), while the background knowledge lengthens them and may increase the time required to generate an explanation, the resulting explanations are far more useful since they do not rely on the (usually unsupportable) assumption that all tuples in the feature space are possible. Furthermore and as this paper shows, background knowledge can be applied in the context of heuristic explanations when an accurate analysis of their correctness is required.

Acknowledgments

This research was partially funded by the Australian Government through the Australian Research Council Industrial Transformation Training Centre in Optimisation Technologies, Integrated Methodologies, and Applications (OPTIMA), Project ID IC200100009. This work was also partially supported by the AI Interdisciplinary Institute ANITI, funded by the French program "Investing for the Future – PIA3" under Grant agreement no. ANR-19-PI3A-0004, and by the H2020-ICT38 project COALA "Cognitive Assisted agile manufacturing for a Labor force supported by trustworthy Artificial intelligence".

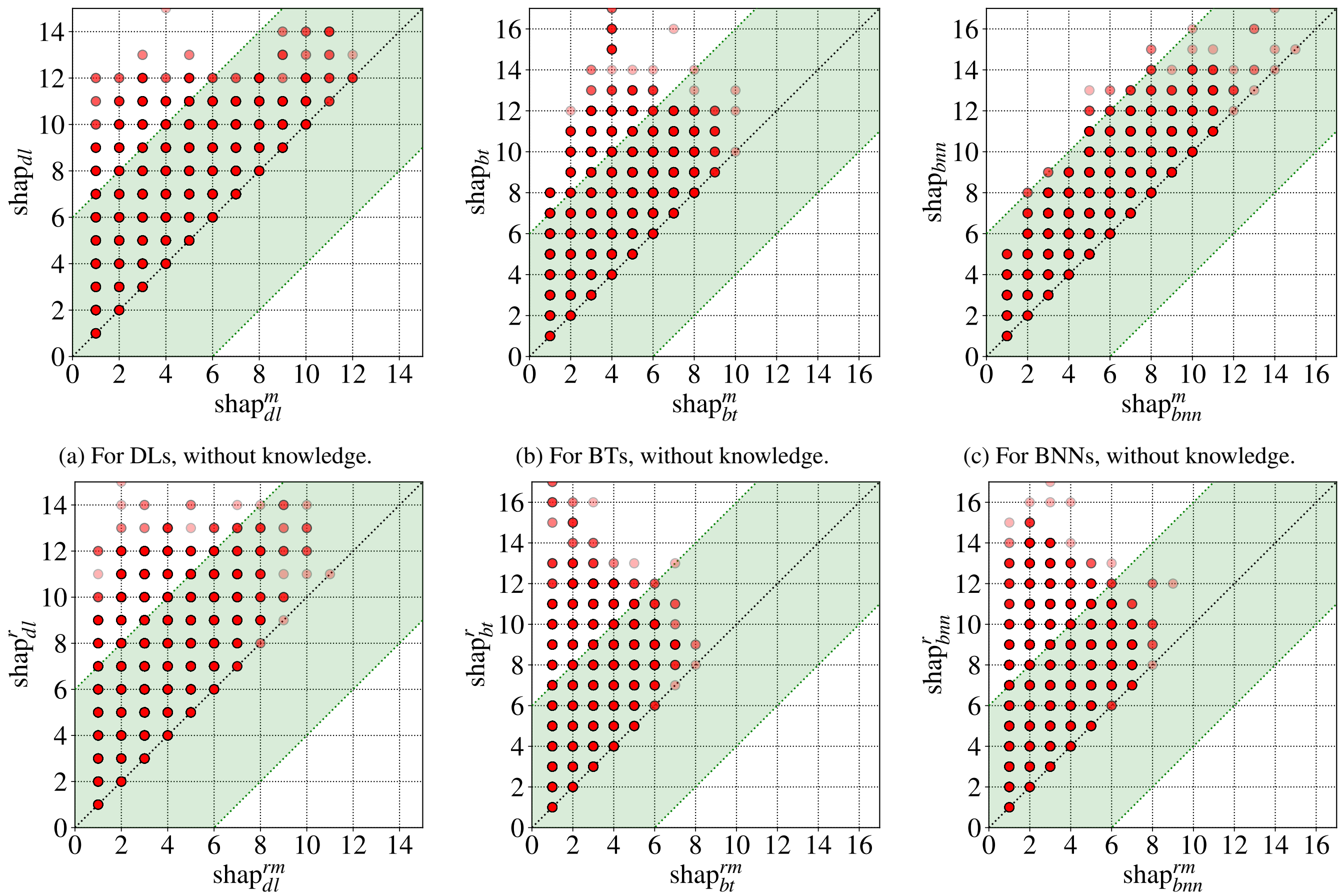
References

ACM. 2018. Fathers of the Deep Learning Revolution Receive ACM A.M. Turing Award. http://tiny.cc/9plzpz.

Agrawal, R.; and Srikant, R. 1994. Fast algorithms for mining association rules. In *VLDB*, 487–499.

Amgoud, L.; and Ben-Naim, J. 2022. Axiomatic Foundations of Explainability. In Raedt, L. D., ed., *IJCAI*, 636–642.

Angwin, J.; Larson, J.; Mattu, S.; and Kirchner, L. 2016. Machine Bias. http://tiny.cc/dd7mjz.



(a) For DLs, without knowledge.

(b) For BTs, without knowledge.

(c) For BNNs, without knowledge.

(d) For DLs, with background knowledge.

(e) For BTs, with background knowledge.

(f) For BNNs, with background knowledge.

Figure 14: Size of SHAP explanations for DLs, BTs and BNNs.

Arenas, M.; Baez, D.; Barceló, P.; Pérez, J.; and Subercaseaux, B. 2021. Foundations of Symbolic Languages for Model Interpretability. In *NeurIPS*.

Arenas, M.; Barceló, P.; Romero, M.; and Subercaseaux, B. 2022. On Computing Probabilistic Explanations for Decision Trees. *CoRR*, abs/2207.12213.

Audemard, G.; Koriche, F.; and Marquis, P. 2020. On Tractable XAI Queries based on Compiled Representations. In *KR*, 838–849.

Bailey, J.; and Stuckey, P. J. 2005. Discovery of Minimal Unsatisfiable Subsets of Constraints Using Hitting Set Dualization. In *PADL*, 174–186.

Belaid, M.; Bessiere, C.; and Lazaar, N. 2019. Constraint Programming for Association Rules. In Berger-Wolf, T. Y.; and Chawla, N. V., eds., *SIAM*, 127–135.

Belov, A.; Lynce, I.; and Marques-Silva, J. 2012. Towards efficient MUS extraction. *AI Commun.*, 25(2): 97–116.

Bendík, J.; Cerná, I.; and Benes, N. 2018. Recursive Online Enumeration of All Minimal Unsatisfiable Subsets. In *ATVA*, 143–159.

Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds. 2021. *Handbook of Satisfiability*. IOS Press. ISBN 978-1-64368-160-3.

Blanc, G.; Lange, J.; and Tan, L. 2021. Provably efficient, succinct, and precise explanations. In *NeurIPS*.

Boumazouza, R.; Alili, F. C.; Mazure, B.; and Tabia, K. 2021. ASTERYX: A model-Agnostic SaT-basEd appRoach for sYmbolic and score-based eXplanations. In *CIKM*, 120–129.

Camburu, O.; Giunchiglia, E.; Foerster, J.; Lukasiewicz, T.; and Blunsom, P. 2019. Can I Trust the Explainer? Verifying Post-hoc Explanatory Methods. *CoRR*, abs/1910.02065.

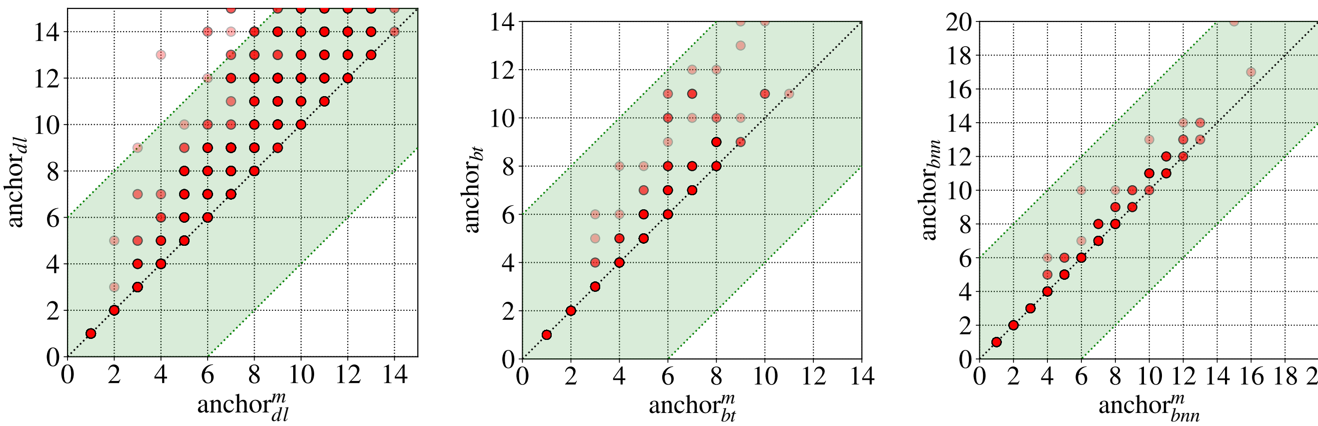
Chen, T.; and Guestrin, C. 2016. XGBoost: A Scalable Tree Boosting System. In *KDD*, 785–794.

Clark, P.; and Niblett, T. 1989. The CN2 Induction Algorithm. *Machine Learning*, 3: 261–283.

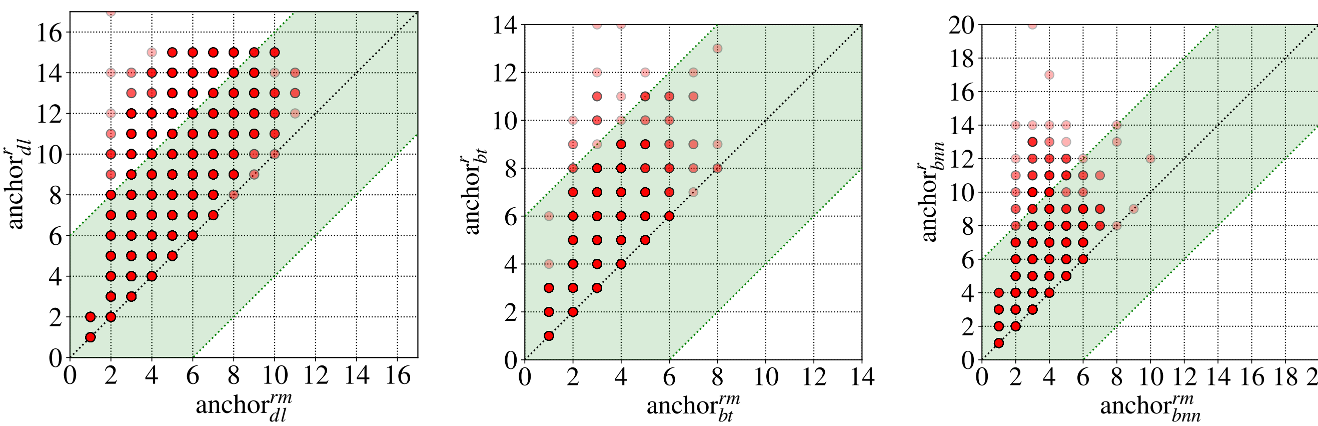
Darwiche, A.; and Hirth, A. 2020. On the Reasons Behind Decisions. In *ECAI*, 712–720.

Darwiche, A.; and Marquis, P. 2021. On Quantifying Literals in Boolean Logic and Its Applications to Explainable AI. *J. Artif. Intell. Res.*, 72: 285–328.

Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository.



(a) For DLs, without knowledge. (b) For BTs, without knowledge. (c) For BNNs, without knowledge.



(d) For DLs, with background knowledge. (e) For BTs, with background knowledge. (f) For BNNs, with background knowledge.

Figure 15: Size of Anchor explanations for DLs, BTs and BNNs.

Ferreira, J.; de Sousa Ribeiro, M.; Gonçalves, R.; and Leite, J. 2022. Looking Inside the Black-Box: Logic-based Explanations for Neural Networks. In *KR*, 432–442.

Friedman, J. H. 2001. Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, 29(5): 1189–1232.

Gario, M.; and Micheli, A. 2015. PySMT: a Solver-Agnostic Library for Fast Prototyping of SMT-Based Algorithms. In *SMT Workshop*.

Ghosh, B.; and Meel, K. S. 2019. IMLI: An Incremental Framework for MaxSAT-Based Learning of Interpretable Classification Rules. In *AIES*, 203–210.

Gorji, N.; and Rubin, S. 2022. Sufficient Reasons for Classifier Decisions in the Presence of Domain Constraints. In *AAAI*, 5660–5667.

Hipp, J.; Güntzer, U.; and Nakhaeizadeh, G. 2000. Algorithms for Association Rule Mining - A General Survey and Comparison. *SIGKDD Explor.*, 2(1): 58–64.

Huang, X.; Izza, Y.; Ignatiev, A.; Cooper, M. C.; Asher, N.; and Marques-Silva, J. 2022. Tractable Explanations for d-DNNF Classifiers. In *AAAI*, 5719–5728.

Hubara, I.; Courbariaux, M.; Soudry, D.; El-Yaniv, R.; and Bengio, Y. 2016. Binarized Neural Networks. In *NIPS*, 4107–4115.

Ignatiev, A. 2020. Towards Trustable Explainable AI. In *IJCAI*, 5154–5158.

Ignatiev, A.; Izza, Y.; Stuckey, P. J.; and Marques-Silva, J. 2022. Using MaxSAT for Efficient Explanations of Tree Ensembles. In *AAAI*, 3776–3785.

Ignatiev, A.; Lam, E.; Stuckey, P. J.; and Marques-Silva, J. 2021. A Scalable Two Stage Approach to Computing Optimal Decision Sets. In *AAAI*, 3806–3814.

Ignatiev, A.; and Marques-Silva, J. 2021. SAT-Based Rigorous Explanations for Decision Lists. In *SAT*, 251–269.

Ignatiev, A.; Morgado, A.; and Marques-Silva, J. 2018. PySAT: A Python Toolkit for Prototyping with SAT Oracles. In *SAT*, 428–437.

Ignatiev, A.; Morgado, A.; and Marques-Silva, J. 2019. RC2: an Efficient MaxSAT Solver. *J. Satisf. Boolean Model. Comput.*, 11(1): 53–64.

Ignatiev, A.; Narodytska, N.; Asher, N.; and Marques-Silva, J. 2020. From Contrastive to Abductive Explanations and Back Again. In *AI*IA*, 335–355.

Ignatiev, A.; Narodytska, N.; and Marques-Silva, J. 2019a. Abduction-Based Explanations for Machine Learning Models. In *AAAI*, 1511–1519.

Ignatiev, A.; Narodytska, N.; and Marques-Silva, J. 2019b. On Relating Explanations and Adversarial Examples. In *NeurIPS*, 15857–15867.

Ignatiev, A.; Narodytska, N.; and Marques-Silva, J. 2019c. On Validating, Repairing and Refining Heuristic ML Explanations. *CoRR*, abs/1907.02509.

Ignatiev, A.; Pereira, F.; Narodytska, N.; and Marques-Silva, J. 2018. A SAT-Based Approach to Learn Explainable Decision Sets. In *IJCAR*, 627–645.

Ignatiev, A.; Previti, A.; Liffiton, M. H.; and Marques-Silva, J. 2015. Smallest MUS Extraction with Minimal Hitting Set Dualization. In *CP*, 173–182.

Izza, Y.; Ignatiev, A.; and Marques-Silva, J. 2022. On Tackling Explanation Redundancy in Decision Trees. *J. Artif. Intell. Res.*, 75: 261–321.

Izza, Y.; Jabbour, S.; Raddaoui, B.; and Boudane, A. 2020. On the Enumeration of Association Rules: A Decomposition-based Approach. In Bessiere, C., ed., *IJCAI*, 1265–1271.

Izza, Y.; and Marques-Silva, J. 2021. On Explaining Random Forests with SAT. In *IJCAI*.

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

Kohavi, R. 1996. Scaling Up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybrid. In *KDD*, 202–207.

Lakkaraju, H.; Bach, S. H.; and Leskovec, J. 2016. Interpretable Decision Sets: A Joint Framework for Description and Prediction. In *KDD*, 1675–1684.

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

Liffiton, M. H.; Previti, A.; Malik, A.; and Marques-Silva, J. 2016. Fast, flexible MUS enumeration. *Constraints An Int. J.*, 21(2): 223–250.

Liffiton, M. H.; and Sakallah, K. A. 2008. Algorithms for Computing Minimal Unsatisfiable Subsets of Constraints. *J. Autom. Reasoning*, 40(1): 1–33.

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

Lundberg, S. M.; and Lee, S. 2017. A Unified Approach to Interpreting Model Predictions. In *NeurIPS*, 4765–4774.

Malfa, E. L.; Michelmore, R.; Zbrzezny, A. M.; Paoletti, N.; and Kwiatkowska, M. 2021. On Guaranteed Optimal Robust Explanations for NLP Models. In *IJCAI*, 2658–2665.

Malioutov, D.; and Meel, K. S. 2018. MLIC: A MaxSAT-Based Framework for Learning Interpretable Classification Rules. In *CP*, 312–327.

Marques-Silva, J.; Gerspacher, T.; Cooper, M. C.; Ignatiev, A.; and Narodytska, N. 2020. Explaining Naive Bayes and Other Linear Classifiers with Polynomial Time and Delay. In *NeurIPS*.

Marques-Silva, J.; Gerspacher, T.; Cooper, M. C.; Ignatiev, A.; and Narodytska, N. 2021. Explanations for Monotonic Classifiers. In *ICML*, 7469–7479.

Marques-Silva, J.; Heras, F.; Janota, M.; Previti, A.; and Belov, A. 2013. On Computing Minimal Correction Subsets. In *IJCAI*, 615–622.

Marques-Silva, J.; and Ignatiev, A. 2022. Delivering Trustworthy AI through Formal XAI. In *AAAI*, 3806–3814.

Mencia, C.; Previti, A.; and Marques-Silva, J. 2015. Literal-Based MCS Extraction. In *IJCAI*, 1973–1979.

Miller, T. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artif. Intell.*, 267: 1–38.

Narodytska, N.; Shrotri, A. A.; Meel, K. S.; Ignatiev, A.; and Marques-Silva, J. 2019. Assessing Heuristic Machine Learning Explanations with Model Counting. In *SAT*, 267–278.

Olson, R. S.; Cava, W. G. L.; Orzechowski, P.; Urbanowicz, R. J.; and Moore, J. H. 2017. PMLB: a large benchmark suite for machine learning evaluation and comparison. *BioData Min.*, 10(1): 36:1–36:13.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E. Z.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, 8024–8035.

Reiter, R. 1987. A Theory of Diagnosis from First Principles. *Artif. Intell.*, 32(1): 57–95.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *KDD*, 1135–1144.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2018. Anchors: High-Precision Model-Agnostic Explanations. In *AAAI*, 1527–1535.

Rivest, R. L. 1987. Learning Decision Lists. *Mach. Learn.*, 2(3): 229–246.

Rudin, C. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat. Mach. Intell.*, 1(5): 206–215.

Shih, A.; Choi, A.; and Darwiche, A. 2018. A Symbolic Approach to Explaining Bayesian Network Classifiers. In *IJCAI*, 5103–5111.

Shrotri, A.; Narodytska, N.; Ignatiev, A.; Meel, K. S.; Marques-Silva, J.; and Vardi, M. 2022. Constraint-Driven Explanations of Black-Box ML Models. In *AAAI*.

Slack, D.; Hilgard, S.; Jia, E.; Singh, S.; and Lakkaraju, H. 2020. Fooling LIME and SHAP: Adversarial Attacks on Post hoc Explanation Methods. In *AIES*, 180–186.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I. J.; and Fergus, R. 2014. Intriguing properties of neural networks. In *ICLR (Poster)*.

Wäldchen, S.; MacDonald, J.; Hauch, S.; and Kutyniok, G. 2021. The Computational Complexity of Understanding Binary Classifier Decisions. *J. Artif. Intell. Res.*, 70: 351–387.

Yu, J.; Ignatiev, A.; Stuckey, P. J.; and Bodic, P. L. 2020. Computing Optimal Decision Sets with SAT. In *CP*, 952–970.

Yu, J.; Ignatiev, A.; Stuckey, P. J.; and Le Bodic, P. 2021. Learning Optimal Decision Sets and Lists with SAT. *JAIR*, 72: 1251–1279.

Zaki, M. J.; Parthasarathy, S.; Ogihara, M.; and Li, W. 1997. New Algorithms for Fast Discovery of Association Rules. In *KDD*, 283–286.

Zhang, C.; and Zhang, S. 2002. *Association Rule Mining, Models and Algorithms*.