On Efficiently Explaining Graph-Based Classifiers

Xuanxiang Huang, Yacine Izza, Alexey Ignatiev, Joao Marques-Silva

On Efficiently Explaining Graph-Based Classifiers

Xuanxiang Huang[1] , **Yacine Izza**[1] , **Alexey Ignatiev**[2] , **Joao Marques-Silva**[3]
[1]University of Toulouse, France
[2]Monash University, Melbourne, Australia
[3]IRIT, CNRS, Toulouse, France
{xuanxiang.huang,yacine.izza}@univ-toulouse.fr, alexey.ignatiev@monash.edu,
joao.marques-silva@irit.fr

Abstract

Recent work has shown that not only decision trees (DTs) may not be interpretable but also proposed a polynomial-time algorithm for computing one PI-explanation of a DT. This paper shows that for a wide range of classifiers, globally referred to as decision graphs, and which include decision trees and binary decision diagrams, but also their multi-valued variants, there exist polynomial-time algorithms for computing one PI-explanation. In addition, the paper also proposes a polynomial-time algorithm for computing one contrastive explanation. These novel algorithms build on explanation graphs (XpG's). XpG's denote a graph representation that enables both theoretical and practically efficient computation of explanations for decision graphs. Furthermore, the paper proposes a practically efficient solution for the enumeration of explanations, and studies the complexity of deciding whether a given feature is included in some explanation. For the concrete case of decision trees, the paper shows that the set of all contrastive explanations can be enumerated in polynomial time. Finally, the experimental results validate the practical applicability of the algorithms proposed in the paper on a wide range of publicly available benchmarks.

1 Introduction

The emerging societal impact of Machine Learning (ML) and its foreseen deployment in safety critical applications, puts additional demands on approaches for verifying and explaining ML models (Weld and Bansal 2019). The vast majority of approaches for explainability in ML (often referred to as eXplainable AI (XAI) (Gunning and Aha 2019)) are heuristic, offering no formal guarantees of soundness, with well-known examples including tools like LIME, SHAP or Anchors (Ribeiro, Singh, and Guestrin 2016b; Lundberg and Lee 2017; Ribeiro, Singh, and Guestrin 2018). (Recent surveys (Guidotti et al. 2019) cover a wider range of heuristic methods.) Moreover, recent work has shed light on the important practical limitations of heuristic XAI approaches (Narodytska et al. 2019b; Ignatiev, Narodytska, and Marques-Silva 2019c; Camburu et al. 2019; Slack et al. 2020; Lakkaraju and Bastani 2020; Dimanov et al. 2020; Ignatiev 2020).

In contrast, formal approaches to XAI have been proposed in recent years (Shih, Choi, and Darwiche 2018; Ignatiev, Narodytska, and Marques-Silva 2019a; Shih, Choi, and Darwiche 2019; Ignatiev, Narodytska, and Marques-Silva 2019b; Darwiche and Hirth 2020; Audemard, Koriche, and Marquis 2020; Audemard et al. 2021) (albeit it can be related to past work on logic-based explanations (e.g. (Shanahan 1989; Falappa, Kern-Isberner, and Simari 2002; Pérez and Uzcátegui 2003))). The most widely studied form of explanation consists in the identification of prime implicants (PI) of the decision function associated with an ML classifier, being referred to as PI-explanations. Although PI-explanations offer important formal guarantees, e.g. they represent minimal sufficient reasons for a prediction, they do have their own drawbacks. First, in most settings, finding one PI-explanation is NP-hard, and in some settings scalability is an issue (Shih, Choi, and Darwiche 2018; Ignatiev, Narodytska, and Marques-Silva 2019a). Second, users have little control on the size of computed PI-explanations (and it is well-known the difficulty that humans have in grasping complex concepts). Third, there can be many PI-explanations, and it is often unclear which ones are preferred. Fourth, in practice users may often prefer high-level explanations, in contrast with feature-based, low-level explanations. Despite these drawbacks, it is plain that PI-explanations offer a sound basis upon which one can expect to develop theoretically sound and practically effective approaches for computing explanations. For example, more recent work has demonstrated the tractability of PI-explanations for some ML models (Izza, Ignatiev, and Marques-Silva 2020; Audemard, Koriche, and Marquis 2020; Marques-Silva et al. 2020; Marques-Silva et al. 2021; Izza et al. 2021), in some cases allowing for polynomial delay enumeration (Marques-Silva et al. 2020). Also, recent work (Ignatiev 2020; Izza and Marques-Silva 2021; Ignatiev and Marques-Silva 2021; Izza et al. 2021) showed that, even for ML models for which computing a PI-explanation is NP-hard, scalability may not be an obstacle.

Moreover, it was recently shown that finding explanations can be crucial even for ML models that are generally deemed interpretable[1]. One such example are decision trees (Izza, Ignatiev, and Marques-Silva 2020). Decision trees (DTs) are not only among the most widely used ML models,

[1]Interpretability is regarded a subjective concept, with no accepted rigorous definition (Lipton 2018). In this paper, we equate interpretability with explanation succinctness.

but are also generally regarded as interpretable (Breiman 2001; Freitas 2013; Ribeiro, Singh, and Guestrin 2016a; Montavon, Samek, and Müller 2018; Samek et al. 2019; Molnar 2019; Miller 2019; Guidotti et al. 2019; Rudin 2019; Xu et al. 2019; Silva et al. 2020). However, recent work (Izza, Ignatiev, and Marques-Silva 2020) has shown that paths in DTs may contain literals that are irrelevant for identifying minimal sufficient reasons for a prediction, and that the number of redundant literals can grow asymptotically as large as the number of features. Furthermore, it was also shown (Izza, Ignatiev, and Marques-Silva 2020) that PI-explanations for DTs can be computed in polynomial time. Moreover, independent work showed that finding a smallest explanation is hard for NP (Barceló et al. 2020), thus hinting at the need to finding PI-explanations in the case of DTs.

This paper complements this earlier work with several novel results. First, the paper considers both PI (or abductive) explanations (AXps) and contrastive explanations (CXps) (Miller 2019; Ignatiev et al. 2020), which will be jointly referred to as explanations (XPs). Second, the paper shows that XPs can be computed in polynomial time for a much larger class of classifiers, which will be conjointly referred to as *decision graphs* (Oliver 1992; Kohavi 1994)[2]. For that, the paper introduces a new graph representation, namely the *explanation graph*, and shows that for any classifier (and instance) that can be reduced to an explanation graph, XPs can be computed in polynomial time. (For example, multi-valued variants of decision trees, graphs or diagrams can be reduced to explanation graphs.) The paper also shows that the MARCO algorithm for enumerating MUSes/MCSes (Liffiton et al. 2016) can be adapted to the enumeration of XPs, yielding a solution that is very efficient in practice. For the case of DTs, the paper proves that the set of all CXps can be computed in polynomial time. In turn, this result offers an alternative approach for the enumeration of PI-explanations (AXps), e.g. based on hitting set dualization (Reiter 1987; Liffiton and Sakallah 2008). Finally, we investigate the *explanation membership problem*, i.e. to decide whether a feature (given its assigned value) can be included in some explanation (either AXp or CXp). The paper shows that for arbitrary explanation graphs, the explanation membership problem is in NP, while for a propositional formula in disjunctive normal form (DNF) is shown to be hard for Σ_2^{P}. However, for tree explanation graphs (which can represent explanations of decision trees), deciding explanation membership is shown to be in P.

The paper is organized as follows. Section 2 introduces the definitions and notation used in the rest of the paper. Section 3 studies explanation graphs (XpG's), and shows how XpG's can be used for computing explanations. Afterwards, Section 4 describes algorithms computing one XP (either AXp or CXp) of XpG's, and a MARCO-like algorithm for the enumeration of XPs. Section 4 also proves that for DTs, the set of all CXps can be computed in polynomial time. Some of the previous results are used in Section 5 for investigating the complexity of deciding membership of features in explanations. Section 6 relates the paper's contributions with earlier work. Section 7 discusses experimental results of explaining DTs and reduced ordered binary decision diagrams, including AXps, CXps and their enumeration. Finally, the paper concludes in Section 8.

[2]The term *decision graph* is also used in the context of Bayesian Networks (Jensen 2001; Darwiche 2009), and more recently in explainability (Shih, Choi, and Darwiche 2018; Shih, Choi, and Darwiche 2019). However, and to the best of our knowledge, the term "decision graph" was first proposed in the early 90s (Oliver 1992) to enable more compact representation of DTs.

2 Preliminaries

Classification problems. A classification problem is defined on a set of features (or attributes) $\mathcal{F} = \{1, \ldots, m\}$ and a set of classes $\mathcal{K} = \{c_1, c_2, \ldots, c_K\}$. Each feature $i \in \mathcal{F}$ takes values from a domain $\mathbb{D}_i$. Domains can be boolean, integer or real-valued. Feature space is defined as $\mathbb{F} = \mathbb{D}_1 \times \mathbb{D}_2 \times \ldots \times \mathbb{D}_m$. The notation $\mathbf{x} = (x_1, \ldots, x_m)$ denotes an arbitrary point in feature space, where each x_i is a variable taking values from $\mathbb{D}_i$. Moreover, the notation $\mathbf{v} = (v_1, \ldots, v_m)$ represents a specific point in feature space, where each v_i is a constant representing one concrete value from $\mathbb{D}_i$. An *instance* (or example) denotes a pair $(\mathbf{v}, c)$, where $\mathbf{v} \in \mathbb{F}$ and $c \in \mathcal{K}$. (We also use the term *instance* to refer to $\mathbf{v}$, leaving c implicit.) An ML classifier $\mathbb{C}$ is characterized by a *classification function* κ that maps feature space $\mathbb{F}$ into the set of classes $\mathcal{K}$, i.e. $\kappa : \mathbb{F} \to \mathcal{K}$. ($\kappa$ is assumed to be non-constant.)

Remark on binarization. We underline the importance of not restricting feature domains to be boolean-valued. Although binarization can be used to represent features that are categorical, integer or real-valued, it is also the case that, from the perspective of computing explanations, soundness demands that one must know whether binarization was applied and, if so, which resulting binary features must be related with which original features. The key observation is that if binarization is used, then soundness of results imposes that features *must* be reasoned about in groups of related binary features, and this implies algorithms that work under this assumption. In this paper, we opt to impose no such restriction when reasoning about explanations.

Abductive and contrastive explanations. We now define formal explanations. Prime implicant (PI) explanations (Shih, Choi, and Darwiche 2018) denote a minimal set of literals (relating a feature value x_i and a constant $v_i \in \mathbb{D}_i$ that are sufficient for the prediction[3]. Formally, given $\mathbf{v} = (v_1, \ldots, v_m) \in \mathbb{F}$ with $\kappa(\mathbf{v}) = c$, a PI-explanation (AXp) is any minimal subset $\mathcal{X} \subseteq \mathcal{F}$ such that,

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

AXps can be viewed as answering a 'Why?' question, i.e. why is some prediction made given some point in feature

[3]PI-explanations are related with abduction, and so are also referred to as abductive explanations (AXp) (Ignatiev, Narodytska, and Marques-Silva 2019a). More recently, PI-explanations have been studied from a knowledge compilation perspective (Audemard, Koriche, and Marquis 2020).

space. A different view of explanations is a contrastive explanation (Miller 2019), which answers a 'Why Not?' question, i.e. which features can be changed to change the prediction. A formal definition of contrastive explanation is proposed in recent work (Ignatiev et al. 2020). Given $\mathbf{v} = (v_1, \ldots, v_m) \in \mathbb{F}$ with $\kappa(\mathbf{v}) = c$, a CXp is any minimal subset $\mathcal{Y} \subseteq \mathcal{F}$ such that,

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

Building on the results of R. Reiter in model-based diagnosis (Reiter 1987), (Ignatiev et al. 2020) proves a minimal hitting set (MHS) duality relation between AXps and CXps, i.e. AXps are MHSes of CXps and vice-versa.

Section 5 studies the explanation membership problem, which we define as follows:

Definition 1 (AXp/CXp Membership Problem)**.** *Given $\mathbf{v} \in \mathbb{F}$ and $i \in \mathcal{F}$, with $\kappa(\mathbf{v}) = c \in \mathcal{K}$, the AXp (resp. CXp) membership problem is to decide whether there exists an AXp (resp. CXp) $\mathcal{Z} \subseteq \mathcal{F}$ with $i \in \mathcal{Z}$.*

One can understand the importance of deciding explanation membership in settings where the number of explanations is very large, and we seek to understand whether some feature (given its assigned value) can be relevant for some prediction. Duality between explanations (Ignatiev et al. 2020) yields the following result.

Proposition 1. *Given $\mathbf{v} \in \mathbb{F}$, there exists an AXp $\mathcal{X} \subseteq \mathcal{F}$ with $i \in \mathcal{X}$ iff there exists a CXp $\mathcal{Y} \subseteq \mathcal{F}$ with $i \in \mathcal{Y}$.*

Decision trees, diagrams and graphs. A decision tree $\mathcal{T}$ is a directed acyclic graph having at most one path between every pair of nodes. $\mathcal{T}$ has a root node, characterized by having no incoming edges. All other nodes have one incoming edge. We consider univariate decision trees (as opposed to multivariate decision trees (Brodley and Utgoff 1995)), each non-terminal node is associated with a single feature x_i. Each edge is labeled with a literal, relating a feature (associated with the edge's starting node) with some values (or range of values) from the feature's domain. We will consider literals to be of the form $x_i \bowtie \mathbb{E}_i$, where $\bowtie \in \{\in\}$. x_i is a variable that denotes the value taken by feature i, whereas $\mathbb{E}_i \subseteq \mathbb{D}_i$ is a subset of the domain of feature i. The type of literals used to label the edges of a DT allows the representation of the DTs generated by a wide range of decision tree learners (e.g. (Utgoff, Berkman, and Clouse 1997)). (The syntax of the literals could be enriched. For example, we could use $\bowtie \in \{\in, \notin\}$, or for categorical features we could use $\bowtie \in \{=, \neq\}$, in which case we would need to allow for multi-tree variants of DTs (Appuswamy et al. 2011). However, these alternatives would not change the results in the paper. As formalized later, the literals associated with the outgoing edges are assumed to be mutually inconsistent. Finally, each terminal node is associated with a value from $\mathcal{K}$.

Throughout the paper, we will use the following example of a DT as the first running example.

Example 1. *For the DT in Figure 1, $\mathcal{F} = \{1, 2, 3, 4\}$, denoting respectively Age ($\in \{\mathrm{W}, \mathrm{T}, \mathrm{O}\}$), Income ($\in \{\mathrm{L}, \mathrm{M}, \mathrm{H}\}$), Student ($\in \{\mathrm{N}, \mathrm{Y}\}$) and Credit Rating ($\in \{\mathrm{P}, \mathrm{F}, \mathrm{E}\}$). The prediction is the type of hardware bought, with* N *denoting No Hardware,* T *denoting a Tablet and* L *denoting a Laptop. For Age,* W, T *and* O *denote, respectively,* Age < 30 *(tWenties or younger),* $30 \leq$ Age < 40 *(Thirties) and* $40 \leq$ Age *(forties or Older). For Income,* L, M, H *denote, respectively, (L)ow, (M)edium, and (H)igh. For Student,* N *denotes not a student and* Y *denotes a student. Finally, for Credit Rating,* P, F *and* E *denote, respectively, (P)oor, (F)air and (E)xcellent. For the instance $\mathbf{v} = (\mathrm{O}, \mathrm{L}, \mathrm{Y}, \mathrm{P})$, with prediction* T *(i.e. Tablet), the consistent path is shown highlighted.*

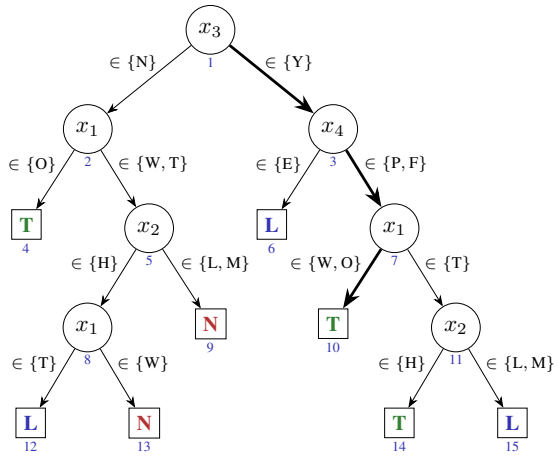

Figure 1: Example DT, $\mathbf{v} = (\mathrm{O}, \mathrm{L}, \mathrm{Y}, \mathrm{P})$ and $\kappa(\mathbf{v}) = \mathbf{T}$

The paper also considers reduced ordered binary decision diagrams (OBDDs) (Bryant 1986; Wegener 2000; Darwiche and Marquis 2002), as well as their multi-valued variant, i.e. reduced ordered multi-valued decision diagrams (OMDDs) (Srinivasan et al. 1990; Kam and Brayton 1990; Bergman et al. 2016). (We will also briefly mention connections with deterministic branching programs (DBPs) (Wegener 2000).) For OBDDs, features must be boolean, and so each edge is labeled with either 0 or 1 (we could instead use $\in \{0\}$ and $\in \{1\}$, respectively). In contrast with DTs, features in OBDDs must also be ordered. For OMDDs, features takes discrete values, and are also ordered. In this case, we label edges the same way we label edges in DTs, i.e. using set membership (and non-membership). (For simplicity, if all edges have a single option, we just label the edge with the value.) Moreover, we will use the following example of an OMDD as the paper's second running example.

Example 2. *For the OMDD in Figure 2, $\mathcal{F} = \{1, 2, 3\}$, with $D_1 = D_2 = \{0, 1\}$, $D_3 = \{0, 1, 2\}$. The prediction is one of three classes $\mathcal{K} = \{\mathrm{R}, \mathrm{G}, \mathrm{B}\}$. For the instance $\mathbf{v} = (0, 1, 2)$, with prediction* R*, the consistent path is shown highlighted.*

Over the years, different works proposed the use of some sort of decision diagrams as an alternative to decision trees, e.g. (Bahl et al. 1989; Oliver 1992; Oliveira and Sangiovanni-Vincentelli 1996; Mues et al. 2004; Cabodi et al. 2021). This paper considers *decision graphs* (Oliver 1992), which can be viewed as a generalization of DTs, OBDDs, OMDDs, etc.

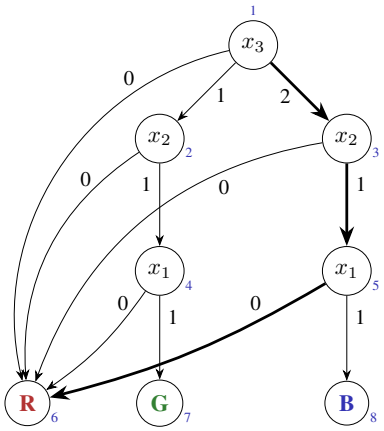

Figure 2: Example OMDD, $\mathbf{v} = (0, 1, 2)$ and $\kappa(\mathbf{v}) = \mathbf{R}$

Definition 2 (Decision Graph (DG))**.** *A DG $\mathcal{G}$ is a 4-tuple $\mathcal{G} = (G, \varsigma, \phi, \lambda)$ where,*

1. *$G = (V, E)$ is a Directed Acyclic Graph (DAG) with a single source (or root) node.*
2. *V is partitioned into terminal (T) and non-terminal (N) nodes. For every $p \in N$, $\deg^+(p) > 0$. For every $q \in T$, $\deg^+(q) = 0$. ($\deg^+$ denotes the outdegree of a node).*
3. *$\varsigma : T \to \mathcal{K}$ maps each terminal node into a class.*
4. *$\phi : N \to \mathcal{F}$ maps each non-terminal node into a feature.*
5. *$\lambda : E \to \mathfrak{L}$, where $\mathfrak{L}$ denotes the set of all literals of the form $x_i \in \mathbb{E}_i$ for $\mathbb{E}_i \subseteq \mathbb{D}_i$, where i is the feature associated with the edge's starting node.*

Furthermore, the following assumptions are made with respect to DGs[4] (where for node $r \in N$, with $\phi(r) = i$, $\mathbb{C}_i$ denotes the set of values of feature i which are consistent with any path connecting the root to r):

- *i. The literals associated with the outgoing edges of each node $r \in N$ represent a partition of $\mathbb{C}_i$.*
- *ii. Every path R_k of DG, that connects the root node to a terminal node, is not inconsistent.*

It is straightforward to conclude that any classifier defined on DTs, OBDDs or OMDDs, and respecting assumptions (i) and (ii) of the above definition, can be represented as a DG. (The same claim can also be made for DBPs.) Also, whereas OBDDs and OMDDs are read-once (i.e. each feature is tested at most once along a path), DTs (and general DGs) need not be read-once. Hence, DGs impose no restriction on the number of times a feature is tested along a path, as long as the literals are consistent. Moreover, the definition of DG (and the associated assumptions) ensures that,

Proposition 2. *For any $\mathbf{v} \in \mathbb{F}$ there exists exactly one terminal node which is connected to the root node of the DG by path(s) such that each edge in such path(s) is consistent with the feature values given by $\mathbf{v}$.*

Proposition 3. *For any terminal node q of a DG, there exists at least one point $\mathbf{v} \in \mathbb{F}$ such there is a consistent path from the root to q.*

3 Explanation Graphs

A difficulty with reasoning about explanations for DTs, DGs or OBDDs, but also for their multi-valued variants (and also in the case of other examples of ML models), is the multitude of cases that one needs to consider. For the concrete case of OBDDs, features are restricted to be boolean. However, for DTs and DGs, features can be boolean, categorical, integer or real. Moreover, for OMDDs, features can be boolean, categorical or integer. Also, it is often the case that $|\mathcal{K}| > 2$. Explanation graphs are a graph representation that abstracts away all the details that are effectively unnecessary for computing AXps or CXps. In turn, this facilitates the construction of unified explanation procedures.

Definition 3 (Explanation Graph (XpG))**.** *An XpG is a 5-tuple $\mathcal{D} = (G_{\mathcal{D}}, S, \upsilon, \alpha_V, \alpha_E)$, where:*

1. *$G_{\mathcal{D}} = (V_{\mathcal{D}}, E_{\mathcal{D}})$ is a labeled DAG, such that:*
   - *$V_{\mathcal{D}} = T_{\mathcal{D}} \cup N_{\mathcal{D}}$ is the set of nodes, partitioned into the terminal nodes $T_{\mathcal{D}}$ (with $\deg^+(q) = 0$, $q \in T_{\mathcal{D}}$) and the non-terminal nodes $N_{\mathcal{D}}$ (with $\deg^+(p) > 0$, $p \in N_{\mathcal{D}}$);*
   - *$E_{\mathcal{D}} \subseteq V_{\mathcal{D}} \times V_{\mathcal{D}}$ is the set of (directed) edges.*
   - *$G_{\mathcal{D}}$ is such that there is a single node with indegree equal to 0, i.e. the root (or source) node.*
2. *$S = \{s_1, \ldots, s_m\}$ is a set of variables;*
3. *$\upsilon : N_{\mathcal{D}} \to S$ is a total function mapping each non-terminal node to one variable in S.*
4. *$\alpha_V : V_{\mathcal{D}} \to \{0, 1\}$ labels nodes with one of two values. (α_V is required to be defined only for terminal nodes.)*
5. *$\alpha_E : E_{\mathcal{D}} \to \{0, 1\}$ labels edges with one of two values.*

In addition, an XpG $\mathcal{D}$ must respect the following properties:

- *i. For each non-terminal node, there is at most one outgoing edge labeled 1; all other outgoing edges are labeled 0.*
- *ii. There is exactly one terminal node $t \in T$ labeled 1 that can be reached from the root node with (at least) one path of edges labeled 1.*

We refer to a *tree XpG* when the DAG associated with the XpG is a tree. Given a DG $\mathcal{G}$ and an instance $(\mathbf{v}, c)$, the (unique) mapping to an XpG is obtained as follows:

1. The same DAG is used.
2. Terminal nodes labeled c in $\mathcal{G}$ are labeled 1 in $\mathcal{D}$. Terminal nodes labeled $c' \neq c$ in $\mathcal{G}$ are labeled 0 in $\mathcal{D}$.
3. A non-terminal node associated with feature i in $\mathcal{G}$ is associated with s_i in $\mathcal{D}$.
4. Any edge labeled with a literal that is consistent with $\mathbf{v}$ in $\mathcal{G}$ is labeled 1 in $\mathcal{D}$. Any edge labeled with a literal that is not consistent with $\mathbf{v}$ in $\mathcal{G}$ is labeled 0 in $\mathcal{D}$.

Since we can represent DTs, OBDDs or OMDDs with DGs, then the construction above ensures that we can also create XpG's for any of these classifiers.

The following examples illustrate the construction of XpG's for the paper's two running examples.

[4] The importance of these assumptions must be highlighted. Whereas for OBDDs/OMDDs these assumptions are guaranteed by construction, this is not the case with DTs nor in general with DGs. In the literature, one can find examples of decision trees with inconsistent paths (e.g. (Valdes et al. 2016, Fig. 4)) but also decision trees exhibiting dead-ends, i.e. DTs for which the classification function is not total (e.g. (Duda, Hart, and Stork 2001, Fig. 8.1)).

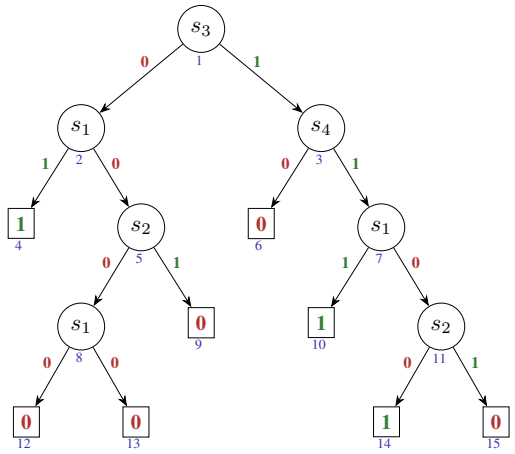

Figure 3: XpG for the DT in Figure 1, given $\mathbf{v} = (\mathrm{O}, \mathrm{L}, \mathrm{Y}, \mathrm{P})$

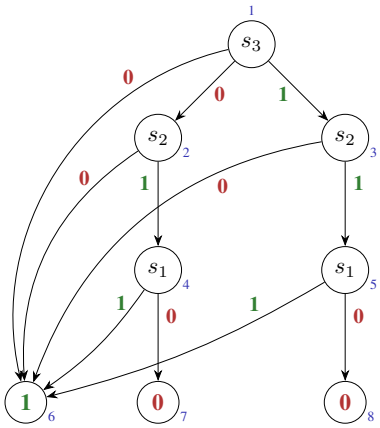

Figure 4: XpG for the OMDD of Figure 2, given $\mathbf{v} = (0, 1, 2)$

Example 3. *For the DT of Example 1 (shown in Figure 1, given the instance* $(\mathbf{v} = (\mathrm{O}, \mathrm{L}, \mathrm{Y}, \mathrm{P}), \mathrm{T})$*, and letting* $S = (s_1, s_2, s_3, s_4)$*, with each* s_i *associated with feature* i*, the resulting XpG is shown in Figure 3.*

Example 4. *For the OMDD of Example 2 (shown in Figure 2), given the instance* $((0, 1, 2), \mathrm{R})$*, and letting* $S = (s_1, s_2, s_3)$*, with each* s_i *associated with feature* i*, the resulting XpG is shown in Figure 4.*

Evaluation of XpG's. Given an XpG $\mathcal{D}$, let $\mathbb{S} = \{0, 1\}^m$, i.e. the set of possible assignments to the variables in S. The evaluation function of the XpG, $\sigma_{\mathcal{D}} : \mathbb{S} \rightarrow \{0, 1\}$, is based on the auxiliary *activation* function $\varepsilon : \mathbb{S} \times V_{\mathcal{D}} \rightarrow \{0, 1\}$. Moreover, for a point $\mathbf{s} \in \mathbb{S}$, $\sigma_{\mathcal{D}}$ and ε are defined as follows:

1. If r is the root node of $G_{\mathcal{D}}$, then $\varepsilon(\mathbf{s}, r) = 1$.
2. Let $p \in \mathsf{parent}(r)$ (i.e. a node can have multiple parents) and let $s_i = \upsilon(p)$. $\varepsilon(\mathbf{s}, r) = 1$ iff $\varepsilon(\mathbf{s}, p) = 1$ and either $\alpha_E(p, r) = 1$ or $s_i = 0$, i.e.

$$\varepsilon(\mathbf{s}, r) \equiv \bigvee_{\substack{p \in \mathsf{parent}(r) \\ \wedge \neg \alpha_E(p,r)}} (\varepsilon(\mathbf{s}, p) \wedge \neg s_i) \bigvee_{\substack{p \in \mathsf{parent}(r) \\ \wedge \alpha_E(p,r)}} \varepsilon(\mathbf{s}, p) \qquad (3)$$

3. $\sigma_{\mathcal{D}}(\mathbf{s}) = 1$ iff for every terminal node $t \in T_{\mathcal{D}}$, with $\alpha_V(t) = 0$, it is also the case that $\varepsilon(\mathbf{s}, t) = 0$, i.e.

$$\sigma_{\mathcal{D}}(\mathbf{s}) \equiv \bigwedge\nolimits_{t \in T_{\mathcal{D}} \wedge \neg \alpha_V(t)} \neg \varepsilon(\mathbf{s}, t) \qquad (4)$$

Observe that terminal nodes labeled 1 are irrelevant for defining the evaluation function. Their existence is implicit (i.e. at least one terminal node with label 1 must exist and be reachable from the root when all the s_i variables take value 1), but the evaluation of $\sigma_{\mathcal{D}}$ is oblivious to their existence. Furthermore, and as noted above, we must have $\sigma_{\mathcal{D}}(1, \ldots, 1) = 1$. If the graph has some terminal node labeled 0, then $\sigma_{\mathcal{D}}(0, \ldots, 0) = 0$.

Example 5. *For the DT of Figure 1, and given the XpG of Figure 3, the evaluation function is defined as follows:*

$$\sigma_{\mathcal{D}}(\mathbf{s}) \leftrightarrow \left(\bigwedge\nolimits_{r \in \{6,9,12,13,15\}} \neg \varepsilon(\mathbf{s}, r) \right)$$

with,

$$\begin{array}{l} [\varepsilon(\mathbf{s}, 1) \leftrightarrow 1] \wedge [\varepsilon(\mathbf{s}, 2) \leftrightarrow \varepsilon(\mathbf{s}, 1) \wedge \neg s_3] \wedge \\ [\varepsilon(\mathbf{s}, 3) \leftrightarrow \varepsilon(\mathbf{s}, 1)] \wedge [\varepsilon(\mathbf{s}, 5) \leftrightarrow \varepsilon(\mathbf{s}, 2) \wedge \neg s_1] \wedge \\ [\varepsilon(\mathbf{s}, 6) \leftrightarrow \varepsilon(\mathbf{s}, 3) \wedge \neg s_4] \wedge [\varepsilon(\mathbf{s}, 7) \leftrightarrow \varepsilon(\mathbf{s}, 3)] \wedge \\ [\varepsilon(\mathbf{s}, 8) \leftrightarrow \varepsilon(\mathbf{s}, 5) \wedge \neg s_2] \wedge [\varepsilon(\mathbf{s}, 9) \leftrightarrow \varepsilon(\mathbf{s}, 5)] \wedge \\ [\varepsilon(\mathbf{s}, 11) \leftrightarrow \varepsilon(\mathbf{s}, 7) \wedge \neg s_1] \wedge [\varepsilon(\mathbf{s}, 12) \leftrightarrow \varepsilon(\mathbf{s}, 8) \wedge \neg s_1] \wedge \\ [\varepsilon(\mathbf{s}, 13) \leftrightarrow \varepsilon(\mathbf{s}, 8) \wedge \neg s_1] \wedge [\varepsilon(\mathbf{s}, 15) \leftrightarrow \varepsilon(\mathbf{s}, 11)] \end{array}$$

(where, for simplicity and for reducing the number of parenthesis, the operator $\wedge$ *has precedence over the operator* $\leftrightarrow$*.) Observe that* $\sigma_{\mathcal{D}}(1, 1, 1, 1) = 1$ *and* $\sigma_{\mathcal{D}}(0, 0, 0, 0) = 0$*.*

Example 6. *For the OMDD of Figure 2, and given the XpG of Figure 4, the evaluation function is defined as follows:*

$$\sigma_{\mathcal{D}}(\mathbf{s}) \leftrightarrow \left(\bigwedge\nolimits_{r \in \{7,8\}} \neg \varepsilon(\mathbf{s}, r) \right)$$

with,

$$\begin{array}{l} [\varepsilon(\mathbf{s}, 1) \leftrightarrow 1] \wedge [\varepsilon(\mathbf{s}, 2) \leftrightarrow \varepsilon(\mathbf{s}, 1) \wedge \neg s_3] \wedge \\ [\varepsilon(\mathbf{s}, 3) \leftrightarrow \varepsilon(\mathbf{s}, 1)] \wedge [\varepsilon(\mathbf{s}, 4) \leftrightarrow \varepsilon(\mathbf{s}, 2)] \wedge \\ [\varepsilon(\mathbf{s}, 5) \leftrightarrow \varepsilon(\mathbf{s}, 3)] \wedge [\varepsilon(\mathbf{s}, 7) \leftrightarrow \varepsilon(\mathbf{s}, 4) \wedge \neg s_1] \wedge \\ [\varepsilon(\mathbf{s}, 8) \leftrightarrow \varepsilon(\mathbf{s}, 5) \wedge \neg s_1] \end{array}$$

Again, we have $\sigma_{\mathcal{D}}(1, 1, 1, 1) = 1$ *and* $\sigma_{\mathcal{D}}(0, 0, 0, 0) = 0$*.*

Properties of XpG's. The definition of $\sigma_{\mathcal{D}}$ is such that the evaluation function is monotone (where we define $0 \preceq 1$, $\mathbf{s}_1 \preceq \mathbf{s}_2$ if for all i, $s_{1,i} \preceq s_{2,i}$, and for monotonicity we require $\mathbf{s}_1 \preceq \mathbf{s}_2 \rightarrow \sigma_{\mathcal{D}}(\mathbf{s}_1) \preceq \sigma_{\mathcal{D}}(\mathbf{s}_2)$.

Proposition 4. *Given an XpG* $\mathcal{D}$*,* $\sigma_{\mathcal{D}}$ *is monotone.*

Proof. [Sketch] Observe that ε is monotone (and negative) on $\mathbf{s} \in \mathbb{S}$, and $\sigma_{\mathcal{D}}$ is monotone (and negative) on ε. Hence, $\sigma_{\mathcal{D}}$ is monotone (and positive) on $\mathbf{s}$. □

Given the definition of $\sigma_{\mathcal{D}}$, any PI will consist of a conjunction of positive literals (Crama and Hammer 2011). Furthermore, we can view an XpG as a classifier, mapping features $\{1, \ldots, m\}$ (each feature i associated with a variable $s_i \in S$) into $\{0, 1\}$, with instance $((1, \ldots, 1), 1)$. As a result, we can compute the AXps and CXps of an XpG $\mathcal{D}$ (given the instance $((1, \ldots, 1), 1)$).

Example 7. *Observe that by setting $s_2 = s_3 = 0$, we still guarantee that $\sigma_{\mathcal{D}}(1,0,0,1) = 1$. However, setting either $s_1 = 0$ or $s_4 = 0$, will cause $\sigma_{\mathcal{D}}$ to change value. Hence, one AXp for the XpG is $\{1,4\}$. With respect to the original instance $((\text{O},\text{L},\text{Y},\text{P}),\text{T})$, selecting $\{1,4\}$ indicates that $(x_1 = \text{O}) \land (x_4 = \text{P})$ (i.e. Age in the forties or Older and a Credit Rating of Poor) suffices for the prediction of* T.

Example 8. *With respect to Example 6, we can observe that s_2 is not used for defining $\sigma_{\mathcal{D}}$. Hence, it can be set to 0. Also, as long as $s_1 = 1$, the prediction will remain unchanged. Thus, we can also set s_3 to 0. As a result, one AXp is $\{1\}$. With respect to the original instance $((x_1,x_2,x_3),c) = ((0,1,2),\text{R})$, selecting $\{1\}$ indicates that $x_1 = 0$ suffices for the prediction of* R.

As suggested by the previous discussion and examples, we have the following result.

Proposition 5. *There is a one-to-one mapping between AXps and CXps of $\sigma_{\mathcal{D}}$ and the AXps and CXps of the original classification problem (and instance) from which the XpG $\mathcal{D}$ is obtained.*

Proof. [Sketch] The construction of the XpG from a DG ensures that for any node in the XpG, if $\varepsilon(\mathbf{s},r) = 1$, then there exists some assignment to the features corresponding to unset variables, such that there is one consistent path in the DG from the root to r. Thus, if for some pick of unset variables, we have that $\varepsilon(\mathbf{s},q) = 1$, for some $q \in T_{\mathcal{D}}$ with $\alpha_V(q) = 0$, then that guarantees that in the DG there is an assignment to the features associated with the unset variables, such that a prediction other than c is obtained. □

4 Computing Explanations

It is well-known that prime implicants of monotone functions can be computed in polynomial time (e.g. (Goldsmith, Hagen, and Mundhenk 2005; Goldsmith, Hagen, and Mundhenk 2008)). Moreover, whereas there are algorithms for finding one PI of a monotone function in polynomial time, there is evidence that enumeration of PIs cannot be achieved with polynomial delay (Gurvich and Khachiyan 1999).

Nevertheless, and given the fact that $\sigma_{\mathcal{D}}$ is defined on a DAG, this paper proposes dedicated algorithms for computing one AXp and one CXp which build on iterative graph traversals. Furthermore, the MARCO algorithm (Liffiton et al. 2016) is adapted to exploit the algorithms for computing one AXp and one CXp, in the process ensuring that AXps/CXps can be enumerated with exactly one SAT oracle call per each computed explanation. (A recent work on explaining monotonic classifiers (Marques-Silva et al. 2021) proposes a poly-time algorithm to compute one AXp (resp. CXp) and a practically efficient algorithm for the iterative enumeration of XPs.)

4.1 Finding One XP

Different polynomial-time algorithms can be envisioned for finding one prime implicant of an XpG (and also of a monotone function). For the concrete case of $\sigma_{\mathcal{D}}$, we consider the well-known deletion-based algorithm (Chinneck and Dravnieks 1991), which iteratively removes literals from the implicant, and checks the value of $\sigma_{\mathcal{D}}$ using the DAG representation. (It is also plain that we could consider instead the algorithms QuickXplain (Junker 2004) or Progression (Marques-Silva, Janota, and Belov 2013), or any other algorithm for finding a minimal set subject to a monotone predicate (Marques-Silva, Janota, and Mencía 2017).)

Algorithm 1 Check existence of path to 0-labeled terminal

input: XpG: $\mathcal{D} = (G_{\mathcal{D}}, S, \upsilon, \alpha_V, \alpha_E)$; Ref set: $R \subseteq S$

```
1:  procedure pathToZero(D, R)
2:      Q ← init(root(G_D))
3:      while not empty(Q) do
4:          (Q, p) ← dequeue(Q)
5:          if isTerminal(G_D, p) then
6:              if α_V(p) = 0 then
7:                  return true
8:          else
9:              s_i ← υ(p)
10:             for all q ∈ children(G_D, p) do
11:                 if s_i ∈ R or α_E(p, q) = 1 then
12:                     Q ← enqueue(Q, q)
13:     return false
```

As highlighted in the running examples, if $\sigma_{\mathcal{D}}(\mathbf{u}) = 1$, for some $\mathbf{u} \in \mathbb{S}$, then in the original classifier this means the prediction remains unchanged. The only way we have to change the prediction is to allow some features to take some other value from their domain. As a result, we equate $s_i = 1$ with declaring the original feature as *set* (or as fixed), whereas we equate $s_i = 0$ with declaring the original feature as *unset* (or as free). By changing some of the S variables from 1 to 0, we are allowing some of the features to take one value from their domains. If we manage to change the value of the evaluation function to 0, this means that in the original classifier there exists a pick of values to the unset features which allows the prediction to change to some class other than c. As a result, the algorithms proposed in this section are solely based on finding a subset maximal set of features declared free (respectively, fixed), which is sufficient for the prediction not to change (respectively, to change).

To enable the integration of the algorithms, the basic algorithms for finding one XP are organized such that one XP is computed given a starting seed.

Checking path to node with label 0. All algorithms are based on graph traversals, which check whether a prediction of 0 can be reached given a set of value picks for the variables in S. This graph traversal algorithm is simple to envision, and is shown in Algorithm 1. As can be observed, the algorithm returns 1 if a terminal labeled 0 can be reached. Otherwise, it returns 0. Variables in set R serve to ignore the values of outgoing edges of a node if the variable is in R. The algorithm has a linear run time on the XpG's size (i.e. $|V_{\mathcal{D}}| + |E_{\mathcal{D}}|$).

Extraction of one AXp and one CXp given seed. Given a seed set $A \subseteq S$ of set variables, and so a set $C = S \setminus A$ of unset variables (which are *guaranteed* to be kept unset), Algorithm 2 drops variables from A (i.e. makes variables unset,

Algorithm 2 Extraction of one AXp given seed A

input: XpG: $\mathcal{D} = (G_{\mathcal{D}}, S, \upsilon, \alpha_V, \alpha_E)$; Seed set: $A \subseteq S$

procedure findAXp($\mathcal{D}, A$)
  for all $s_i \in A$ **do** // Inv.: **not** pathToZero($\mathcal{D}, S \setminus A$)
    if not pathToZero($\mathcal{D}, S \setminus (A \setminus \{s_i\})$) **then**
      $A \leftarrow A \setminus \{s_i\}$
  return A

Algorithm 3 Extraction of one CXp given seed C

input: XpG: $\mathcal{D} = (G_{\mathcal{D}}, S, \upsilon, \alpha_V, \alpha_E)$; Seed set: $C \subseteq S$

procedure findCXp($\mathcal{D}, C$)
  for all $s_i \in C$ **do** // Inv.: pathToZero($\mathcal{D}, C$)
    if pathToZero($\mathcal{D}, C \setminus \{s_i\}$) **then**
      $C \leftarrow C \setminus \{s_i\}$
  return C

and so allows the original features to take one of the values in their domains). Since $\sigma_{\mathcal{D}}$ is monotone, the deletion-based algorithm is guaranteed to find a subset-minimal set of fixed variables such that the XpG evaluates to 1.

Similarly, given a seed set $C \subseteq S$ of unset variables, and so a set $A = S \setminus C$ of set variables (which are *guaranteed* to be kept set), Algorithm 3 drops variables from C (i.e. makes variables set, and so forces the original features to take the value specified by the instance).

4.2 Enumeration of Explanations

As indicated earlier in this section, we use a MARCO-like (Liffiton et al. 2016) algorithm for enumerating XPs of an XpG (see Algorithm 4). (An in-depth analysis of MARCO is included in earlier work (Liffiton et al. 2016).) Algorithm 4 exploits hitting set duality between AXps and CXps (Ignatiev et al. 2020), and represents the sets to hit (resp. block) as a set of positive (resp. negative) clauses $\mathcal{H}$, defined on a set of variables S. The algorithm iteratively calls a SAT oracle on $\mathcal{H}$ while the formula is satisfiable. Given a model, which splits S into variables assigned value 1 (i.e. set) and variables assigned value 0 (i.e. unset), we check if the model enables the prediction to change (i.e. we check the existence of a path to a terminal node labeled 0, with C as the reference set). If no such path exists, then we extract one AXp, using A as the seed. Otherwise, we extract one CXp, using C as the seed. The resulting XP is then used to block future assignments to the variables in S from repeating XPs.

4.3 Enumerating CXps for DTs

The purpose of this section is to show that, if the XpG is a tree (e.g. in the case of a DT), then the number of CXps is polynomial on the size of the XpG. Furthermore, it is shown that the set of all CXps can be computed in polynomial time. This result has a number of consequences, some of which are discussed in Section 5. For the concrete case of enumeration of XPs of tree XpG's, since we can enumerate all CXps in polynomial time, then we can exploit the well-known results of Fredman&Khachiyan (Fredman and Khachiyan 1996) to

Algorithm 4 Enumeration of AXps and CXps

input: XpG: $\mathcal{D} = (G_{\mathcal{D}}, S, \upsilon, \alpha_V, \alpha_E)$

procedure Enumerate($\mathcal{D}$)
  $\mathcal{H} \leftarrow \emptyset$ // $\mathcal{H}$ defined on set S
  repeat
    $(\mathsf{outc}, \mathbf{r}) \leftarrow \mathsf{SAT}(\mathcal{H})$
    if outc $=$ **true then**
      $A \leftarrow \{s_i \in S \,|\, r_i = 1\}$
      $C \leftarrow \{s_i \in S \,|\, r_i = 0\}$
      if not pathToZero($\mathcal{D}, C$) **then**
        $X \leftarrow$ findAXp($\mathcal{D}, A$)
        reportAXp(X)
        $\mathcal{H} \leftarrow \mathcal{H} \cup \{(\vee_{s_i \in X} \neg s_i)\}$
      else
        $X \leftarrow$ findCXp($\mathcal{D}, C$)
        reportCXp(X)
        $\mathcal{H} \leftarrow \mathcal{H} \cup \{(\vee_{s_i \in X} s_i)\}$
  until outc $=$ **false**

prove that enumeration of AXps can be obtained in quasi-polynomial time. The key observation is that, since we can enumerate all the CXps in polynomial time, then we can construct the associated hypergraph, thus respecting the conditions of Fredman&Khachiyan's algorithms (Fredman and Khachiyan 1996; Khachiyan et al. 2006).

Proposition 6. *For a tree XpG, the number of CXps is polynomial on the size of the XpG, and can be enumerated in polynomial time.*

Proof. To change the prediction, we must make a path to a prediction $c' \in \mathcal{K} \setminus \{c\}$ consistent. In a tree, the number of paths (connecting the root to a terminal) associated with a prediction in $c' \in \mathcal{K} \setminus \{c\}$ is linear on the size of the tree. Observe that each path yielding a prediction other that c contributes at most one CXp, because the consistency of the path (in order to predict a class other than c) requires that all the inconsistent literals be allowed to take some consistent value. We can thus conclude that the number of CXps is linear on the size of a tree XpG. The algorithm for listing the CXps exploits the previous remarks, but takes into consideration that some paths may contribute candidate CXps that are supersets of others (and so not actual CXps); these must be filtered out. □

5 Deciding Explanation Membership

To the best of our knowledge, the problem of deciding the membership of some literal in a prime implicant has not been studied in detail before. However, in the case of explainability, it is paramount to be able to answer the query of whether a feature (and assigned value) are included in some explanation. This section briefly analyzes the complexity of deciding membership in PIs. Clearly, the results for AXps/CXps track the results for PIs. For the general case of DNFs, we prove that PI membership is hard for Σ_2^{P}. For general XpG's, we show that the problem is in NP. Finally, for tree XpG's, we show that PI/AXp/CXp membership is in P. Hence, for DTs, we can decide in polynomial time

whether some feature is included in some AXp or CXp.

Proposition 7. *Deciding PI membership for a DNF is hard for* Σ_2^{P}.

Proof. [Sketch] We reduce deciding membership in a minimal unsatisfiable subset (MUS), which is known to be hard for Σ_2^{P} (Liberatore 2005), to PI testing for DNFs. Let $\varphi = \{\gamma_1, \ldots, \gamma_m\}$ be an unsatisfiable CNF formula. We want to decide whether γ_i is included in some MUS. The reduction works as follows. First, create a DNF $\neg\varphi$, which is valid. Then, define a boolean function $\psi : \mathbb{S} \to \{0, 1\}$, by conjoining a selector variable s_j with each term $\neg\gamma_j$. Clearly, $\psi(1, \ldots, 1) = 1$, and $\psi(0, \ldots, 0) = 0$. Furthermore, it is plain that any assignment to the s_i variables that selects any MUS of φ, will be a prime implicant of ψ, and vice-versa. Hence, γ_j is in some MUS of φ iff s_j is in some prime implicant of ψ. □

Proposition 8. *Deciding PI/AXp/CXp membership for an XpG* $\mathcal{D}$ *is in* NP.

Proof. To show that PI membership is in NP, we consider a concrete variable $s_i \in S$. Moreover, we guess an assignment to the variables in $S \setminus \{s_i\}$, say $\mathbf{u} \in \mathbb{S}$, such that $u_i = 1$. To decide the membership of $s_i = 1$ in the PI represented by $\mathbf{u}$, we proceed as follows:

1. Check that given the assignment $\mathbf{u}$, $\sigma_{\mathcal{D}}(\mathbf{u}) = 1$. This is done in polynomial time by running Algorithm 1 (and failing to reach a terminal node with label 0).
2. The next step is to pick each $u_j = 1$ (one of which is u_i), change its value to 0, and check that $\sigma_{\mathcal{D}}(\mathbf{u}) = 0$. This is again done by running Algorithm 1 (at most m times). This way we establish subset minimality.

Overall, we can check in polynomial time that $\mathbf{u}$ represents a prime implicant containing u_i. Hence, the membership decision problem is in NP. □

Proposition 9. *Deciding PI/AXp/CXp membership for a tree XpG is in* P.

Proof. From Proposition 6, we know that enumeration of all CXps can be achieved in polynomial time. Hence, we can simply run the algorithm outlined in the proof of Proposition 6, list all the features that occur in CXps, and decide whether a given feature is included in that list. For AXps the membership problem is also in P, simply by taking Proposition 1 into account. For PIs, the results match those of AXps. □

6 Related Work

Our work can be related with recent work on bayesian classifiers and decision graphs (Shih, Choi, and Darwiche 2018; Shih, Choi, and Darwiche 2019; Darwiche and Hirth 2020), but also languages from the knowledge compilation (KC) map (Audemard, Koriche, and Marquis 2020; Audemard et al. 2021). In addition, we build on the recent results on the interpretability and the need for explainability of DTs (Izza, Ignatiev, and Marques-Silva 2020). The algorithms described in some of the previous work (Shih, Choi, and Darwiche 2018; Shih, Choi, and Darwiche 2019; Darwiche and Hirth 2020) cover PI-explanations (and also minimum cardinality explanations, which we do not consider), but do not consider contrastive explanations. The focus of this earlier work is on ordered decision diagrams, and the proposed algorithms operate on binary features. Furthermore, the proposed algorithms are based on the compilation to some canonical representation (referred to as an ODD). If the goal is to find a few explanations, the algorithms described in this paper are essentially guaranteed to scale in practice, whereas compilation to a canonical representation is less likely to scale (e.g. see (Marques-Silva et al. 2020)). Similarly, other recent work (Audemard, Koriche, and Marquis 2020) investigates languages from the knowledge compilation map, which consider binary features. In addition, the tractable classifiers considered in (Audemard, Koriche, and Marquis 2020) for AXps do not intersect those studied in this paper. In a companion work, (Audemard et al. 2021) prove that for several XAI queries proposed in (Audemard, Koriche, and Marquis 2020), including AXp extraction, there exist polynomial algorithms for the case of DTs. In (Barceló et al. 2020), the focus is on the complexity of *smallest* PI-explanations, and the results prove its tractability for FBDDs, which generalize OBDDs and DTs. Lastly, (Van den Broeck et al. 2021; Arenas et al. 2021) show that computing SHAP explanations (Lundberg and Lee 2017) is tractable for the KC languages d-DNNFs, including FBDDs, OBDDs, DTs and SDDs (Darwiche and Marquis 2002).

7 Experimental Results

This section presents the experiments carried out to assess the practical effectiveness of the proposed algorithms. The assessment is performed on the computation of abductive (AXp) and contrastive (CXp) explanations for two case studies of DGs: OBDDs and DTs. The experiments consider a selection of datasets that are publicly available and originate from UCI Machine Learning Repository (Dua and Graff 2017), Penn Machine Learning Benchmarks (Olson et al. 2017) and openML (Vanschoren et al. 2013). These benchmarks are organized into two categories: the first category contains binary classification datasets with fully binary features, and counts 11 datasets; the second category comprises binary and multidimensional classification datasets with categorical and/or ordinal (i.e. integer or real-valued) features, and counts 34 datasets. Hence, the total number of considered datasets is 45. The subset of the binary datasets is considered for generating OBDDs, while the remaining selected datasets are used for learning DTs.

To learn OBDDs, we first train Decision List (DL) models on the given binary datasets and then compile the obtained DLs into OBDDs using the approach proposed in (Narodytska et al. 2019a). DLs are learned using Orange3 (Demšar et al. 2013), the order of rules is determined by Orange3 and the last rule is the default rule. The compilation to OBDDs is performed using BuDDy (Lind-Nielsen 1999).

A rule is of the form "IF antecedent THEN prediction", where the antecedent is a conjunction of features, and the prediction is the class variable y. The antecedent of default

Dataset	(#F	#TI)	OBDD		XPs	AXp				CXp				Runtime			
			#N	%A	avg	Mx	m	avg	%L	Mx	m	avg	%L	Tot	Mx	m	avg
corral	(6	64)	6	100	4	4	1	2	34	4	2	2	22	0.072	0.002	0.001	0.001
dbworld-bodies	(4702	62)	7	92	4	2	1	1	1	3	1	2	1	0.072	0.002	0.001	0.001
dbworld-bodies-stemmed	(3721	62)	6	84	3	3	1	1	1	4	1	2	1	0.056	0.001	0.000	0.001
dbworld-subjects	(242	63)	14	84	5	2	1	1	2	5	2	4	1	0.090	0.003	0.001	0.001
dbworld-subjects-stemmed	(229	63)	18	84	6	3	1	1	2	5	2	4	1	0.190	0.007	0.001	0.003
mofn_3_7_10	(10	251)	21	98	11	33	1	5	34	33	3	7	23	1.183	0.022	0.001	0.005
mux6	(6	64)	9	100	5	4	1	2	51	4	3	3	24	0.103	0.004	0.001	0.002
parity5+5	(10	222)	71	80	8	11	1	2	59	15	5	6	14	1.237	0.015	0.003	0.006
spect	(22	93)	284	87	11	24	1	4	22	36	1	7	14	1.726	0.074	0.007	0.019
threeOf9	(9	205)	33	95	8	16	1	3	39	18	3	5	21	0.921	0.017	0.002	0.004
xd6	(9	325)	11	100	7	18	1	4	34	27	3	3	18	0.647	0.010	0.001	0.002

Table 1: Listing all XPs (AXp's and CXp's) for OBDDs. Columns **#F** and **#TI** report, respectively, the number of features, and the number of tested instances, in the dataset. (Note that for a dataset containing more than 1000 instances, 30% of its instances, randomly selected, are used to be explained. Moreover, duplicate rows in the datasets are filtered.) Column **XPs** reports the average number of total explanations (AXp's and CXp's). Sub-Columns **#N** and **%A** show, respectively, total number of nodes and test accuracy of an OBDD. Sub-columns **Mx**, **m** and **avg** of column **AXp** (resp., **CXp**) show, respectively, the maximum , minimum and average number of explanations. The average length of an explanation (AXp/CXp) is given as **%L**. Sub-columns **Tot**, **Mx**, **m** and **avg** of column **RunTime** reports, respectively, the total, maximal, minimal and average time in second to list all the explanations for all tested instances.

Dataset	(#F	#TI)	DT			XPs	AXp				CXp				Runtime			
			D	#N	%A	avg	Mx	m	avg	%L	Mx	m	avg	%L	Tot	Mx	m	avg
adult	(12	1766)	6	83	78	8	11	1	2	41	12	2	5	13	5.76	0.010	0.001	0.003
agaricus-lepiota	(22	2437)	6	37	100	6	6	1	3	17	7	2	4	7	5.30	0.006	0.001	0.002
anneal	(38	886)	6	29	99	9	8	1	3	14	10	2	6	5	4.02	0.015	0.002	0.005
bank	(19	10 837)	6	113	88	18	38	1	9	33	21	4	9	12	87.11	0.032	0.002	0.008
cancer	(9	449)	6	37	87	7	8	1	3	39	7	2	4	21	1.12	0.006	0.001	0.003
car	(6	519)	6	43	96	4	4	1	2	39	6	1	2	24	0.71	0.004	0.001	0.001
chess	(36	959)	6	33	97	7	10	1	3	12	10	1	5	5	2.99	0.012	0.001	0.003
churn	(20	1500)	6	21	75	2	1	1	1	5	1	1	1	5	1.04	0.002	0.001	0.001
colic	(22	357)	6	55	81	11	18	1	5	23	10	3	6	8	1.31	0.011	0.001	0.004
collins	(23	485)	6	29	75	4	1	1	1	11	4	1	3	5	0.58	0.002	0.001	0.001
dermatology	(34	366)	6	33	90	7	6	1	2	14	11	1	5	4	0.97	0.007	0.001	0.003
divorce	(54	150)	5	15	90	6	8	1	3	7	4	1	3	3	0.73	0.010	0.002	0.005
dna	(180	901)	6	61	90	10	28	1	4	3	12	2	5	2	32.15	0.097	0.010	0.036
hayes-roth	(4	84)	6	23	78	3	3	1	1	54	3	1	2	27	0.06	0.001	0.000	0.001
hepatitis	(19	155)	5	17	77	6	10	1	3	18	5	2	3	10	0.29	0.004	0.001	0.002
house-votes-84	(16	298)	6	49	91	9	30	1	5	25	10	2	4	13	0.93	0.016	0.001	0.003
iris	(4	149)	5	23	90	5	3	1	2	58	4	2	3	39	0.16	0.003	0.001	0.001
irish	(5	470)	4	13	97	3	2	1	1	33	2	1	2	23	0.27	0.001	0.000	0.001
kr-vs-kp	(36	959)	6	49	96	7	23	1	4	12	8	2	4	5	3.03	0.014	0.001	0.003
lymphography	(18	148)	6	61	76	11	15	1	5	28	12	3	6	10	0.54	0.009	0.001	0.004
molecular-biology-promoters	(58	106)	6	17	86	4	6	1	2	6	5	1	2	3	0.43	0.008	0.003	0.004
monk1	(6	124)	4	17	100	3	2	1	1	38	3	1	2	18	0.11	0.002	0.000	0.001
monk2	(6	169)	6	67	82	6	7	1	2	65	9	2	5	23	0.31	0.005	0.001	0.002
monk3	(6	122)	6	35	80	4	6	1	2	45	4	2	3	23	0.15	0.004	0.001	0.001
mouse	(5	57)	3	9	83	3	4	1	1	41	3	2	2	25	0.04	0.001	0.000	0.001
mushroom	(22	2438)	6	39	100	6	5	1	2	18	7	2	4	7	5.43	0.007	0.001	0.002
new-thyroid	(5	215)	3	11	95	4	2	1	1	54	3	2	3	21	0.12	0.001	0.000	0.001
pendigits	(16	3298)	6	121	88	8	12	1	2	37	13	5	6	9	10.19	0.011	0.002	0.003
seismic-bumps	(18	774)	6	37	89	7	12	1	4	17	7	2	4	11	3.02	0.009	0.001	0.004
shuttle	(9	17 400)	6	63	99	4	4	1	1	34	6	2	3	13	33.67	0.005	0.001	0.002
soybean	(35	622)	6	63	88	7	4	1	1	15	7	3	5	4	3.12	0.012	0.002	0.005
spambase	(57	1262)	6	63	75	10	22	1	3	11	15	3	7	3	6.63	0.019	0.002	0.005
tic-tac-toe	(9	958)	6	69	93	9	13	1	4	51	12	3	6	20	2.94	0.009	0.001	0.003
zoo	(16	59)	6	23	91	5	2	1	1	24	6	2	4	9	0.12	0.003	0.001	0.002

Table 2: Listing all XPs (AXp's and CXp's) for DTs. Sub-Columns **#D** **#N** and **%A** report, respectively, tree's max depth, total number of nodes and test accuracy of a DT. The remaining columns hold the same meaning as described in the caption of Table 1.

rule is empty. Rules are translated into terms, and then conjoined into a Boolean function.

Example 9. *Given a DL* = $\{x_1 \land x_2 \rightarrow \mathbf{1}, \bar{x}_1 \land x_2 \rightarrow \mathbf{0}, \emptyset \rightarrow \mathbf{0}\}$*, it represents Boolean function:* $F_G = (x_1 \land x_2 \land y) \lor$

$(\overline{x_1 \wedge x_2} \wedge (\bar{x}_1 \wedge x_2) \wedge \bar{y}) \vee (\overline{x_1 \wedge x_2} \wedge \overline{\bar{x}_1 \wedge x_2} \wedge \bar{y})$. *After compilation, we compute $F_{G|y=1}$ on OBDD to eliminate class variable y, therefore any path ending in y (resp. $\bar{y}$) is now a path to* $\mathbf{1}$ *(resp.* $\mathbf{0}$*).*

For training DTs, we use the learning tool IAI (*Interpretable IA*) (Bertsimas and Dunn 2017; IAI 2020), which provides shallow DTs that are highly accurate. To achieve high accuracy in the DTs, the maximum depth is tuned to 6 while the remaining parameters are kept in their default set up. (Note that the test accuracy achieved for the trained classifiers, both OBDDs and DTs, is always greater than 75%).

All the proposed algorithms are implemented in Python, in the `XpG` package[5]. The PySAT package (Ignatiev, Morgado, and Marques-Silva 2018) is used to instrument incremental SAT oracle calls in XP enumeration (see Algorithm 4) and the `dd` [6] package, implemented in Python and Cython, is used to integrate BuDDy, which is implemented in C. The experiments are performed on a MacBook Pro with a 6-Core Intel Core i7 2.6 GHz processor with 16 GByte RAM, running macOS Big Sur.

Table 1 summarizes the obtained results of explaining OBDDs. (The table's caption also describes the meaning of each column.) As can be observed, the maximum running time to enumerate XPs is less than 0.074 sec for all tested XpG's in any OBDD and does not exceed 0.02 sec on average. In terms of the number of XPs, the total number of AXps and CXps per instance is relatively small. Thus the overall cost of the SAT oracle calls made for XP enumeration is negligible. In addition, these observations apply even for large OBDDs, e.g. OBDD learned from the *spect* dataset has 284 nodes and results in 11 XPs on average.

Similar observations can be made with respect to explanation enumeration for DTs, the results of which are detailed in Table 2. Exhaustive enumeration of XPs for a XpG built from a DT takes only a few milliseconds. Indeed, the largest average runtime (obtained for the *dna* dataset) is 0.036 sec. Furthermore and as can be observed, the average length %L of an XP is in general relatively small, compared to the total number of features of the corresponding dataset. Also, the total number of XPs per instance is on average less than 11 and never exceeds 18.

Although the DGs considered in the experiments can be viewed as relatively small and shallow (albeit this only reflects the required complexity given the public datasets available), the run time of the enumerator depends essentially on solving a relatively simple CNF formula ($\mathcal{H}$) which grows linearly with the number of XPs. (The run time of the actual extractors in negligible.) This suggests that the proposed algorithms will scale for significantly larger DGs, characterized also by a larger total number of XPs.

8 Conclusions

The paper introduces explanation graphs, which allow several classes of graph-based classifiers to be explained with the same algorithms. These algorithms allow for a single abductive or a single contrastive explanation to be computed in polynomial time, and enumeration of explanations to be achieved with a single call to a SAT oracle per computed explanation. The paper also relates the evaluation of explanation graphs with monotone functions. In addition, the paper proves that for decision trees, computing all contrastive explanations and deciding feature membership in some explanation can be solved in polynomial time. The experimental results demonstrate the practical effectiveness of the ideas proposed in the paper.

Future work will investigate how the results in this paper can be extended to other classes of classifiers, by building on this but also on other recent related works (Huang et al. 2021; Audemard et al. 2021; Izza et al. 2021).

[5]https://github.com/yizza91/xpg

[6]https://github.com/tulip-control/dd

Acknowledgments

This work was 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

Appuswamy, R.; Franceschetti, M.; Karamchandani, N.; and Zeger, K. 2011. Network coding for computing: Cut-set bounds. *IEEE Trans. Inf. Theory* 57(2):1015–1030.

Arenas, M.; Barceló, P.; Bertossi, L. E.; and Monet, M. 2021. The tractability of SHAP-score-based explanations for classification over deterministic and decomposable boolean circuits. In *AAAI*, 6670–6678.

Audemard, G.; Bellart, S.; Bounia, L.; Koriche, F.; Lagniez, J.; and Marquis, P. 2021. On the computational intelligibility of boolean classifiers. *CoRR* abs/2104.06172.

Audemard, G.; Koriche, F.; and Marquis, P. 2020. On tractable XAI queries based on compiled representations. In *KR*.

Bahl, L. R.; Brown, P. F.; de Souza, P. V.; and Mercer, R. L. 1989. A tree-based statistical language model for natural language speech recognition. *IEEE Trans. Acoust. Speech Signal Process.* 37(7):1001–1008.

Barceló, P.; Monet, M.; Pérez, J.; and Subercaseaux, B. 2020. Model interpretability through the lens of computational complexity. In *NeurIPS*.

Bergman, D.; Ciré, A. A.; van Hoeve, W.; and Hooker, J. N. 2016. *Decision Diagrams for Optimization*. Springer.

Bertsimas, D., and Dunn, J. 2017. Optimal classification trees. *Mach. Learn.* 106(7):1039–1082.

Breiman, L. 2001. Statistical modeling: The two cultures. *Statistical science* 16(3):199–231.

Brodley, C. E., and Utgoff, P. E. 1995. Multivariate decision trees. *Mach. Learn.* 19(1):45–77.

Bryant, R. E. 1986. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Computers* 35(8):677–691.

Cabodi, G.; Camurati, P. E.; Ignatiev, A.; Marques-Silva, J.; Palena, M.; and Pasini, P. 2021. Optimizing binary decision diagrams for interpretable machine learning classification. In *DATE*.

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.

Chinneck, J. W., and Dravnieks, E. W. 1991. Locating minimal infeasible constraint sets in linear programs. *INFORMS J. Comput.* 3(2):157–168.

Crama, Y., and Hammer, P. L. 2011. *Boolean Functions - Theory, Algorithms, and Applications*. Cambridge University Press.

Darwiche, A., and Hirth, A. 2020. On the reasons behind decisions. In *ECAI*, 712–720.

Darwiche, A., and Marquis, P. 2002. A knowledge compilation map. *J. Artif. Intell. Res.* 17:229–264.

Darwiche, A. 2009. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press.

Demšar, J.; Curk, T.; Erjavec, A.; Črt Gorup; Hočevar, T.; Milutinovič, M.; Možina, M.; Polajnar, M.; Toplak, M.; Starič, A.; Štajdohar, M.; Umek, L.; Žagar, L.; Žbontar, J.; Žitnik, M.; and Zupan, B. 2013. Orange: Data mining toolbox in python. *Journal of Machine Learning Research* 14:2349–2353.

Dimanov, B.; Bhatt, U.; Jamnik, M.; and Weller, A. 2020. You shouldn't trust me: Learning models which conceal unfairness from multiple explanation methods. In *ECAI*, 2473–2480.

Dua, D., and Graff, C. 2017. UCI machine learning repository. http://archive.ics.uci.edu/ml. University of California, Irvine.

Duda, R. O.; Hart, P. E.; and Stork, D. G. 2001. *Pattern classification, 2nd Edition*. Wiley.

Falappa, M. A.; Kern-Isberner, G.; and Simari, G. R. 2002. Explanations, belief revision and defeasible reasoning. *Artif. Intell.* 141(1/2):1–28.

Fredman, M. L., and Khachiyan, L. 1996. On the complexity of dualization of monotone disjunctive normal forms. *J. Algorithms* 21(3):618–628.

Freitas, A. A. 2013. Comprehensible classification models: a position paper. *SIGKDD Explorations* 15(1):1–10.

Goldsmith, J.; Hagen, M.; and Mundhenk, M. 2005. Complexity of DNF and isomorphism of monotone formulas. In *MFCS*, 410–421.

Goldsmith, J.; Hagen, M.; and Mundhenk, M. 2008. Complexity of DNF minimization and isomorphism testing for monotone formulas. *Inf. Comput.* 206(6):760–775.

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

Gunning, D., and Aha, D. W. 2019. Darpa's explainable artificial intelligence (XAI) program. *AI Mag.* 40(2):44–58.

Gurvich, V., and Khachiyan, L. 1999. On generating the irredundant conjunctive and disjunctive normal forms of monotone boolean functions. *Discret. Appl. Math.* 96-97:363–373.

Huang, X.; Izza, Y.; Ignatiev, A.; Cooper, M. C.; Asher, N.; and Marques-Silva, J. 2021. Efficient explanations for knowledge compilation languages. *CoRR* abs/2107.01654.

IAI. 2020. Interpretable AI. https://www.interpretable.ai/.

Ignatiev, A., and Marques-Silva, J. 2021. SAT-based rigorous explanations for decision lists. *CoRR* abs/2105.06782.

Ignatiev, A.; Narodytska, N.; Asher, N.; and Marques-Silva, J. 2020. From contrastive to abductive explanations and back again. In *AIxIA*, 335–355.

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

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. 2020. Towards trustable explainable AI. In *IJCAI*, 5154–5158.

Izza, Y., and Marques-Silva, J. 2021. On explaining random forests with SAT. *CoRR* abs/2105.10278.

Izza, Y.; Ignatiev, A.; Narodytska, N.; Cooper, M. C.; and Marques-Silva, J. 2021. Efficient explanations with relevant sets. *CoRR* abs/2106.00546.

Izza, Y.; Ignatiev, A.; and Marques-Silva, J. 2020. On explaining decision trees. *CoRR* abs/2010.11034.

Jensen, F. V. 2001. *Bayesian Networks and Decision Graphs*. Springer.

Junker, U. 2004. QUICKXPLAIN: preferred explanations and relaxations for over-constrained problems. In *AAAI*, 167–172.

Kam, T. Y.-k., and Brayton, R. K. 1990. Multi-valued decision diagrams. Technical Report UCB/ERL M90/125, University of California Berkeley.

Khachiyan, L.; Boros, E.; Elbassioni, K. M.; and Gurvich, V. 2006. An efficient implementation of a quasi-polynomial algorithm for generating hypergraph transversals and its application in joint generation. *Discret. Appl. Math.* 154(16):2350–2372.

Kohavi, R. 1994. Bottom-up induction of oblivious read-once decision graphs: Strengths and limitations. In *AAAI*, 613–618.

Lakkaraju, H., and Bastani, O. 2020. "how do I fool you?": Manipulating user trust via misleading black box explanations. In *AIES*, 79–85.

Liberatore, P. 2005. Redundancy in logic I: CNF propositional formulae. *Artif. Intell.* 163(2):203–232.

Liffiton, M. H., and Sakallah, K. A. 2008. Algorithms for computing minimal unsatisfiable subsets of constraints. *J. Autom. Reason.* 40(1):1–33.

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

Lind-Nielsen, J. 1999. Buddy : A binary decision diagram package. http://buddy.sourceforge.net.

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.

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. *CoRR* abs/2106.00154.

Marques-Silva, J.; Janota, M.; and Belov, A. 2013. Minimal sets over monotone predicates in boolean formulae. In *CAV*, 592–607.

Marques-Silva, J.; Janota, M.; and Mencía, C. 2017. Minimal sets on propositional formulae. problems and reductions. *Artif. Intell.* 252:22–50.

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

Molnar, C. 2019. *Interpretable Machine Learning*. https://christophm.github.io/interpretable-ml-book/.

Montavon, G.; Samek, W.; and Müller, K. 2018. Methods for interpreting and understanding deep neural networks. *Digit. Signal Process.* 73:1–15.

Mues, C.; Baesens, B.; Files, C. M.; and Vanthienen, J. 2004. Decision diagrams in machine learning: an empirical study on real-life credit-risk data. *Expert Syst. Appl.* 27(2):257–264.

Narodytska, N.; Ryzhyk, L.; Ganichev, I.; and Sevinc, S. 2019a. BDD-based algorithms for packet classification. In *FMCAD*, 64–68.

Narodytska, N.; Shrotri, A. A.; Meel, K. S.; Ignatiev, A.; and Marques-Silva, J. 2019b. Assessing heuristic machine learning explanations with model counting. In *SAT*, 267–278.

Oliveira, A. L., and Sangiovanni-Vincentelli, A. L. 1996. Using the minimum description length principle to infer reduced ordered decision graphs. *Mach. Learn.* 25(1):23–50.

Oliver, J. J. 1992. Decision graphs – an extension of decision trees. Technical Report 92/173, Monash University.

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

Pérez, R. P., and Uzcátegui, C. 2003. Preferences and explanations. *Artif. Intell.* 149(1):1–30.

Reiter, R. 1987. A theory of diagnosis from first principles. *Artif. Intell.* 32(1):57–95.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016a. Model-agnostic interpretability of machine learning. *CoRR* abs/1606.05386.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016b. "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.

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

Samek, W.; Montavon, G.; Vedaldi, A.; Hansen, L. K.; and Müller, K., eds. 2019. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer.

Shanahan, M. 1989. Prediction is deduction but explanation is abduction. In *IJCAI*, 1055–1060.

Shih, A.; Choi, A.; and Darwiche, A. 2018. A symbolic approach to explaining bayesian network classifiers. In *IJCAI*, 5103–5111.

Shih, A.; Choi, A.; and Darwiche, A. 2019. Compiling bayesian network classifiers into decision graphs. In *AAAI*, 7966–7974.

Silva, A.; Gombolay, M. C.; Killian, T. W.; Jimenez, I. D. J.; and Son, S. 2020. Optimization methods for interpretable differentiable decision trees applied to reinforcement learning. In *AISTATS*, 1855–1865.

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.

Srinivasan, A.; Kam, T.; Malik, S.; and Brayton, R. K. 1990. Algorithms for discrete function manipulation. In *ICCAD*, 92–95.

Utgoff, P. E.; Berkman, N. C.; and Clouse, J. A. 1997. Decision tree induction based on efficient tree restructuring. *Mach. Learn.* 29(1):5–44.

Valdes, G.; Luna, J. M.; Eaton, E.; Simone II, C. B.; Ungar, L. H.; and Solberg, T. D. 2016. MediBoost: a patient stratification tool for interpretable decision making in the era of precision medicine. *Nature Scientific Reports* 6(1):37854.

Van den Broeck, G.; Lykov, A.; Schleich, M.; and Suciu, D. 2021. On the tractability of SHAP explanations. In *AAAI*, 6505–6513.

Vanschoren, J.; van Rijn, J. N.; Bischl, B.; and Torgo, L. 2013. OpenML: networked science in machine learning. *SIGKDD Explorations* 15(2):49–60.

Wegener, I. 2000. *Branching Programs and Binary Decision Diagrams*. SIAM.

Weld, D. S., and Bansal, G. 2019. The challenge of crafting intelligible intelligence. *Commun. ACM* 62(6):70–79.

Xu, F.; Uszkoreit, H.; Du, Y.; Fan, W.; Zhao, D.; and Zhu, J. 2019. Explainable AI: A brief survey on history, research areas, approaches and challenges. In *NLPCC*, 563–574.