Efficient Interpretability Reasoning for Decision Trees

Joao Marques-Silva[1,*,†], Alexey Ignatiev[2,†]

[1]ICREA & University of Lleida), Catalunya, Spain
[2]Monash University, Melbourne, Australia

Abstract

When studying machine learning (ML) models induced from the same training data, one query of interest is predictive equivalence, i.e. to decide whether two models compute the same function. By keeping a single representative of the ML models computing the same function, one removes unwanted redundancy from the Rashomon set associated with the given training data. Furthermore, it is often paramount to answer queries about ML models when not all inputs are known, i.e. when there exists missing data. Examples of relevant queries include prediction sufficiency, but also several other queries related to explainable artificial intelligence (XAI). This paper shows that most of these queries related with reasoning about decision trees can be solved in polynomial time. Furthermore, the paper proves a lower bound on the complexity of enumeration of abductive explanations also in the case of DTs.

Keywords

XAI, Symbolic XAI, Constraint-based enumeration

1. Introduction

In decision tree (DT) learning, the Rashomon set [1, 2] captures the set of DTs for a given classification problem which are within some distance from some accepted definition of optimal DT [2]. Recent years have witnessed renewed interest in Rashomon sets, and several practical uses have been studied [3, 4, 5, 6, 2, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]. For the same classification problem, different decision trees capture different classification functions, and it is often of interest to study a sample of the decision trees in the Rashomon set. It has been shown that approximations of Rashomon sets can contain a significant fraction of DTs implementing the same function [17], i.e. DTs that are predictive equivalent. Hence, one query studied by McTavish et al. is to decide predictive equivalence. Besides predictive equivalence, answering different queries in the presence of missing data represents another class of computational problems studied by McTavish et al. One example is to decide prediction sufficiency, i.e. whether the specified inputs suffice to decide the prediction. Finally, the work of [17] also studied the removal of explanation redundancy in DTs, a topic also studied in earlier works [18, 19, 20].

At the core of the solutions proposed in [17] for those computational problems, i.e. deciding predictive equivalence, reasoning about missing data and removing explanation redundancy, is the well-known method of Quine-McCluskey (QM) [21, 22, 23] for simplifying propositional formulas. However, the QM method solves a problem that is well-known to be hard for the second level of the polynomial hierarchy [24, 25], and requires worst-case exponential space and time on the size of the given DTs. Concretely, the first step of the QM method is the computation of all prime implicants and minterms, which are worst-case exponential even in the case of DTs. The second step of the QM method is a set-covering step, which is NP-hard, and so worst-case exponential with known algorithms.

Contributions. This paper makes the following contributions. First, the paper proves that the computational problems studied in [17] can all be solved in polynomial time. Thus, the algorithms proposed in this paper are shown to be exponentially more efficient than the solutions proposed in [17], in theory and in practice. Second, the paper proposes polynomial-time algorithms for solving several

Joint Proceedings of the Workshops and Doctoral Consortium of the 42nd International Conference on Logic Programming (ICLP 2026), co-located with the Federated Logic Conference (FLoC 2026), July 18–19 and 24–25, 2026, Lisbon, Portugal.
jpms@icrea.cat (J. Marques-Silva); alexey.ignatiev@monash.edu (A. Ignatiev)
http://jpmarquessilva.github.io (J. Marques-Silva); https://alexeyignatiev.github.io/ (A. Ignatiev)
0000-0002-6632-3086 (J. Marques-Silva); 0000-0002-4535-2902 (A. Ignatiev)
© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-11-03

CEUR-WS.org/Vol-4311/RCRA_regular_05.pdf

XAI-related queries, in addition to those studied in [17]. The proposed algorithms for answering XAI-related queries are applicable even when there exists missing data in the ML model's input. Third, the paper proves a lower bound on the delay of enumerating abductive explanations in the case of DTs, which makes it unlikely that polynomial delay algorithms might exist for enumerating abductive explanations [26]. Finally, the experiments confirm the performance gap that the algorithms proposed in this paper attain when compared with recent work [17].

Organization. Section 2 introduces the notation and core definitions used throughout the paper. Section 3 proposes new definitions of logic-based explanations to account for missing data. Section 4 briefly overviews recent work on Rashomon sets, prediction sufficiency with missing data, and XAI queries. Section 5 proposes a simple polynomial-time algorithm for predictive equivalence of DTs, and argues that tractability of predictive equivalence holds for other families of ML models. Section 6 investigates several XAI queries in the context of missing data, and proves that most are solved in polynomial time. Section 7 proves that polynomial delay enumeration of AXps, even without missing data, is as hard as hypergraph transversals. Section 8 summarizes experiments showing that the use of QM proposed in earlier work [17] does not scale in practice. Finally, the paper concludes in Section 9.

2. Notation & Definitions

Classification models. A classification problem is defined on a set of features $\mathcal{F} = \{1, \ldots, m\}$. Each feature i has a domain $\mathbb{D}_i$. Domains can be categorical or ordinal. If ordinal, domains can be integer or real-valued. Feature space is defined as the cartesian product of the domains, $\mathbb{F} = \mathbb{D}_1 \times \cdots \times \mathbb{D}_m$. A classifier computes a non-constant classification function κ, that maps feature space to a set of classes $\mathbb{K} = \{c_1, \ldots, c_K\}$, i.e. $\kappa : \mathbb{F} \to \mathbb{K}$. Given a feature i, x_i represents a variable that takes values from $\mathbb{D}_i$, whereas v_i represents a constant taken from $\mathbb{D}_i$. An instance is a pair $(\mathbf{v}, c)$, with $\mathbf{v} \in \mathbb{F}$ and $c \in \mathbb{K}$, such that $\kappa(\mathbf{v}) = c$. A classification model is defined as a tuple $C = (\mathcal{F}, \mathbb{F}, \mathbb{K}, \kappa)$. (The domains of features are left implicit for simplicity.) Although the paper studies classification models, the same theoretical framework can be used to analyze regression problems [27, 28].

Partial assignments & consistency. Let $\mathbb{OP}$ denote some universe of relational operators. In this paper, we consider $\mathbb{OP} = \mathbb{OP}_s \cup \mathbb{OP}_t$, with $\mathbb{OP}_s = \{=, <, \leq, \geq, >\}$ and $\mathbb{OP}_t = \{\in\}$. (Logic negation can be used to obtain additional operators.) Let $\mathbb{L} = \{(x_i, R_i, \mathsf{op}_i) \,|\, (i \in \mathcal{F}) \land (R_i \subseteq \mathbb{D}_i) \land (\mathsf{op}_i \in \mathbb{OP}_t)\} \cup \{(x_i, R_i, \mathsf{op}_i) \,|\, (i \in \mathcal{F}) \land (R_i \in \mathbb{D}_i) \land (\mathsf{op}_i \in \mathbb{OP}_s)\}$ denote all the literals that can be obtained from the variables associated with the features in $\mathcal{F}$ and subsets or values from their domains. (Throughout the paper, we will simplify the notation. For example, (x_i, v_i) will be used instead of $(x_i, v_i, =)$, and both represent the literal $x_i = v_i$. Moreover, $(x_i, v_i, =)$ is equivalent to writing $(x_i, \{v_i\}, \in)$.) The set of *partial assignments* is defined by the powerset of $\mathbb{L}$, i.e. $\mathbb{A} = 2^{\mathbb{L}}$. Thus, a partial assignment $\mathcal{A} \in \mathbb{A}$ is a subset of $\mathbb{L}$. In the case of complete assignments (or inputs to an ML model), we require that all features are assigned a single value from their domain, $\mathbb{I} = \{\mathcal{A} \in \mathbb{A} \,|\, \forall (i \in \mathcal{F}).|\{(x_i, R_i, \mathsf{op}_i) \in \mathcal{A}\}| = 1 \land (x_i, v_i, =) \in \mathcal{A}\}$. A point $\mathbf{v} = (v_1, \ldots, v_m)$ in feature space can be viewed as a partial assignment:

$$A(\mathbf{v}) \coloneqq \{(x_1, v_1, =), \ldots, (x_m, v_m, =)\}$$

Finally, the features associated with a partial assignment $\mathcal{A}$ are defined as follows,

$$\mathsf{Feat}(\mathcal{A}) \coloneqq \{i \in \mathcal{F} \,|\, (x_i, R_i, \mathsf{op}_i) \in \mathcal{A} \setminus \{(x_i, \mathbb{D}_i, \in) \,|\, i \in \mathcal{F}\}\}$$

With each literal $(x_i, R_i, \mathsf{op}_i) \in \mathbb{L}$ we associate a domain $\mathsf{dom}((x_i, R_i, \mathsf{op}_i)) = \{\mathbf{x} = (x_1, \ldots, x_i, \ldots, x_m) \in \mathbb{F} \,|\, x_i \,\mathsf{op}_i\, R_i\}$. Thus, the domain of a literal is the set of points in feature space for which the literal holds true. Similarly, we define the domain of a partial assignment $\mathcal{A}$, i.e. $\mathsf{dom}(\mathcal{A}) = \cap_{(x_i, v_i, \mathsf{op}_i)} \mathsf{dom}(x_i, v_i, \mathsf{op}_i)$. In addition, a partial assignment is said to be *consistent*, if it

exhibits a non-empty domain:

$$\mathsf{Consistent}(\mathcal{A}) \;\coloneqq [\mathsf{dom}(\mathcal{A}) \neq \emptyset]$$

By default, we will expect partial assignments to be consistent. Consistency is also defined with respect to pairs of partial assignments:

$$\mathsf{Consistent}(\mathcal{A}_1, \mathcal{A}_2) \;\coloneqq [\mathsf{dom}(\mathcal{A}_1) \cap \mathsf{dom}(\mathcal{A}_2) \neq \emptyset]$$

Section 3 describes how partial assignments provide a suitable framework for formalizing explanations in DTs. It should be noted that (prime) implicants have also been represented as partial assignments in the past [29, 30].

Decision trees. A DT is a connected directed acyclic graph $\mathcal{T} = (V, E)$, with a single root node $r \in V$. $r = \mathsf{Root}(\mathcal{T})$ indicates that $r \in V$ is the root of DT $\mathcal{T}$. The tree nodes V are partitioned into non-terminal and terminal nodes. Each non-terminal node s (such that predicate $\mathsf{NonTerminal}(s; \mathcal{T})$ holds true)[1] is associated with a feature i, i.e. $i = \mathsf{Feature}(s; \mathcal{T})$. For a non-terminal node s, associated with feature i, the domain of i is partitioned among the children of node s. Each terminal node s (such that predicate $\mathsf{Terminal}(s; \mathcal{T})$ holds true) is associated with a class from $\mathbb{K}$, denoted by $\mathsf{Class}(s; \mathcal{T})$. Whereas non-terminal nodes have two or more children, terminal nodes have no children. Moreover, for each node s in the DT, $\mathsf{Parent}(s; \mathcal{T})$ represents the parent of s in the tree. For the root node $r = \mathsf{Root}(\mathcal{T})$, $\mathsf{Parent}(r; \mathcal{T}) = \mathsf{none}$. $\mathsf{Children}(s, \mathcal{T})$, for a non-terminal s, denotes the children nodes of s. A DT computes a classification function $\kappa_{\mathcal{T}}$, mapping feature space to $\mathbb{K}$. For convenience, $\mathsf{NonTerminal}(\mathcal{T})$ denotes the nodes in V that are non-terminal. Similarly, $\mathsf{Terminal}(\mathcal{T})$ denotes the nodes in V that are terminal.

A path P in a DT is a sequence of nodes $P = \langle s_1, \ldots, s_r \rangle$, such that $s_1 = \mathsf{Root}(\mathcal{T})$, $s_j = \mathsf{Parent}(s_{j+1}; \mathcal{T})$, $j = 1, \ldots, r$, and $\mathsf{Terminal}(s_r, ; \mathcal{T})$ holds true. Moreover, each pair (s_j, s_{j+1}) represents both an edge of P and of $\mathcal{T}$, i.e. $(s_j, s_{j+1}) \in E$. We will use $\mathsf{Terminal}(P; \mathcal{T})$ to denote the terminal node of path P in DT $\mathcal{T}$. The paths of DT $\mathcal{T}$ are denoted by $\mathsf{Paths}(\mathcal{T})$. Moreover, if (s_p, s_q) is an edge of $\mathcal{T}$, then $\mathsf{Literal}(s_p, s_q; \mathcal{T})$ represents the literal associated with $\mathsf{Feature}(s_p; \mathcal{T})$ and edge (s_p, s_q) on the path of $\mathcal{T}$ containing that edge. For a path $P \in \mathsf{Paths}(\mathcal{T})$, $\mathsf{Literals}(P) = \{\mathsf{Literal}(s_p, s_q; \mathcal{T}) \,|\, (s_p, s_q) \in P\}$. We note that $\mathsf{Literals}(P)$ represents a partial assignment and so the above definition of consistency also applies to the literals in a DT path.

In this paper, we assume the following conditions regarding DTs: (i) any path in the DT is consistent; (ii) any point in feature space is consistent with at least one tree path; and (iii) any point in feature space that is consistent with some tree path must not be consistent with any other tree path.

Two DTs $\mathcal{T}$ and $\mathcal{T}'$ are *predictive equivalent* if it holds that: $\forall(\mathbf{x} \in \mathbb{F}).\, (\kappa_{\mathcal{T}}(\mathbf{x}) = \kappa_{\mathcal{T}'}(\mathbf{x}))$. Thus, predictive equivalence signifies that the two DTs cannot be distinguished over any point in feature space. For any pair of DTs $\mathcal{T}_1, \mathcal{T}_2$, the predicate $\mathsf{PredictiveEquivalent}(\mathcal{T}_1, \mathcal{T}_2)$ takes value $\top$ (i.e. it holds true) if the pair of DTs is predictive equivalent; otherwise it takes value $\bot$. Although the paper focuses mostly on DTs, it is plain that the definition of predictive equivalence can be used to relate two arbitrary ML models. For example, we could have a DT that is predictive equivalent to some random forest (RF) [31]. In Section 5 we revisit predictive equivalence in this more general setting.

Running example. The DT shown in Figure 1 will be used as the running example throughout the paper. The running example illustrates the definitions introduced earlier in this section.

Prediction sufficiency. A consistent partial assignment $\mathcal{A}_S$ is *sufficient for* (or entails) for some prediction if,

$$\exists(c \in \mathbb{K})\forall(\mathbf{x} \in \mathbb{F}).\; \left(\bigwedge\nolimits_{(x_i, R_i, \mathsf{op}_i) \in \mathcal{A}_S} x_i \,\mathsf{op}_i\, R_i\right) \rightarrow (\kappa(\mathbf{x}) = c) \tag{1}$$

In this case, we say that $\mathcal{A}_S$ is sufficient for class c. Thus, given $\mathcal{A}_S$, the prediction is c *independently* of the values assigned to the features in $\mathcal{F} \setminus \mathsf{Feat}(\mathcal{A}_S)$.

[1] For predicates, parameterizations are shown after ';', but will be omitted for simplicity.

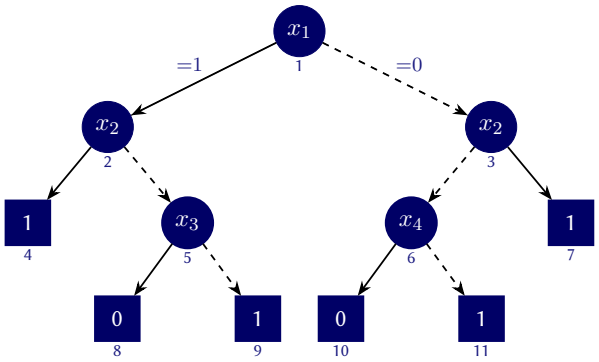


Figure 1:
Example DT $\mathcal{T}$, with $\mathcal{F} = \{1,2,3,4\}$, $\mathbb{D}_1 = \mathbb{D}_2 = \mathbb{D}_3 = \mathbb{D}_4 = \{0,1\}$, $\mathbb{K} = \{0,1\}$. All literals in the tree are of one of two forms: $(x_i, 0, =)$ or $(x_i, 1, =)$, with $i = \{1,2,3,4\}$. The cases $(x_i, 0, =)$ are shown with a dashed line, and the cases $(x_i, 1, =)$ are shown with a solid line. Also, $V = \{1, \ldots, 11\}$, $\mathsf{Root}(\mathcal{T}) = 1$, $\mathsf{Terminal}(\mathcal{T}) = \{4, 7, 8, 9, 10, 11\}$, $\mathsf{NonTerminal}(\mathcal{T}) = \{1, 2, 3, 5, 6\}$, $\mathsf{Paths}(\mathcal{T}) = \{\langle 1, 2, 4\rangle, \langle 1, 2, 5, 8\rangle, \langle 1, 2, 5, 9\rangle, \langle 1, 3, 6, 10\rangle, \langle 1, 3, 6, 11\rangle, \langle 1, 3, 7\rangle\}$.

Missing data. A topic studied in the recent work of [17] is that of *missing data*, where the values of some features are unknown.[2] In the presence of missing data, we are interested in deciding whether a prediction can be uniquely determined. This amounts to deciding whether the partial assignment to the features with known values is sufficient for some prediction. In addition, the partial assignment is required to be consistent. If that consistent partial assignment is sufficient for the prediction, then we will be interested in several XAI queries; these are detailed in Section 6.

To reason about missing data, we allow the domain of each feature $i \in \mathcal{F}$ to be extended with a value $\mathfrak{u}$, different from any value in $\mathbb{D}_i$, and which will signify that the value of i is missing (and so it is unknown). If a feature i is assigned value $\mathfrak{u}$, this corresponds to a literal of the form $(x_i, \mathbb{D}_i, \in)$, i.e. feature i can take *any* value from its domain. Alternatively, we can simply use partial assignments, where missing data is not included in the partial assignment.

Example 1. *For the DT in Figure 1, we will consider the following input, with missing data:* $\mathcal{A}_S = \{(x_2, 1, =), (x_3, 1, =)\}$. *Thus, the value of features 1 and 4 are unknown. This partial assignment can also be represented by the point* $(\mathfrak{u}, 1, 1, \mathfrak{u})$. *By inspection it is clear that the partial assignment* $\mathcal{A}_S$ *is both consistent and sufficient for predicting class 1.*

3. Logic-Based Explanations & Missing Data

Past work on logic-based explainability mostly studied instance-based explanations [32], where the instance is a pair representing a point in feature space and the associated prediction. As a result, explanations were computed starting from the set of all features $\mathcal{F}$. In this paper we will assume instead sufficient consistent partial assignments as the starting step for reasoning about explanations, thereby generalizing past work on logic-based explanations. Nevertheless, to account for missing data, we will need to introduce changes to the common definitions of abductive and contrastive explanations [32].

Abductive explanations (AXps). AXps have been proposed in a number of works [33, 26], and represent an example of formally defined explanations. An AXp answers a "Why (the prediction)?" question, and can be interpreted as a logic rule. Given a sufficient consistent partial assignment $\mathcal{A}_S$ for prediction c, let $S = \mathsf{Feat}(\mathcal{A}_S)$, $X \subseteq S$ and let,

$$\mathsf{Lits}(\mathcal{A}_S, X) \coloneqq \{(x_i, R_i, \mathsf{op}_i) \in \mathcal{A}_S \,|\, i \in X\}$$

Clearly, $X = \mathsf{Feat}(\mathsf{Lits}(\mathcal{A}_S, X))$. A *weak abductive explanation* (WAXp) is a subset of S, such that the corresponding literals also entail the prediction c:

$$X \subseteq \mathsf{Feat}(\mathcal{A}_S) \land \forall(\mathbf{x} \in \mathbb{F}).$$
$$\left(\bigwedge\nolimits_{(x_i, R_i, \mathsf{op}_i) \in \mathsf{Lits}(\mathcal{A}_S, X)} x_i \,\mathsf{op}_i\, R_i\right) \rightarrow (\kappa(\mathbf{x}) = c) \tag{2}$$

[2] Missing data has been studied ubiquitously in the field of ML; [17] initiated its study in the context of XAI.

[3] In addition, a consistent partial assignment is sufficient for some prediction iff the set of features is a WAXp. This generalizes the fact that any $\mathbf{v} \in \mathbb{F}$ is sufficient for $\kappa(\mathbf{v})$, and so $\mathcal{F}$ is a WAXp. Subset-minimal WAXps are referred to as *abductive explanations* (AXps). Accordingly, predicate $\mathsf{AXp}(X; \mathcal{A}_S, c, M)$ is defined to hold true if X is an AXp. WAXps (resp. AXps) are also referred to as sufficient (resp. minimal) reasons for a prediction [34, 35].

Example 2. *For the example DT of Figure 1 and the partial assignment of Example 1, it is apparent by inspection that $X = \{2\}$ is a (W)AXp, i.e. if the value of feature 2 is fixed to value 1, then the prediction is guaranteed to be 1, independently of the values of the other features.*

Contrastive explanations (CXps). CXps answer a "Why Not (some other prediction)?" question, and identify the conditions that can be used to change the prediction [36, 32].[4]

Let $\mathcal{A}_S$ denote a sufficient consistent partial assignment for some class c, and let $S = \mathsf{Feat}(\mathcal{A}_S)$. Moreover, let U denote the features with unknown values, i.e. $U = \mathcal{F} \setminus S$. To define *weak contrastive explanations* WCXps) with missing data, we need to decide how to handle the features in U. One solution would be to treat the features in U as existential variables, and so we would handle U the same way as the features in the WCXp, but without U being accounted for. One downside of such a definition of WCXp is that it would ascribe specific values to the features in U as required by the witness for the WCXp, and these values might not be realistic for the given model, or contradict what one might expect from the missing data. As a result, in this paper we opt for a stricter definition of WCXp, one that ensures *independence* from the possible values assigned to the features with unknown values. From U, we define the following set:

$$\mathbb{U} \coloneqq \Pi_{j=1}^{m} \mathrm{ITE}(j \in U; \mathbb{D}_j, \{\mathfrak{u}\})$$

i.e. $\mathbb{U}$ ranges over all possible domain values for the features that are unspecified in $\mathcal{A}_S$. Next, for a given $\mathbf{u} \in \mathbb{U}$, we define a subset of feature space, as follows:

$$\mathbb{F}(\mathbf{u}; U) \coloneqq \{\mathbf{x} \in \mathbb{F} \,|\, \forall(j \in U).(x_j = u_j)\}$$

i.e. we allow any point in feature space where the features in U are fixed to the values specified in $\mathbf{u}$.

Example 3. *For the running example Figure 1 and the partial assignment from Example 1, with $U = \{1, 4\}$, we get,*

$$\mathbb{U} = \mathbb{D}_1 \times \{\mathfrak{u}\} \times \{\mathfrak{u}\} \times \mathbb{D}_4$$

And an example is $\mathbf{u} = (0, \mathfrak{u}, \mathfrak{u}, 1)$. Thus, in this case we get,

$$\begin{aligned}\mathbb{F}((0, \mathfrak{u}, \mathfrak{u}, 1); \{1, 4\}) = \{&(0, 0, 0, 1), (0, 0, 1, 1),\\ &(0, 1, 0, 1), (0, 1, 1, 1)\}\end{aligned}$$

A WCXp is a subset of S, such that the corresponding literals (and the literals not occurring in the reference consistent sufficient partial assignment) suffice to change the prediction to a class other than c:

$$\begin{aligned}&Y \subseteq \mathsf{Feat}(\mathcal{A}_S) \land \forall(\mathbf{u} \in \mathbb{U}) \exists(\mathbf{x} \in \mathbb{F}(\mathbf{u}; U)).\\ &\left(\bigwedge\nolimits_{(x_i, R_i, \mathsf{op}_i) \in \mathsf{Lits}(\mathcal{A}_S, S \setminus Y)} x_i \,\mathsf{op}_i\, R_i\right) \land (\kappa(\mathbf{x}) \neq c)\end{aligned} \tag{3}$$

Thus, if we fix the features of the literals in $\mathcal{A}_S$ except for those in Y, then there exists at least one point in feature space such that the prediction is a class other than c, independently of the value assigned to the features not specified in $\mathcal{A}_S$. As shown in Section 6, the decision to universally quantify the features in U does not impact the complexity in the case of DTs. Predicate $\mathsf{WCXp}(Y; \mathcal{A}_S, c, M)$ is defined to hold true when (3) holds. A subset-minimal WCXp is referred to as a contrastive explanation (CXp). Accordingly, predicate $\mathsf{CXp}(Y; \mathcal{A}_S, c, M)$ is defined to hold true Y is a CXp.

[3] For predicates, parameterizations are shown after ';', but will be omitted for simplicity.

[4] Contrastive explanations were introduced informally in earlier work [37].

Example 4. *For the example DT of Figure 1 and the partial assignment of Example 1, i.e.* $\{(x_2, 1, =), (x_3, 1, =)\}$*, we can observe by inspection that by picking values for the features in* $U = \{1, 4\}$*, there are assignments for which the prediction does not change. Hence, we declare that there are no CXps, i.e. without knowing and changing the value of some of the features with unknown values, we are unable to guarantee a change of prediction.*
Now, let us consider a different partial assignment $\mathcal{A}_S = \{(x_2, 1, =), (x_3, 1, =), (x_4, 1, =)\}$*. In this case, for any value assigned to feature 1, we are able to change the prediction, by changing the value of feature 2. Hence,* $\{2\}$ *is a CXp for the given partial assignment.*

Duality of explanations. As illustrated by the examples above, minimal hitting set duality between AXps and CXps does not hold in the presence of missing data. This is a direct consequence of the proposed universal quantification of the features in U. The alternative definition, hinted above, would ensure minimal hitting set duality. (Minimal hitting set duality has been shown to be a fairly robust concept, finding a wide range of applications since the seminal work of Reiter [38] (also, see [39, 40] for additional examples). Given the proposed definition of WCXps, explanations in the presence of missing data represent one exception.)

Example 5. *Regarding Examples 2 and 4, for the partial assignment* $\{(x_1, 1, =), (x_2, 1, =)\}$ *there is one AXp* $\{2\}$*, but there is no CXp. In contrast, for the partial assignment* $\{(x_2, 1, =), (x_3, 1, =), (x_4, 1, =)\}$*, there is one AXp* $\{2\}$ *and one CXp* $\{2\}$*.*

4. Related Work

Rashomon sets. For a given classification problem, the Rashomon set [1, 2] captures the set of DTs for a given classification problem which are within some distance from some accepted definition of optimal DT [2]. Recent work [17] has shown that a significant percentage of the DTs in computed Rashomon sets can represent predictive equivalent DTs. Multiple copies of predictive equivalent DTs are uninteresting, since all these predictive equivalent DTs represent the same classification function.

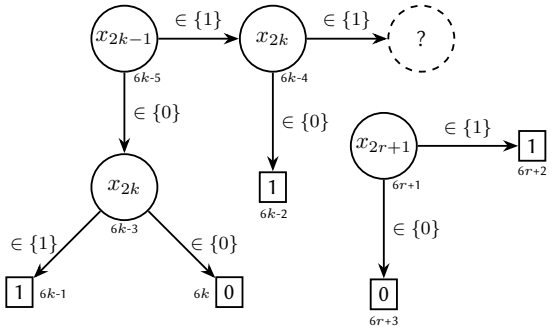
Predictive equivalence. With the purpose of identifying and removing predictive equivalent DTs, McTavish et al. [17] proposed to apply the well-known Quine-McCluskey method [21, 22, 23] for the simplification of boolean formulas. Nevertheless, it is well-known that QM solves an optimization problem that is hard for the second level of the polynomial-hierarchy [24, 25]. Moreover, a polynomial time algorithm for predictive equivalence of DTs was proposed in the late 1990s [41] for decision trees.

XAI queries. Besides deciding predictive equivalence of DTs, the work of McTavish et al. [17] also studied two concrete queries. The first query is to provide smallest/minimal explanations for predictions, and the second query is to decide prediction sufficiency in the presence of missing data. In a more recent report, McTavish et al. propose a linear time algorithm for deciding prediction sufficiency [42].

The limitations of the QM method. As shown in later sections, all the problems studied in [17] can be solved in polynomial time. Nevertheless, one might argue that, despite solving an optimization problem hard for the second level of the polynomial hierarchy, the QM method might still be competitive with those polynomial-time solutions. Unfortunately, this is not the case. We briefly analyze examples that demonstrate the inadequacy of using the QM method.

As noted earlier in the paper, the QM method starts by generating all the prime implicants and minterms of a propositional formula. In the case of the work of McTavish et al., the propositional formula is extracted from the DT, and captures the implicants for each prediction. As a result, it suffices to construct DTs for which the extracted formula is guaranteed to generate exponentially many prime implicants and, as a result, exponentially many minterms. Since AXps are prime implicants given a specific instance [33], then it suffices to devise DTs with exponentially many AXps. This result has been proved in earlier work [19]. Thus, it is the case that there exist parameterizable DTs, exhibiting exponentially many AXps, for which the size required by the QM method will be exponentially large.

We study one such example here, which will serve to demonstrate the practical limitations of using the QM method in Section 8.



(a) Main gadget (b) Final gadget

Figure 2:
Gadgets used in the proof of Proposition 1. The main gadget is repeated r times, i.e. , $k = 1, \ldots, r$.

Proposition 1. There exist DTs $\mathcal{T}$ for which the number of AXps for one of the instances is exponential in the size of $\mathcal{T}$.

Proof. (Sketch) We construct a DT with $2r + 1$ features and $6r + 3$ nodes, for some integer r. For this DT, a lower bound on the number of AXps is shown to be 2^r. Concretely, we select an instance, and prove that 2^r is a lower bound of the number of AXps

The DT is constructed with two gadgets, shown in Figure 2. The main gadget is replicated r times. In addition, the final gadget is linked to the last of the r replicas of the main gadget. Clearly, the DT size is polynomial on the number of features. The chosen instance is $((1, \ldots, 1), 1)$. A lower bound on the number of AXps is derived as follows. There are $r + 1$ ways to change the prediction of class 1 to class 0. One way involves changing the value of feature x_{2r+1}. Each of the other ways involves a pair of variables x_{2k-1} and x_{2k}, $k = 1, \ldots, r$. Thus, for ensuring that the prediction cannot be changed, one must choose from 2^r possible ways. Each such way is subset-minimal, and so it represents an AXp. □

5. Tractable Predictive Equivalence

As noted earlier in Section 4, the use of QM is unnecessary for deciding predictive equivalence of DTs, and a polynomial time algorithm has been known since the late 1990s [41]. However, the algorithm of Zantema [41] tackles trees with binary or categorical features, it iteratively transforms the DT, and its analysis is non-trivial. (Also, to the best of our knowledge, there is no publicly available implementation of this algorithm.) This section proposes a simple algorithm for predictive equivalence, with worst-case complexity matching that of Zantema's algorithm, but that removes restrictions on the domains of features, and that avoids the additional overhead incurred by Zantema's algorithm. Furthermore, the proposed algorithm is used to derive an asymptotically more efficient parallel algorithm.

Predictive equivalence for DTs. Algorithms 1 and 2 summarize the main steps of our approach for deciding predictive equivalence. Algorithm 1 traverses $\mathcal{T}_1$. For each terminal node and associated set of assignments (and indirectly a tree path of $\mathcal{T}_1$), Algorithm 1 invokes Algorithm 2 to check consistency between paths with different predictions. DT $\mathcal{T}_2$ is then traversed, and for each terminal node such that the literals in the path are consistent with those obtained from $\mathcal{T}_1$, difference in predictions represents a counterexample to prediction equivalence. The updates to the sets of literals, and the consistency checks are all attained in constant time. As a result, the algorithm exhibits a running time in $\mathcal{O}(|\mathcal{T}_1| \times |\mathcal{T}_2|)$.

Parallelization. The organization of Algorithms 1 and 2 focuses on the run time complexity. However, a different organization could be envisioned, i.e. one that analyzes the paths of $\mathcal{T}_1$ one at a time, and compares each path of $\mathcal{T}_1$ for consistency with $\mathcal{T}_2$. This solution would not be as efficient, but it reveals different approaches to parallelize deciding predictive equivalence. One solution is to run in parallel the checking of each path of $\mathcal{T}_1$ against $\mathcal{T}_2$ (or vice-versa). Another solution is to enumerate all pairs of paths and compare each pair in parallel.

Tractable predictive equivalence: more cases. For several other families of classifiers, predictive equivalence can be decided in polynomial time. We briefly review those results. First, since equivalence checking is in polynomial time for several propositional languages [43, 44], then for these languages,

Algorithm 1: Predictive equivalence: traversal of $\mathcal{T}_1$.

Input: $\mathcal{T}_1$: DT 1; $\mathcal{T}_2$: DT 2
Output: **true** if DTs are predictive equivalent; else **false**

function PredictiveEquivalent($\mathcal{T}_1$, $\mathcal{T}_2$)
  $\mathcal{L}[s] \leftarrow \emptyset$ **for** s **in** Nodes($\mathcal{T}_1$)
  return PredEquivalentRecur($\mathcal{T}_1$, Root($\mathcal{T}_1$), $\mathcal{L}$, $\mathcal{T}_2$)
function PredEquivalentRecur($\mathcal{T}_1$, s, $\mathcal{L}$, $\mathcal{T}_2$)
  if Terminal(s; $\mathcal{T}_1$) **then**
    return NoConsist($\mathcal{T}_2$, Root($\mathcal{T}_2$), $\mathcal{L}$, Class(s; $\mathcal{T}_1$))
  for $t \in$ Children(s; $\mathcal{T}_1$) **do**
    lit $\leftarrow$ Literal(s, t; $\mathcal{T}_1$)
    $\mathcal{L}[s] \leftarrow$ LitReduce($\mathcal{L}[s] \cup \{\text{lit}\}$)
    if not PredEquivalentRecur($\mathcal{T}_1, t, \mathcal{L}, \mathcal{T}_2$) **then**
      return false
    $\mathcal{L}[s] \leftarrow$ LitReduce($\mathcal{L}[s] \setminus \{\text{lit}\}$)
  return true

Algorithm 2: Predictive equivalence: traversal of $\mathcal{T}_2$.

Input: $\mathcal{T}$: DT; s: tree node; $\mathcal{L}$: literals; c: class
Output: **true** is no consistency; **false** otherwise

function NoConsist($\mathcal{T}, s, \mathcal{L}, c$)
  if Terminal(s; $\mathcal{T}$) **then**
    return ITE(Class(s; $\mathcal{T}$) $= c$, **true**, **false**)
  for $t \in$ Children(s; $\mathcal{T}$) **do**
    lit $\leftarrow$ Literal(s, t; $\mathcal{T}$)
    if Consistent($\mathcal{L}[s] \cup \{\text{lit}\}$) **then**
      $\mathcal{L}[s] \leftarrow$ LitReduce($\mathcal{L}[s] \cup \{\text{lit}\}$)
      if not NoConsist($\mathcal{T}, r, \mathcal{L}, c$) **then**
        return false
      $\mathcal{L}[s] \leftarrow$ LitReduce($\mathcal{L}[s] \setminus \{\text{lit}\}$)
  return true

it is the case that predictive equivalence can be decided in polynomial-time. SDDs and OBDDs are two well-known examples. Second, for multi-variate decision trees, there are specific languages of node constraints that enable polynomial-time computation of explanations [45]. This is the case when each node queries a Horn clause, among others. For these cases, Algorithms 1 and 2 can also be used, ensuring that predictive equivalence can be decided in polynomial time.

6. Missing Data & Tractable XAI Queries

In Section 3, we investigated how AXps/CXps can be redefined to account for missing data. A downside of this redefinition is that minimal hitting set duality between AXps and CXps no long holds. In turn, this might affect the complexity of XAI queries studied in the past [32]. This section studies several other XAI queries in the presence of missing data, and proves that, in all cases considered, tractability is not affected. Nevertheless, new algorithms have to be devised.

XAI queries. Given the relationship between prediction(s) in the presence of missing data and WAXps, we can envision several more XAI queries that can be answered in the presence of missing data:

Algorithm 3: Deciding prediction sufficiency

Input: $\mathcal{T}$: DT; $\mathcal{A}$: partial assignment
Output: Value in $\mathbb{K} \cup \{\mathfrak{u}\}$

```
1: function PredictionSufficient(T, A)
2:     rv, cind ← IsSufficient(T, Root(T), A, ∅)
3:     if rv = u or cind then
4:         return u
5:     return rv
```

1. Deciding WAXp/WCXp: (i) decide whether a partial assignment and a set of features with missing data is a WAXp; and (ii) decide whether changes to the partial assignment can be made so as to change the prediction.
2. Finding one AXp/CXp: (i) find one CXp; and (ii) find on AXp.
3. Deciding feature necessity: (i) decide whether a given feature must be included in all AXps; and (ii) decide whether a given feature must be included in all CXps.
4. Deciding feature relevancy: (i) decide whether a given feature is included in some CXp; and (ii) decide whether a given feature is included in some AXp.
5. Enumeration of WCXps/WAXps: (i) enumerate the set of CXps; and (ii) enumerate the set of AXps.

The queries above are premised on a partial assignment $\mathcal{A}$ being sufficient for predicting a value other than $\mathfrak{u}$. As a result, a related query is to decide whether $\mathcal{A}$ is indeed sufficient for predicting a value other than $\mathfrak{u}$.

Tractability results for DTs. We now show that most of the queries list above are tractable. The exceptions are (a) the enumeration of AXps, since there can exist exponentially many AXps [19]; and (b) finding a smallest-size AXp [19]. Section 7 proves that even enumeration of AXps with polynomial delay is unlikely.

Proposition 2. For DTs, in the presence of missing data, the following queries are in P:
(q1) Deciding whether a partial assignment and a set of features with missing data is a WAXp;
(q2) Deciding whether changes to the partial assignment can be made so as to change the prediction;
(q3) Finding one (smallest) CXp;
(q4) Finding one AXp;
(q5) Deciding whether a given feature is included in some CXp, i.e. CXp relevancy;
(q6) Decide whether a given feature must be included in all CXps, i.e. CXp necessity;
(q7) Deciding whether a given feature is included in some AXp, i.e. AXp relevancy;
(q8) Deciding whether a given feature must be included in all AXps, i.e. AXp necessity; and
(q9) Enumerating the set of CXps.

Proof. (q1) corresponds to prediction sufficiency, which can be decided in polynomial-time for a set of fixed features [20, 42]. The algorithm of [42] runs in linear time in the size of the DT but applies to binary decision trees. Algorithms 3 and 4 details a general-purpose algorithm allowing features with categorical or ordinal domains, and the universe of literals introduced in Section 2.
We now prove (q9), since it will serve to prove the other cases. Without missing data, the enumeration of CXps is known to be in polynomial-time [18, Proposition 6]. In the presence of missing data, we proceed as follows. Compute the set of all CXps for DT, treating the features with unknown values, i.e. set U, as fixed features with value $\mathfrak{u}$, thereby allowing the features in U to be included in CXps. Next, remove any CXp with a feature in U. If $\mathbb{C}(\mathcal{A}_S, U)$ is the set of all CXps allowing the features in U to take value $\mathfrak{u}$, then the set of AXps becomes:

$$\mathbb{C}(\mathcal{A}_S) = \{Y \in \mathbb{C}(\mathcal{A}_S, U) \,|\, Y \cap U = \emptyset\}\}$$

Thus, it is the case that (q9) is in P.
To answer (q2), we first answer (q9), and then check if the set of features is one of the CXps.

Algorithm 4: Recursive procedure to find prediction given partial assignment

Input: $\mathcal{T}$: DT; s: tree node; $\mathcal{A}$: partial assign; $\mathcal{L}$: lits
Output: Value in $\mathbb{K} \cup \{\mathfrak{u}\}$, Conflict indication

```
1:  function IsSufficient(T, s, A, L)
2:      if Terminal(s; T) then
3:          return Class(s; T), false
4:      rv ← u
5:      for t ∈ Children(s; T) do
6:          lit ← Literal(s, t; T)
7:          if Consistent(A, L ∪ {lit}) then
8:              nrv, cind ← IsSufficient(T, r, A, L ∪ {lit})
9:              if rv = u then
10:                 rv ← nrv
11:             if nrv ≠ u and nrv ≠ rv or cind then
12:                 return u, true
13:     return rv, false
```

To answer (q3), we first answer (q9), and then return any of the computed CXps. For a smallest CXp, just sort the CXps by size, and return the one of the smallest size.
To answer (q4), we first answer (q9), and then find a (subset-)minimal hitting set in polynomial-time.
To answer (q5), we first answer (q9), and then check whether the feature occurs in any CXp.
To answer (q6), we first answer (q9), and then check whether the feature occurs in all CXps.
To answer (q7), we first enumerate all CXps as in the case of (q9), thus obtaining $\mathbb{C}(\mathcal{A}_S, U)$. However, instead of dropping CXps that contain a feature in U, we remove from the CXps any feature that occurs in U. In case all features are removed, then we remove the CXp:

$$\mathbb{H}(\mathcal{A}_S) = \{Y \setminus U \,|\, Y \in \mathbb{C}(\mathcal{A}_S, U)\} \setminus \{\emptyset\}$$

The resulting sets can be used for enumerating the AXps (i.e. computing all the minimal hitting sets) that do not include features in U. To decide relevancy of a feature, we just need to check whether the feature occurs in any of those sets.
To answer (q8), we first construct the same sets as for answering (q7). Any feature occurring in the intersection of all the sets will be included in all the minimal hitting sets. □

For deciding whether a partial assignment if a WAXp or for computing one AXp, we can either use the algorithms proposed in earlier work [20], or exploit instead Algorithms 3 and 4. Since deciding the WAXp condition corresponds to deciding prediction sufficient, in this paper we opt for interfacing Algorithms 3 and 4. Moreover, the algorithms described in the proof of Proposition 2 are illustrated with the DT of Figure 1.

Example 6. *We consider again the consistent partial assignment $\mathcal{A}_S = \{(x_2, 1, =), (x_3, 1, =)\}$, which is sufficient for prediction 1. We assign to features 1 and 4 in U the value $\mathfrak{u}$. Then we apply the algorithm for enumerating (W)CXps proposed in [18, Proposition 6], thereby obtaining:*

$$\mathbb{WC}(\mathcal{A}_S, U) = \{\{1, 2\}, \{1, 2, 4\}\}$$
$$\mathbb{C}(\mathcal{A}_S, U) = \{\{1, 2\}\}$$

(The CXps are obtained by traversing the paths with prediction other than 1, and collecting the literals inconsistent with the given assignment.)
As a result, the set of CXps obtained by removing CXps containing features 1 or 4 is: $\mathbb{C}(\mathcal{A}_S) = \emptyset$, i.e. we are unable to change the prediction independently of the values of features with unknown values.
Next, we proceed to compute the set of AXps. First, we obtain the set $\mathbb{H}(\mathcal{A}_S) = \{\{2\}\}$. Thus, the only AXp not containing features with unknown values is: $\{2\}$.

Algorithm 5: Construct DT from hypergraph

Input: L: list of nodes; E: hyperedges
Output: DT $\mathcal{T}$

function ConstructDT(L, E)
  if $\emptyset \in E$ **then**
    nn $\leftarrow$ TreeTermNode(0)
  else if $E = \emptyset$ **then**
    nn $\leftarrow$ TreeTermNode(1)
  else
    fn $\leftarrow$ L.pop(0)
    nn $\leftarrow$ TreeNode(fn)
    $E_l \leftarrow \{S \,|\, S \in E \land \text{fn} \notin S\}$
    nn.left $\leftarrow$ ConstructDT(L, E_l)
    $R \leftarrow L$.copy()
    $E_r \leftarrow \{S \setminus \{\text{fn}\} \,|\, S \in E \land \text{fn} \in S\}$
    nn.right $\leftarrow$ ConstructDT(R, E_r)
  return nn

7. Complexity of AXp Enumeration in DTs

This section studies the complexity of AXp enumeration for ML models for which the enumeration of CXps is in polynomial time. For example, this is the case of DTs [18, 20].

Hypergraph transversals. An hypergraph is a tuple $G = (V, E)$, with $E \subseteq 2^V$. The hypergraph transversal problem (HypTr) for G consists of enumerating all the minimal hitting sets (i.e. the transversals) of E [46]. It is well-known that the hypergraph transversal problem can be solved with a quasi polynomial time delay algorithm [47, 46]. To the best of our knowledge, polynomial delay algorithms have not been proposed and, given past work, should be unexpected.

AXp enumeration as hypergraph transversal. If the set of CXps is unknown (or it worst-case exponential), and so it is computed on-demand, then enumeration of AXps/CXps is usually achieved by adapting the MARCO algorithm [48], e.g. [32].

In this section, the number of CXps is assumed to be polynomial in the size of the ML model representation. Past work proposed the use of algorithms for finding the hypergraph transversal (also referred to as minimal hitting set dualization). In theory, this problem is well-known to be in quasi polynomial time [46, 49, 50, 51].

Hypergraph transversal as AXp enumeration. The previous reduction of AXp enumeration to the hypergraph transversal problem provides an upper bound on the complexity of AXp enumeration. Nevertheless, an open question is whether AXp enumeration might be implemented more efficiently than solving the hypergraph transversal problem. We now show that this is not the case, by reducing the hypergraph transversal problem to that of enumerating the AXps of a DT, thus proving a lower bound on the complexity of AXp enumeration for DTs.

Proposition 3. The problem of hypergraph transversal reduces in polynomial-time to the problem of AXp enumeration for DTs, $\texttt{HypTr} \leq_{\text{p}} \texttt{AllAXp}_{\text{DT}}$.

Proof. Let $G = (V, E)$, with $V = \{1, \ldots, n\}$ and $E \subseteq 2^V$, represent an hypergraph. We reduce HypTr for G to the problem of computing the AXps of a binary DT $\mathcal{T}$, as follows. The set of binary features is $\mathcal{F} = V$. The instance to explain is $((1, \ldots, 1), 1)$. For each set $S \in E$, we create a path in $\mathcal{T}$ ending in a terminal node predicting 0. All other paths in $\mathcal{T}$ end in terminal nodes predicting 1. To construct the DT $\mathcal{T}$, we pick any order of the features. We will prove that any order will do. Thus, WLOG, we pick the order $\langle 1, \ldots, m \rangle$. The DT is then constructed as shown in Algorithm 5. Given a hypergraph $G = (V, E)$, ConstructDT is invoked with the arguments list(V) and E. For each element i of V, the branch with value 0 means that i is *picked* along this path. Thus, the elements of E without i are

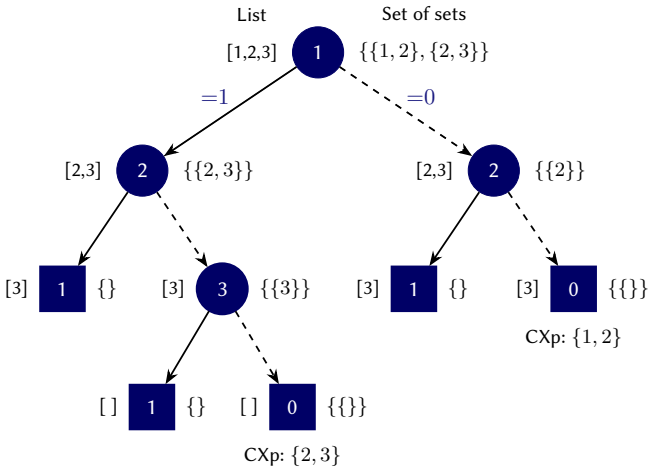


Figure 3: Construction of DT for $G = (\{1,2,3\}, \{\{1,2\}, \{2,3\}\})$. Solid lines denote an assignment of 1 to the feature, whereas dashed lines denote an assignment of 0 to the feature. For the instance $((1,1,1),1)$, the set of CXps is $\{\{1,2\}, \{2,3\}\}$, i.e. the set of sets for which the minimal hitting sets are to be computed.

discarded, and i is dropped form the elements of E containing i. The branch with value 1 means that the sets containing i are no longer considered, i.e. i is not picked along this path. Each leaf tagged with a set containing the empty set represents one of the sets to hit, and there is exactly one such path for each set in E. Each leaf tagged the empty set represents a case we do not care about. □

Figure 3 illustrates the intuition for the reduction in the proof above with a simple hypergraph $G = (\{1,2,3\}, \{\{1,2\}, \{2,3\}\})$. The CXps are $\{1,2\}$ and $\{2,3\}$, and so the AXps represent the transversal of the starting hypergraph G.

Since we prove lower and upper bounds using HypTr, then it is the case that,

Corollary 1. $\texttt{AllAXp}_{\text{DT}}$ is computationally as hard as HypTr.

8. Experiments

The results in [17] offers comprehensive evidence regarding the importance of eliminating predictive equivalent DTs from the Rashomon set. Moreover, most of the algorithms proposed in the previous section exhibit negligible running times, even for DTs of fairly large size. (Our results concur with similar results were obtained in earlier work when computing instance-specific explanations [20].) As a result, the focus of our experiments is to demonstrate the practical effect of exercising the worst-case running time of the QM method used by McTavish et al. Since the proof of Proposition 1 only considers a single instance, the experiments target the complete enumeration of AXps for the instance $((1,\ldots,1),1)$, for different values of r.

Experimental setup. The algorithm described in [17] uses SymPy's implementation of QM [52].[5] We also implemented a prototype in Python, that mimics a standard implementation of the prime implicant generation step of the QM method. Furthermore, all the algorithms described in the paper were implemented in Python, with the exception of parallel predictive equivalent checking. Furthermore, the results were validated, and the performance was assessed. All experiments were run on a Macbook with an M3 Pro processor. For the DT used in the proof of Proposition 1, different values of r were considered.

Results. Table 1 summarizes the results for values of r ranging from 3 to 9, i.e. DTs with up to $6 \times r + 3 = 57$ nodes (for $r = 9$). As can be observed, for class 1 (resp. class 0), the number of AXps grows exponentially (resp. linearly) with the value of r. SymPy's QM algorithm shows large running times, and it was stopped for $r = 7$, i.e. DT with 45 nodes. For the independent generation of the sets of prime implicants (for the two classes), the running times also grow exponentially (as expected). For $r = 9$, the running times for computing the AXps (see column AXp time) are already unwieldy. The inefficiency observed is attributed to the fact that Python was used and, more importantly and as noted earlier in the paper, the computational overhead of generating the set of AXps. This overhead

[5] https://www.sympy.org/.

r	SymPy	#AXps = 0	#AXps = 1	AXp time
3	0.13	4	22	0.01
4	0.57	5	46	0.07
5	39.60	6	94	0.84
6	2789.45	7	190	11.28
7	>150000.00	8	382	161.25
8	– –	9	766	2264.62
9	– –	10	1534	64458.55

Table 1
Results for different values of r, $r \leq 9$. SymPy's QM was stopped for $r = 7$ after 150000s. Running times in seconds. The number of DT nodes is: $6 \times r + 3$ and the number of features is $2 \times r + 1$.

r	# nodes	# feat.	LB #AXps	one AXp	isWAXp?	PE?
200	1203	401	2^{200}	0.054s	0.0002s	0.029s
500	3003	1001	2^{500}	0.377s	0.0004s	0.169s
1000	6003	2001	2^{1000}	1.480s	0.0009s	0.727s

Table 2
Running times for DTs much larger than those in Table 1, with $r \in \{200, 500, 1000\}$.

is significant, and worsens for trees with more AXps. For the algorithms proposed in this paper as alternatives to the work of McTavish et al, we consider the cases $r = 200, 500, 1000$, as shown in Table 2. The results illustrate the scalability of the DT predictive equivalence algorithm described in Section 5, for DTs for which a lower bounds on the number of prime implicants is well-beyond what practical implementations of the QM method would achieve.

Finally, we underscore that the experimental results consider a special DT construction aiming at highlighting the performance limitations of using an implementation of the QM method. In practice, DTs of fairly large size can occur in some high-risk [53] application domains, where high accuracy is paramount. For example, [54] reports a DT with 161 nodes and 20 real-valued features. In those cases, the use of the QM method will also not scale, even if high-performance solutions are exploited [55] (and these might not ensure deterministic execution). (Furthermore, we note that SymPy is known to be inefficient for formulas with more than 8 boolean variables.[6])

9. Conclusions

Recent work [17] studied the problem of predictive equivalence in the context of eliminating redundant decision trees (DTs) from the Rashomon set. The proposed solution is based on the Quine-McCluskey (QM) method [21, 22, 23]. However, it is well-known that the QM method solves an optimization problem that is hard for the second level of the polynomial hierarchy. In addition, the same work studied prediction sufficiency in the presence of missing data and explanation redundancy, and proposed their solution using the representation obtained with the QM method. This paper proves that the use of the QM method is unnecessary, since all problems studied in [17] can be solved in polynomial time in the case of DTs. Also, regarding predictive equivalence, the paper observes that it can be decided in polynomial time for several other families of ML models. Furthermore, the paper proves that several queries in the presence of missing data relate with similar queries in XAI, and so can also be solved with polynomial-time algorithms in the case of DTs. Finally, the paper proves a lower bound on the enumeration of abductive explanations for DTs, which in turn suggest that enumeration with polynomial delay, while not necessarily impossible, appears to be unlikely.

[6]This is discussed for example in https://docs.sympy.org/latest/modules/logic.html.

10. Declaration on Generative AI

No generative AI tools were used in preparing this paper.

References

[1] L. Breiman, Statistical modeling: The two cultures, Statistical science 16 (2001) 199–231.

[2] R. Xin, C. Zhong, Z. Chen, T. Takagi, M. I. Seltzer, C. Rudin, Exploring the whole Rashomon set of sparse decision trees, in: NeurIPS, 2022.

[3] A. Fisher, C. Rudin, F. Dominici, All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously, J. Mach. Learn. Res. 20 (2019) 177:1–177:81.

[4] L. Semenova, C. Rudin, A study in Rashomon curves and volumes: A new perspective on generalization and model simplicity in machine learning, CoRR abs/1908.01755 (2019). arXiv:1908.01755.

[5] C. Rudin, C. Chen, Z. Chen, H. Huang, L. Semenova, C. Zhong, Interpretable machine learning: Fundamental principles and 10 grand challenges, Statistic Surveys 16 (2022) 1–85.

[6] L. Semenova, C. Rudin, R. Parr, On the existence of simpler machine learning models, in: FAccT, 2022, pp. 1827–1858.

[7] J. Donnelly, S. Katta, C. Rudin, E. P. Browne, The Rashomon importance distribution: Getting RID of unstable, single model-based variable importance, in: NeurIPS, 2023.

[8] L. M. Paes, R. Cruz, F. P. Calmon, M. Díaz, On the inevitability of the Rashomon effect, in: ISIT, 2023, pp. 549–554.

[9] P. Biecek, H. Baniecki, M. Krzyzinski, Performance is not enough: a story of the Rashomon's quartet, CoRR abs/2302.13356 (2023). arXiv:2302.13356.

[10] K. Kobylinska, M. Krzyzinski, R. Machowicz, M. Adamek, P. Biecek, Exploration of the Rashomon set assists trustworthy explanations for medical data, IEEE J. Biomed. Health Informatics 28 (2024) 6454–6465.

[11] H. Hsu, G. Li, S. Hu, C. Chen, Dropout-based Rashomon set exploration for efficient predictive multiplicity estimation, in: ICLR, 2024.

[12] M. Ciaperoni, H. Xiao, A. Gionis, Efficient exploration of the Rashomon set of rule-set models, in: KDD, 2024, pp. 478–489.

[13] S. Li, R. Wang, Q. Deng, A. S. Barnard, Exploring the cloud of feature interaction scores in a Rashomon set, in: ICLR, 2024.

[14] S. Li, A. S. Barnard, Q. Deng, Practical attribution guidance for Rashomon sets, CoRR abs/2407.18482 (2024). arXiv:2407.18482.

[15] J. Donnelly, Z. Guo, A. J. Barnett, H. McTavish, C. Chen, C. Rudin, Rashomon sets for prototypical-part networks: Editing interpretable models in real-time, in: CVPR, 2025, pp. 4528–4538.

[16] S. Nguyen, K. Hoffman, T. H. McCormick, Unique Rashomon sets for robust active learning, CoRR abs/2503.06770 (2025). arXiv:2503.06770.

[17] H. McTavish, Z. Boner, J. Donnelly, M. I. Seltzer, C. Rudin, Leveraging predictive equivalence in decision trees, in: ICML, 2025.

[18] X. Huang, Y. Izza, A. Ignatiev, J. Marques-Silva, On efficiently explaining graph-based classifiers, in: KR, 2021, pp. 356–367.

[19] G. Audemard, S. Bellart, L. Bounia, F. Koriche, J. Lagniez, P. Marquis, On the explanatory power of boolean decision trees, Data Knowl. Eng. 142 (2022) 102088.

[20] Y. Izza, A. Ignatiev, J. Marques-Silva, On tackling explanation redundancy in decision trees, J. Artif. Intell. Res. 75 (2022) 261–321.

[21] W. V. Quine, The problem of simplifying truth functions, American mathematical monthly (1952) 521–531.

[22] W. V. Quine, A way to simplify truth functions, American mathematical monthly (1955) 627–631.

[23] E. J. McCluskey, Minimization of Boolean functions, Bell system technical Journal 35 (1956) 1417–1444.
[24] C. Umans, The minimum equivalent DNF problem and shortest implicants, in: FOCS, 1998, pp. 556–563.
[25] C. Umans, T. Villa, A. L. Sangiovanni-Vincentelli, Complexity of two-level logic minimization, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 25 (2006) 1230–1246.
[26] A. Ignatiev, N. Narodytska, J. Marques-Silva, Abduction-based explanations for machine learning models, in: AAAI, 2019, pp. 1511–1519.
[27] G. Audemard, S. Bellart, J. Lagniez, P. Marquis, Computing abductive explanations for boosted regression trees, in: IJCAI, 2023, pp. 3432–3441.
[28] J. Marques-Silva, Logic-based explainability: Past, present and future, in: ISoLA, 2024, pp. 181–204.
[29] I. Dillig, T. Dillig, K. L. McMillan, A. Aiken, Minimum satisfying assignments for SMT, in: CAV, 2012, pp. 394–409.
[30] L. Amgoud, Explaining black-box classifiers: Properties and functions, Int. J. Approx. Reason. 155 (2023) 40–65.
[31] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32.
[32] J. Marques-Silva, Logic-based explainability in machine learning, in: Reasoning Web, 2022, pp. 24–104.
[33] A. Shih, A. Choi, A. Darwiche, A symbolic approach to explaining bayesian network classifiers, in: IJCAI, 2018, pp. 5103–5111.
[34] A. Darwiche, A. Hirth, On the (complete) reasons behind decisions, J. Log. Lang. Inf. 32 (2023) 63–88.
[35] A. Darwiche, Logic for explainable AI, in: LICS, 2023, pp. 1–11.
[36] A. Ignatiev, N. Narodytska, N. Asher, J. Marques-Silva, From contrastive to abductive explanations and back again, in: AIxIA, 2020, pp. 335–355.
[37] T. Miller, Explanation in artificial intelligence: Insights from the social sciences, Artif. Intell. 267 (2019) 1–38.
[38] R. Reiter, A theory of diagnosis from first principles, Artif. Intell. 32 (1987) 57–95.
[39] J. Slaney, Set-theoretic duality: A fundamental feature of combinatorial optimisation, in: ECAI, 2014, pp. 843–848.
[40] J. Marques-Silva, M. Janota, C. Mencía, Minimal sets on propositional formulae. problems and reductions, Artif. Intell. 252 (2017) 22–50. URL: https://doi.org/10.1016/j.artint.2017.07.005.
[41] H. Zantema, Decision trees: Equivalence and propositional operations, Technical Report 14, Utrecht University, 1998.
[42] H. McTavish, Z. Boner, J. Donnelly, M. Seltzer, C. Rudin, Leveraging predictive equivalence in decision trees, CoRR abs/2506.14143 (2025). arXiv:2506.14143.
[43] A. Darwiche, P. Marquis, A knowledge compilation map, J. Artif. Intell. Res. 17 (2002) 229–264.
[44] G. Van den Broeck, A. Darwiche, On the role of canonicity in knowledge compilation, in: AAAI, 2015, pp. 1641–1648.
[45] C. Carbonnel, M. C. Cooper, J. Marques-Silva, Tractable explaining of multivariate decision trees, in: KR, 2023, pp. 127–135.
[46] L. Khachiyan, E. Boros, K. M. Elbassioni, V. Gurvich, An efficient implementation of a quasi-polynomial algorithm for generating hypergraph transversals and its application in joint generation, Discret. Appl. Math. 154 (2006) 2350–2372.
[47] M. L. Fredman, L. Khachiyan, On the complexity of dualization of monotone disjunctive normal forms, J. Algorithms 21 (1996) 618–628.
[48] M. H. Liffiton, A. Previti, A. Malik, J. Marques-Silva, Fast, flexible MUS enumeration, Constraints 21 (2016) 223–250.
[49] T. Eiter, G. Gottlob, K. Makino, New results on monotone dualization and generating hypergraph transversals, SIAM J. Comput. 32 (2003) 514–537.
[50] D. J. Kavvadias, E. C. Stavropoulos, Monotone boolean dualization is in co-np[$\log^2$n], Inf. Process. Lett. 85 (2003) 1–6.

[51] G. Gottlob, E. Malizia, Achieving new upper bounds for the hypergraph duality problem through logic, SIAM J. Comput. 47 (2018) 456–492.
[52] A. Meurer, et al., SymPy: symbolic computing in Python, PeerJ Comput. Sci. 3 (2017) e103.
[53] C. Rudin, Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, Nature Machine Intelligence 1 (2019) 206–215.
[54] M. M. Ghiasi, S. Zendehboudi, A. A. Mohsenipour, Decision tree-based diagnosis of coronary artery disease: CART model, Comput. Methods Programs Biomed. 192 (2020) 105400.
[55] O. Coudert, Two-level logic minimization: an overview, Integr. 17 (1994) 97–140.