On Explaining Decision Trees

Yacine Izza, Alexey Ignatiev, Joao Marques-Silva

On Explaining Decision Trees

Yacine Izza[1], Alexey Ignatiev[2], and Joao Marques-Silva[3]

[1] ANITI, Univ. Toulouse, France
[2] Monash Univ., Australia
[3] ANITI, IRIT, CNRS, France

Abstract. Decision trees (DTs) epitomize what have become to be known as interpretable machine learning (ML) models. This is informally motivated by paths in DTs being often much smaller than the total number of features. This paper shows that in some settings DTs can hardly be deemed interpretable, with paths in a DT being arbitrarily larger than a PI-explanation, i.e. a subset-minimal set of feature values that entails the prediction. As a result, the paper proposes a novel model for computing PI-explanations of DTs, which enables computing one PI-explanation in polynomial time. Moreover, it is shown that enumeration of PI-explanations can be reduced to the enumeration of minimal hitting sets. Experimental results were obtained on a wide range of publicly available datasets with well-known DT-learning tools, and confirm that in most cases DTs have paths that are proper supersets of PI-explanations.

1 Introduction

Decision trees (DTs) are well known machine learning (ML) models, studied since at least the 1970s [22,27,14,50,52]. DTs embody what is widely regarded as an interpretable ML model [20,51,41,59,40,38,23,76,61][1]. Motivated by this perception, there has been extensive work on learning DTs (and related logic ML models) with properties deemed important for interpretability, e.g. number of nodes, maximum/average depth, etc. [2,67,25,31,1,7,56,24,26,69,66,57,43,17,5,72,34,4,77,68,10,33,71,70,35,45,11,44][2]. Moreover, there has been work on *distilling* or approximating complex ML models with (soft) decision trees [21,9,8,73,75,55,74]. Nevertheless, recent work highlights that interpretability should correlate with how shallow DTs are [36,41].

In contrast with earlier work, this paper investigates the limits of interpretability of DTs. Concretely, the paper proposes Boolean functions for which a minimal DT contains paths with a number of literals that is arbitrarily larger (growing with the number of features) than a PI-explanation[3] of constant size. Experimental results demonstrate

[1] Interpretability is generally accepted to be a subjective concept, without a rigorous definition [36]. In this paper we measure interpretability in terms of the overall succinctness of the information provided by a model to explain a given prediction. The association of DTs with interpretability can be traced at least to Breiman [13], who summarizes the interpretability of DTs as follows: "*On interpretability, trees rate an A+*".

[2] The association of DTs with interpretability is also illustrated by Interpretable AI (https://www.interpretable.ai/), which offers interpretability solutions based on optimal decision trees.

[3] A PI-explanation is a subset-minimal set of feature-value pairs that entails the prediction [60].

that for widely used tools for constructing DT classifiers, the resulting DTs often contain paths that are proper supersets of PI-explanations (which we refer as explanation-redundant paths, in possibly irredundant DTs). Perhaps more importantly, for a significant number of datasets, and for the obtained DTs, most of their paths are explanation-redundant. The results also indicate that for some DTs, as much as 98% of feature space will be explained by some path that is explanation-redundant. Motivated by these negative results, we propose a hitting set formulation for computing PI-explanations of DTs, distinguishing explanations restricted to literals in a tree path, and explanations unrestricted to literals in a tree path. In addition, we propose a polynomial time algorithm for computing a single PI-explanation for any given instance. Finally, the paper reduces enumeration of PI-explanations of DTs to the problem of enumerating minimal hitting sets (MHSes), and proposes a solution based on iterative calls to an NP oracle (e.g. SAT or a 0-1 ILP) oracle.

The paper is structured as follows. Section 2 introduces the notation and definitions used in the remainder of the paper. Section 3 studies functions that elicit poor DT interpretability. Section 4 proposes a polynomial-time algorithm for computing a single PI-explanation for a DT. Section 5 proposes a solution for enumerating PI-explanations of DTs, by reducing the problem to the computation of MHSes. Section 6 studies the DTs obtained with two state-of-the-art tools, on publicly available datasets, and shows that paths in learned decision trees are often proper supersets of PI-explanations. Moreover, the experimental results confirm that run times for extracting PI-explanations are negligible. Finally, Section 7 concludes the paper.

2 Preliminaries

Classification problems. We consider a classification problem defined on a set of features $\mathcal{F} = \{1, \ldots, n\}$, where each feature i takes values from a (categorical) domain D_i [4], and n denotes the number of features. Feature space is defined by $\mathbb{F} = D_1 \times D_2 \times \ldots \times D_n$, each defining the range of (categorical) values of each feature x_i. To refer to an arbitrary point in feature space we use the notation $\mathbf{x} = (x_1, \ldots, x_n)$, whereas to refer to a concrete point in feature space we use the notation $\mathbf{v} = (v_1, \ldots, v_n)$, with $v_i \in D_i$, $i = 1, \ldots, n$. We consider a binary classification problem, with two classes $\mathcal{K} = \{\ominus, \oplus\}$ [5]. (For simplicity, we will often use 0 for $\ominus$ and 1 for $\oplus$.) An *instance* (or example) denotes a pair $(\mathbf{v}, \pi)$, where $\mathbf{v} \in \mathbb{F}$ and $\pi \in \mathcal{K}$. A machine learning model computes a function μ that maps the feature space into the set of classes: $\mu : \mathbb{F} \to \mathcal{K}$. To train an ML model (in our case we are interested in DTs), we start from a set of examples $\mathcal{E} = \{e_1, \ldots, e_m\}$, where each $e_j = (\mathbf{v}_j, \pi_j)$, such that $\mathbf{v}_j \in \mathbb{F}$ and $\pi_j \in \mathcal{K}$, and m is the number of examples.

Decision trees. 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

[4] For simplicity, most of the examples in the paper consider $D_i \triangleq \{0, 1\}$ (i.e. binary features).

[5] The results in the paper are readily applicable to multiple classes; the case $|\mathcal{K}| = 2$ is considered solely for simplicity.

(as opposed to multivariate decision trees [16]); hence, a non-terminal node is associated with a single feature x_i, and each outgoing edge is associated with one (or more) values from D_i. Each terminal node is associated with a value of $\mathcal{K}$. An example of a decision tree is shown in Figure 2. The number of nodes in a DT is r. When $|\mathcal{K}| = 2$, a tree is characterized by two sets of paths, where each path starts at the root and ends at a terminal node. The set $\mathcal{P} = \{P_1, \ldots, P_{k_1}\}$ denotes the paths ending in a $\oplus$ prediction. The set $\mathcal{Q} = \{Q_1, \ldots, Q_{k_2}\}$ denotes the paths ending in a $\ominus$ prediction. We will also use $\mathcal{R} = \mathcal{P} \cup \mathcal{Q}$. A literal is of the form $x_i \bowtie v_i$, where $\bowtie \in \{=, \neq\}$ [6]. x_i is a variable that denotes the value taken by feature i, whereas $v_i \in D_i$ is a constant. To model the operation of some DT learning tools [64], we allow generalized literals of the form $x_i \in S_i$, with $S_i \subsetneq D_i$, such that the literal is consistent if the feature is assigned a value in S_i. Given this generalization, DTs correspond to multi-edge decision trees [6]. Moreover, two literals are inconsistent if they cause the feature to take values that are inconsistent. For example, the literals $(x_1 = 0)$ and $(x_1 = 1)$ are inconsistent.

Each path in $\mathcal{T}$ is associated with a consistent conjunction of literals, denoting the values assigned to the features so as to reach the terminal node in the path. We will represent the set of literals of some tree path by $\mathcal{L}(R_k)$, where R_k is either a path in $\mathcal{P}$ or $\mathcal{Q}$. Each path in the tree *entails* the prediction represented by path's terminal node. Let π denote the prediction associated with path R_k. Then,

$$\forall(\mathbf{x} \in \mathbb{F}).\ \left[\bigwedge\nolimits_{(x_i \bowtie v_i) \in \mathcal{L}(R_k)} (x_i \bowtie v_i)\right] \rightarrow (\mu(\mathbf{x}) = \pi) \tag{1}$$

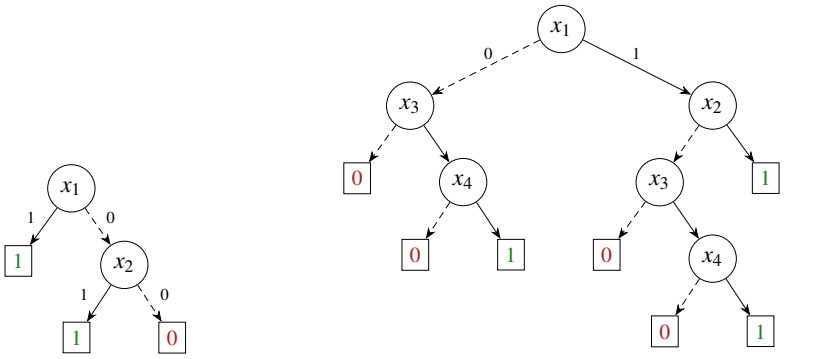
where $\mathbf{x} = (x_1, \ldots, x_i, \ldots, x_n)$ and $\pi \in \{\ominus, \oplus\}$. Any pair of paths in $\mathcal{R}$ must have at least one pair of inconsistent literals.

Interpretability & DTs. Interpretability is generally regarded as a subjective concept, without a rigorous definition [36], albeit different authors have proposed different requirements for interpretability [36,62]. Throughout this paper, we associate interpretability with irreducible sets of feature-value pairs that are sufficient for the prediction[7]. Moreover, as argued in Section 1, it is generally accepted that DTs epitomize interpretability. Nevertheless, there is recent work that relates DT interpretability with DTs being shallow [36,41], but also work that proposes counterfactual explanations for meeting interpretability desiderata for DTs [62].

PI-explanations. The paper uses the definition of PI-explanation [60], based on prime implicants of some decision function. Let us consider some ML model, computing a classification function μ on feature space $\mathbb{F}$, a point $\mathbf{v} \in \mathbb{F}$, with prediction $\pi = \mu(\mathbf{v})$, and let E denote a set of literals consistent with $\mathbf{v}$ (and defined on features variables $\mathbf{x}$). We say that E is a PI-explanation for π given $\mathbf{v}$, if the set of literals E entails the prediction, and any proper subset of literals of E does not entail the prediction. Formally,

[6] The ideas described in the paper generalize to univariate DTs where features are either categorical or real- or integer-valued ordinal, and literals are of the form $x_i \bowtie v_i$, where $\bowtie \in \{<, \leq, =, \neq, \geq, >\}$. The paper's main results can be extended to more general settings, but that these are beyond the scope of the paper.

[7] Clearly, these subsets should be succinct, as it is generally accepted that human decision makers are only able to understand explanations with a reasonably small number of features.



(a) DT for $f(x_1,x_2) = x_1 \lor x_2$ (b) DT for $f(x_1,\ldots,x_n) = \bigvee_{i=1}^{n/2} x_{2i-1} \land x_{2i}$, with $n = 4$

Fig. 1: Example DTs

the following conditions hold:

$$\forall(\mathbf{x} \in \mathbb{F}).\ \left[\bigwedge\nolimits_{l_i \in E}(l_i)\right] \rightarrow (\kappa(\mathbf{x}) = \pi) \tag{2a}$$

$$\forall(E' \subsetneq E)\exists(\mathbf{x} \in \mathbb{F}).\ \left[\bigwedge\nolimits_{l_i \in E'}(l_i)\right] \land (\kappa(\mathbf{x}) \neq \pi) \tag{2b}$$

Given a DT $\mathcal{T}$ and some path R_k, associated with some prediction π, we say that path R_k is *explanation-redundant* (or simply *redundant*) if R_k is not a PI-explanation of π given the ML model $\mathcal{T}$. If we associate DT paths with instance explanations, then path explanation-redundancy will manifest itself in instance explanations. The concept of explanation-redundancy is illustrated in Example 1.

Example 1. Consider the DT in Figure 1a, for function $f(x_1,x_2) = x_1 \lor x_2$, and instance $(0,1)$. The path corresponds to the explanation $\{(x_1,0),(x_2,1)\}$ for prediction $f(0,1) = 1$. Clearly, this path is explanation-redundant, as a PI-explanation for prediction 1 is $(x_2,1)$ (as is readily concluded from the function definition).
As a less abstract example, we observe that the tree in Figure 1a also models the example DT used in [78, Ch. 01,page 5][8], with x_1 denoting *"is $y > 0.73$?"*, x_2 denoting *"is $x > 0.64$?"*, class 1 denoting *cross*, and class 0 denoting *circle*. Hence, a PI-explanation for the instance (*no*,*yes*) with prediction "cross" is *yes* to question *"is $x > 0.64$?"*, independently of the answer to question *"is $y > 0.73$?"*. Section 3 investigates explanation-redundancy in greater detail.

Related work. As referenced in Section 1, there exists a growing body of work on exploiting DTs for interpretability. To our best knowledge, the assessment of paths in DTs when compared to PI-explanations has not been investigated. Recent work [18] outlines logical encodings of decision trees, but that is orthogonal to the work reported in this paper. In addition, there has been work on applying explainable AI (XAI) to decision trees [37], but with the focus of improving the quality of local (heuristic) explanations,

[8] This DT is shown in the supplementary material.

where the goal is to relate a local approximate model against a reference model; hence there is no immediate relationship with PI-explanations.

3 Decision Trees May Not be Interpretable

This section shows that there exist Boolean functions for which a learned decision tree will exhibit paths containing all features, and for which a PI-explanation has a constant size. Thus, if we associate explanations with DT paths, there will be explanations that are arbitraly larger (on n) than the actual (constant-size) PI-explanation. As the experimental results demonstrate, and as discussed later in this section, it is fairly frequent in practice for DT paths to include more literals that those in the associated PI-explanations.

Proposition 1. *There exist functions for which an irreducible DT contains paths which are a proper superset of a PI-explanation. Furthermore, the difference in the number of literals is $n-k$, where n is the number of features and k is the (constant) size of a PI-explanation.*

Proof. Let us consider the following Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ (with even n):

$$f(x_1, x_2, \ldots, x_{n-1}, x_n) = \bigvee_{i=1}^{n/2} x_{2i-1} \land x_{2i} \tag{3}$$

For the case $n = 4$, different off-the-self DT learners will obtain the DT shown in Figure 1b. (To obtain the decision tree, we considered a dataset composed of *all* possible instances, and used ITI [64][9].) Furthermore, it is immediate to conclude that the decision tree shown is irreducible (i.e. no nodes can be removed while keeping accuracy). Moreover, let the target instance be $(\mathbf{v}, \boldsymbol{\pi}) = ((1,0,1,1),1)$. In this case, the explanation (i.e. the path) extracted from the DT is $(x_1 = 1) \land (x_2 = 0) \land (x_3 = 1) \land (x_4 = 1)$, which guarantees that the prediction is 1. However, it is immediate from the function definition (3), that $(x_3 = 1) \land (x_4 = 1)$ entails $f(x_1, x_2, x_3, x_4) = 1$, *independently* of the value assigned to x_1 and x_2, i.e. in this case the PI-explanation is $(x_3 = 1) \land (x_4 = 1)$. The same analysis generalizes to an arbitrary number of variables. For an instance of the form $((1,0,1,0,\ldots,1,0,1,1),1)$, the DT would indicate an explanation with n literals, whereas the PI-explanation has size 2, namely $(x_{n-1} = 1) \land (x_n = 1)$. □

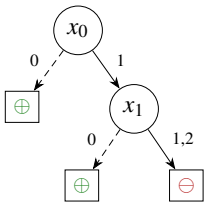
It should be noted that the issue above does not depend on whether the DT is redundant (e.g. in the cases shown, the DTs are *not* redundant); the reported issues result solely from a fundamental limitation of DTs for succinctly representing certain classes of functions[10]. Figure 2 exemplifies that redundancy may occur even in very simple DTs. The DT was obtained with the optimal decision tree package from

[9] ITI is available from https://www-lrn.cs.umass.edu/iti/. We considered a number of publicly available DT learners, and reached the same conclusions (in terms of path length) in all cases.

[10] Indeed, it is well-known that DTs are not as succinct as decision lists (DLs) [52] (and so not as succinct as decision sets (DSs)). This means that there exist functions that have succint DLs or DSs, but not DTs. Although not the focus of the paper, we conjecture that similar results can be obtained for DLs and for restricted cases of DSs, among those where DSs compute functions.

Feature	Name	Domain	Values	Meaning
x_0	Humidity	D_0	$\{0,1\}$	normal,high
x_1	Outlook	D_1	$\{0,1,2\}$	overcast,rain,sunny
x_2	Wind	D_2	$\{0,1\}$	strong,weak

(a) Features for *PlayTennis* dataset



(b) Resulting DT

Fig. 2: Another Example DT

Interpretable AI [10,28][11], where the $\ominus$ and the leftmost $\oplus$ are predicted with 75% and 83.3% confidence, respectively. (The branch annotated with $1,2$ denotes that $x_1 \in \{1,2\}$.) Let the instance be $(\text{Humidity}, \text{Outlook}, \text{Wind}) = (\text{high}, \text{overcast}, \text{weak})$. As an explanation we could use the literals in the tree path, i.e. $\{(\text{Humidity} = \text{high}), (\text{Outlook} = \text{overcast})\}$. However, careful analysis allows us to conclude that $\{(\text{Outlook} = \text{overcast})\}$ suffices to entail the prediction $\oplus$, i.e. as long as 'Outlook' is 'overcast', the prediction will be $\oplus$ *independently* of the value of 'Humidity'.

4 Extracting PI-Explanations from DTs

Deciding explanation-redundancy with NP oracles. Let us consider a decision tree $\mathcal{T}$, with sets of paths $\mathcal{P}$ and $\mathcal{Q}$, denoting respectively the paths with prediction $\oplus$ and $\ominus$. Let us also consider an instance $(\mathbf{v}, \oplus)$, with $\mathbf{v} \in \mathbb{F}$ and $\oplus \in \mathcal{K}$ (the case for $\ominus$ would be similar), and let P_k denote that path consistent with $\mathbf{v}$. To decide whether P_k exhibits explanation-redundancy, one possible solution is to use an NP oracle.

For an instance consistent with P_k, we can model the prediction of the decision tree (for prediction $\oplus$) as follows:

$$\bigwedge\nolimits_{l_j \in \mathcal{L}(P_k)} (l_j) \models \bigvee\nolimits_{P_i \in \mathcal{P}} \bigwedge\nolimits_{l_s \in \mathcal{L}(P_i)} (l_s) \tag{4}$$

Hence, we require the unsatisfiability of,

$$\bigwedge\nolimits_{l_j \in \mathcal{L}(P_k)} (l_j) \land \bigwedge\nolimits_{P_i \in \mathcal{P}} \bigvee\nolimits_{l_s \in \mathcal{L}(P_i)} (\neg l_s) \tag{5}$$

Now, if there exists a literal l_j that can be dropped from P_k such that unsatisfiability is preserved, then P_k exhibits explanation-redundancy. Hence, we need at most m calls to an NP oracle to decide explanation-redundancy, where m is the number of features. (This high-level procedure was proposed in earlier work in more general terms [29].) However, the special structure of a DT, makes the problem far simpler, and can be solved in polynomial time, as shown next.

[11] We used the well-known *PlayTennis* dataset [39].

Deciding explanation-redundancy in linear time. Observe that (4) is preserved iff at least one of the features with a literal in P_k has another literal that is false along *any* path that yields prediction $\ominus$. Thus, there is explanation-redundancy if there exists a literal from P_k that can be dropped while the remaining literals in P_k still guarantee that at least one literal is false along any path in $\mathcal{Q}$. Before detailing a polynomial time algorithm for deciding explanation-redundancy, let us consider a concrete example.

Example 2. For the DT from Figure 1b we have,

$$\begin{aligned}
&\mathcal{P} = \{P_1, P_2, P_3\}\\
&\mathcal{L}(P_1) = \{(x_1 = 0), (x_3 = 1), (x_4 = 1)\}\\
&\mathcal{L}(P_2) = \{(x_1 = 1), (x_2 = 0), (x_3 = 1), (x_4 = 1)\}\\
&\mathcal{L}(P_3) = \{(x_1 = 1), (x_2 = 1)\}\\
&\mathcal{Q} = \{Q_1, Q_2, Q_3, Q_4\}\\
&\mathcal{L}(Q_1) = \{(x_1 = 0), (x_3 = 0)\}\\
&\mathcal{L}(Q_2) = \{(x_1 = 0), (x_3 = 1), (x_4 = 0)\}\\
&\mathcal{L}(Q_3) = \{(x_1 = 1), (x_2 = 0), (x_3 = 0)\}\\
&\mathcal{L}(Q_4) = \{(x_1 = 1), (x_2 = 0), (x_3 = 1), (x_4 = 0)\}
\end{aligned}$$

We consider path P_2 (and so any feature space point consistent with P_2). We can readily conclude that if literal $(x_1 = 1)$ is removed from the literals of P_2, all the paths in $\mathcal{Q}$ remain inconsistent. This is true for example because $(x_3 = 1)$ and $(x_4 = 1)$. Similarly, we could drop literal $(x_2 = 0)$.

The example above naturally suggests a quadratic time (on the size n of the decision tree) algorithm to decide whether a tree path exhibits explanation-redundancy. Concretely, for each literal in P_k, we analyze each path in $\mathcal{Q}$ whether it is still inconsistent. If all paths in $\mathcal{Q}$ remain inconsistent, then the literal can be dropped, and the path exhibits explanation-redundancy. Nevertheless, it is possible to devise a more efficient solution, one that runs in linear time.

The proposed algorithm analyzes the features containing literals in P_i[12], in turn allowing each to be declared *universal*, i.e. the feature can take any value. For each feature f_j, the algorithm recursively analyzes paths with a different prediction, checking whether each such path is inconsistent, due to some other literal. If that is the case, the path P_i is explanation-redundant, at least due to feature f_j. The algorithm analyzes the internal nodes of P_i in reverse order, starting from deepest non-terminal node in the path. (For now, we assume that P_i has at most one literal on any given feature; this restriction will be lifted below.) For each non-terminal node $p_j \in P_i$, the associated feature f_j is made universal, i.e. it can take *any* value. Starting at p_j, all child nodes not in P_i are recursively visited, checking for a consistent sub-path (starting at p_j) to a terminal node with prediction $\ominus$. If such path exists, then f_j cannot be made universal, and so it cannot be discarded from a PI-explanation. Otherwise, all sub-paths to prediction $\ominus$ are inconsistent, and so the value of f_j is irrelevant for the prediction. A path is declared redundant iff at least one feature is declared redundant. With filtering (i.e. $\texttt{rec} = 0$), each tree node is analyzed at most once, and so the amortized

[12] As before, we assume a path P_i with prediction $\oplus$.

Algorithm 1: Deciding path redundancy

```
Function DecidePathRedundancy(T)
    foreach R_k ∈ T do
        A ← AggrFeatureNodes(T, R_k) ;
        N ← PathNodes(T, R_k) ;
        isPathRed ← false ;
        foreach f ∈ PathFeatures(T, R_k) do
            isFeatRed ← true ;
            SetUniversal(T, f) ;
            foreach n ∈ A(f) do
                if not ChkDown(T, N, R_k, n, 0) then
                    isFeatRed ← false ;
                    break;
            if isFeatRed then
                isPathRed ← true ;
                break;
        ReportPath(R_k, isPathRed) ;
```

run time of the algorithm over all features is $\mathcal{O}(|\mathcal{T}|)$. If a feature is tested more than once along P_i, the algorithm requires minor modifications. In this case, a decision about whether feature f_i can take any value, can only be made once all nodes involving f_i have been analyzed and, for all such nodes, the feature f_i has been declared redundant. Algorithms 1 and 2 summarize the two main steps of the proposed algorithm. The auxiliar functions serve to test/set whether a feature is universal (i.e. whether it can take any value of its domain) (resp. `Universal`/`SetUniversal`), aggregate the nodes associated with a given feature along some path (`AggrFeatureNodes`), list the nodes in a given path (`PathNodes`), get the prediction of some path (`Prediction`), list the child nodes of some node in the tree (`ChildNodes`), get the feature associated with a node (`Feature`), check whether some node is terminal (`IsTerminal`) and, finally, whether the sub-paths starting at some node can reach a prediction other than the one associated with the target path R_k (`HasPaths`). Finally, the argument `rec` of ChkDown is a flag that serves to avoid re-visiting already visited paths. For removing redundant features this filtering does not apply, as clarified below.

Example 3. We consider again the DT from Figure 1b and path P_2, where $\mathcal{L}(P_2) = \{(x_1 = 1), (x_2 = 0), (x_3 = 1), (x_4 = 1)\}$, and prediction 1. The nodes of P_2 are analyzed in the order $\langle x_4, x_3, x_2, x_1 \rangle$. Clearly, the sub-path consistent with $(x_4 = 0)$ yields prediction 0. Hence, feature x_4 (associated with literal $x_4 = 1$) is not redundant. For x_3, we consider (*only*) the sub-path corresponding to $(x_3 = 0)$. Again, the prediction is 0, and so the feature x_3 is not redundant. For feature x_2, the only sub-path corresponds to $(x_2 = 1)$, for which the prediction remains unchanged. Hence, x_2 can take *any* value, and so it is declared redundant. As a result, P_2 is also declared redundant. It is helpful to analyze the execution of the algorithm for x_1. The sub-paths consistent with $(x_1 = 1)$ correspond to P_1, Q_1 and Q_2. Hence, due to Q_1 and Q_2, it might seem that x_1 might

Algorithm 2: Inconsistent sub-path lookup

```
Function ChkDown(T, N, R_k, n, rec)
1   π ← Prediction(T, R_k) ;
2   C ← ChildNodes(T, n) ;
3   foreach c ∈ C do
4       if c ∈ N and rec = 0 then continue;
5       if not HasPaths(T, c, π) then continue;
6       if IsTerminal(T, c) then
7           return false ;
8       g ← Feature(T, c) ;
9       if not Universal(T, g) then continue;
10      if not ChkDown(T, N, R_k, c, rec) then
11          return false ;
12  return true;
```

be irredundant. However, both Q_1 and Q_2 are inconsistent with other non-redundant literals of P_2, concretely $x_3 = 1$ and $x_4 = 1$. Hence, x_1 is declared redundant.

Extracting one PI-explanation. One approach to find a PI-explanation is to use recent work based on compilation [60] or iterative entailment checks with an NP oracle [29]. However, a computationally simpler solution is based on the ideas described above.

Features are analyzed as proposed in Algorithms 1 and 2. However, a feature already declared as redundant signifies that it can take *any* value from its domain. This may allow inconsistent paths with prediction $\ominus$ to become consistent, thus preventing some other feature from being declared redundant. One consequence is that the filtering of paths exploited in Algorithms 1 and 2 is not longer applicable. Concretely, the analysis of each feature requires visiting all of the sub-paths starting in any of the path nodes testing the feature. Algorithm 2 can still be used, in this case by setting `rec` to 1, i.e. all sub-paths will be analyzed. The worst-case complexity of the resulting algorithm is $\mathcal{O}(|\mathcal{T}|)$ for each feature, thus yielding $\mathcal{O}(|\mathcal{T}|^2)$ for the complete algorithm. Finally, we can conclude that an algorithm for finding one PI-explanation for each path in $\mathcal{T}$ runs in $\mathcal{O}(|\mathcal{T}|^3)$.

Example 4. We consider the DT from Figure 1b and analyze path P_2 (and so *any* instance consistent with P_2). Literals are analyzed in reverse path order. (In this case, aggregation of nodes by feature is optional, as long as a decision with respect to a feature is delayed until all nodes associated with the feature have been analyzed, and keeping track whether non-irredundancy applies to all nodes.) As before, the literal $(x_4 = 1)$ is not redundant; otherwise Q_4 would be consistent. Similarly, the literal $(x_3 = 1)$ is not redundant; otherwise Q_3 would be consistent. In contrast, the feature x_2 can be made universal, as this does not change the prediction, i.e. P_2 is consistent with the prediction, and the other literals $(x_3 = 1)$ and $(x_4 = 1)$ block the paths in $\mathcal{Q}$. A similar analysis applies in the case of feature x_1. In this case, the algorithm analyzes all paths

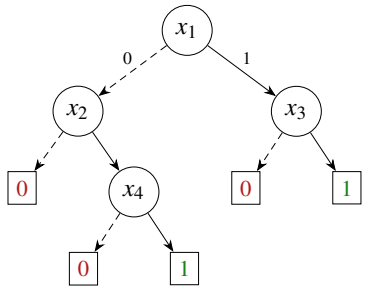


Fig. 3: Path-restricted vs. path-unrestricted explanations

(since Algorithm 2 is invoked with $\texttt{rec} = 1$). Due to the literals $(x_3 = 1)$ and $(x_4 = 1)$, all paths in $\mathcal{Q}$ are inconsistent. As a result, feature x_1 can be declared redundant, and a PI-explanation for P_2 is thus $\{(x_3 = 1), (x_4 = 1)\}$.

Path-restricted vs. path-unrestricted explanations. For the algorithms described earlier we also need to decide the set of literals to consider. One option is the set of literals specified by the instance. Another option is the set of literals specific to the tree path consistent with the instance. If we are interested in finding PI-explanations for the prediction, given the instance, then we should consider the literals specified by the instance. However, if we want to report explanations that relate with the tree path consistent with the instance, then we should consider only the literals in the tree path. Clearly, path-restricted PI-explanations are a subset of path-unrestricted PI-explanations. The following example illustrates the differences between the two approaches.

Example 5. We consider the example DT shown in Figure 3. Given the point $(x_1, x_2, x_3, x_4) = (1, 1, 1, 1)$, the consistent path in the DT consists of the literals $\{(x_1 = 1), (x_3 = 1)\}$. The only PI-explanation that is restricted to this path is the path itself: $\{(x_1 = 1), (x_3 = 1)\}$. However, if we enable path-unrestricted explanations, we will also obtain the PI-explanation $\{(x_2 = 1), (x_3 = 1), (x_4 = 1)\}$, which is not even shown as (part of) a path in the decision tree.

5 Enumeration of PI-Explanations

The enumeration of multiple (or all) PI-explanations can help human decision makers to develop a better understanding of some prediction, but also of the underlying ML model. Recent work [60] compiles a decision function into a Sentential Decision Diagram (SDD), from which the enumeration of PI-explanations can be instrumented. Moreover, from a compiled representation of the PI-explanations, each PI-explanation can be reported in polynomial time. The downside is that these representation are worst-case exponential in the size of the original ML model. Another line of work for computing PI-explanations is based on iterative entailment checks using an NP-oracle [29]. However, this recent work does not address the enumeration of PI-explanations. This section develops a solution for the enumeration of PI-explanations in the case of DTs, by reduction to the enumeration of minimal hitting sets (MHSes).

We consider the situation where the prediction is $\oplus$ for some point $\mathbf{v}$ in feature space, which is consistent with some tree path $P_k \in \mathcal{P}$. As a result, each path in $\mathcal{Q}$ is inconsistent with at least one literal, among those either associated with $\mathbf{v}$ or with P_k. Let the set of literals considered be R. For each path $Q_s \in \mathcal{Q}$, let L_s denote the set of literals in R that are inconsistent with Q_s. For a subset S of R to entail the prediction, it must hit each set of literals L_s. Among the possible sets S, each subset-minimal set is a PI-explanation. Thus, we can list the PI-explanations (starting from the literals taken from $\mathbf{v}$ or from P_k) by enumerating minimal hitting sets.

Example 6. For P_2 in the DT of Figure 1, the sets to hit are:

$$Q_1 : \{(x_1 = 1), (x_3 = 1)\} \quad Q_2 : \{(x_1 = 1), (x_4 = 1)\}$$
$$Q_3 : \{(x_3 = 1)\} \quad Q_4 : \{(x_4 = 1)\}$$

In this case, the only MHS is $\{(x_3 = 1), (x_4 = 1)\}$, representing a single PI-explanation $\{(x_3 = 1), (x_4 = 1)\}$.

Example 7. For the DT in Figure 3, and by considering the literals from the point $\mathbf{v} = (1, 1, 1, 1)$, the sets to hit are then:

$$Q_1 : \{(x_1 = 1), (x_2 = 1)\} \quad Q_2 : \{(x_1 = 1), (x_4 = 1)\}$$
$$Q_3 : \{(x_3 = 1)\}$$

The MHSes are $\{(x_1 = 1), (x_3 = 1)\}$ and $\{(x_2 = 1), (x_4 = 1), (x_3 = 1)\}$, each denoting a path-unrestricted PI-explanation.

6 Experimental Results

This section presents a summary of experimental evaluation of the explanation-redundancy of two state-of-the-art heuristic DT classifiers. Concretely, we use the well-known DT training tools *ITI* (*Incremental Tree Induction*) [64,30] and *IAI* (*Interpretable AI*) [10,28]. ITI is run with the pruning option enabled, which helps avoiding overfitting and aims at constructing shallow DTs. To enforce IAI to produce shallow DTs and achieve high accuracy, it is set to use the optimal tree classifier method with the maximal depth of 6[13]. The experiments consider datasets with categorical (non-binarized) data, which both ITI and IAI can handle[14]. The assessment is performed on a selection of 80 publicly available datasets, which originate from *UCI Machine Learning Repository* [63], *Penn Machine Learning Benchmarks* [48], and *OpenML repository* [46]. (Due to space restrictions, we report the results only for 30 datasets but the results shown extend to the complete benchmark set, and are included in the supplementary materials.) The number of features (data instances, resp.) in the benchmark suite vary from 2 to 118 (106 to 58000, resp.) with the average being 31.2 (6045.3, resp.).

[13] Our results confirm that larger maximal depths would in most cases increase the percentage of redundant paths. A smaller maximal depth would not improve accuracy.

[14] Other known DT learning tools, including scikit-learn [47] and DL8.5 [1,2] can only handle numerical and binary features, respectively, and so could not be included in the experiments.

Dataset	(#F	#S)	IAI									ITI								
			D	#N	%A	#P	%R	%C	%m	%M	%avg	D	#N	%A	#P	%R	%C	%m	%M	%avg
adult	(12	6061)	6	83	78	42	33	25	20	40	25	17	509	73	255	75	91	10	66	22
anneal	(38	886)	6	29	99	15	26	16	16	33	21	9	31	100	16	25	4	12	20	16
backache	(32	180)	4	17	72	9	33	39	25	33	30	3	9	91	5	80	87	50	66	54
bank	(19	36293)	6	113	88	57	5	12	16	20	18	19	1467	86	734	69	64	7	63	27
biodegradation	(41	1052)	5	19	65	10	30	1	25	50	33	8	71	76	36	50	8	14	40	21
cancer	(9	449)	6	37	87	19	36	9	20	25	21	5	21	84	11	54	10	25	50	37
car	(6	1728)	6	43	96	22	86	89	20	80	45	11	57	98	29	65	41	16	50	30
colic	(22	357)	6	55	81	28	46	6	16	33	20	4	17	80	9	33	27	25	25	25
compas	(11	1155)	6	77	34	39	17	8	16	20	17	15	183	37	92	66	43	12	60	27
contraceptive	(9	1425)	6	99	49	50	8	2	20	60	37	17	385	48	193	27	32	12	66	21
dermatology	(34	366)	6	33	90	17	23	3	16	33	21	7	17	95	9	22	0	14	20	17
divorce	(54	150)	5	15	90	8	50	19	20	33	24	2	5	96	3	33	16	50	50	50
german	(21	1000)	6	25	61	13	38	10	20	40	29	10	99	72	50	46	13	12	40	22
heart-c	(13	302)	6	43	65	22	36	18	20	33	22	4	15	75	8	87	81	25	50	34
heart-h	(13	293)	6	37	59	19	31	4	20	40	24	8	25	77	13	61	60	20	50	32
kr-vs-kp	(36	3196)	6	49	96	25	80	75	16	60	33	13	67	99	34	79	43	7	70	35
lending	(9	5082)	6	45	73	23	73	80	16	50	25	14	507	65	254	69	80	12	75	25
letter	(16	18668)	6	127	58	64	1	0	20	20	20	46	4857	68	2429	6	7	6	25	9
lymphography	(18	148)	6	61	76	31	35	25	16	33	21	6	21	86	11	9	0	16	16	16
mortality	(118	13442)	6	111	74	56	8	14	16	20	17	26	865	76	433	61	61	7	54	19
mushroom	(22	8124)	6	39	100	20	80	44	16	33	24	5	23	100	12	50	31	20	40	25
pendigits	(16	10992)	6	121	88	61	0	0	—	—	—	38	937	85	469	25	86	6	25	11
promoters	(58	106)	1	3	90	2	0	0	—	—	—	3	9	81	5	20	14	33	33	33
recidivism	(15	3998)	6	105	61	53	28	22	16	33	18	15	611	51	306	53	38	9	44	16
seismic_bumps	(18	2578)	6	37	89	19	42	19	20	33	24	8	39	93	20	60	79	20	60	42
shuttle	(9	58000)	6	63	99	32	28	7	20	33	23	23	159	99	80	33	9	14	50	30
soybean	(35	623)	6	63	88	32	9	5	25	25	25	16	71	89	36	22	1	9	12	10
spambase	(57	4210)	6	63	75	32	37	12	16	33	19	15	143	91	72	76	98	7	58	25
spect	(22	228)	6	45	82	23	60	51	20	50	35	6	15	86	8	87	98	50	83	65
splice	(2	3178)	3	7	50	4	0	0	—	—	—	88	177	55	89	0	0	—	—	—

Table 1: Explanation-redundancy in decision trees obtained with IAI and ITI.

The experiments are performed on a MacBook Pro with a Dual-Core Intel Core i5 2.3GHz CPU with 8GByte RAM running macOS Catalina. The polynomial-time explanation-redundancy check and a single PI-explanation extraction proposed in Section 4 are implemented in Perl. (An implementation using the Glucose SAT solver was instrumental in validating the results, but for the DTs considered, it was in general slower by at least one order of magnitude.) Performance-wise, training DTs with IAI takes from 4sec to 2310sec with the average run time per dataset being 70sec. In contrast, the time spent on eliminating explanation-redundancy is *negligible*, taking from 0.026sec to 0.4sec per tree, with an average time of 0.06sec. ITI runs much faster that IAI and takes from 0.1sec to 2sec with 0.1sec on average; the elimination of explanation redundancy is slightly more time consuming than for IAI, taking from 0.025sec to 5.4sec with 0.29sec on average. This slowdown results from DTs learned with ITI being deeper on average, and features being tested multiple times.

The summary of results is detailed in Table 1. For each dataset, the table reports the number of features and also instances as *#F* and *#S*, respectively. Thereafter, it shows tree statistics for IAI and ITI, namely, tree depth *D*, number of nodes *#N*, test accuracy *%A* and number of paths *#P*. The percentage of explanation-redundant paths is given as *%R* while the percentage of data instances (measured for the *entire* feature space) covered by redundant paths is *%C*. Focusing solely on the explanation-redundant paths, a single PI-explanation is extracted and the average (min. or max., resp.) percentage of redundant literals per path is denoted by *%avg* (*%m* and *%M*, resp.). Observe that despite the shallowness of the trees produced by IAI and ITI, for the majority of datasets and with a few exceptions, the paths in trees trained by both tools exhibit significant

explanation-redundancy. In particular, on average, 32.1% (46.9%, resp.) of paths are explanation-redundant for the trees obtained by IAI (ITI, resp.). For some DTs, obtained with either IAI and ITI, more than 85% of tree paths are redundant. Also, redundant paths of the trees of IAI (ITI, resp.) cover on average 20.1% (37.7%, resp.) of feature space. Moreover, in some cases, up to 89% and 98% of the entire feature space is covered by the redundant paths for IAI and ITI, respectively. This means that DTs produced by IAI and ITI are unable to provide a user with a succinct explanation for the *vast majority* of data instances. In addition, the average number of redundant literals in redundant paths for both IAI and ITI varies from 16% to 65%, but for some DTs it exceeds 80%.

To summarize, the numbers shown for the selected datasets and for the state-of-the-art DT training tools contrast the common belief in the inherent interpretability of decision tree classifiers. Perhaps as importantly, the performance figures confirm that the elimination of explanation-redundancy in the DTs produced with available tools has negligible computational cost.

7 Conclusions

Decision trees are most often associated with interpretability. This paper shows that in some situations, paths in a decision tree may include many literals that are irrelevant for an explanation, and that this holds true even for irreducible decision trees. Moreover, the paper proposes a linear time test to decide whether a decision tree path contains irrelevant literals, and uses such test to devise a polynomial time algorithm for computing one PI-explanation of a decision tree. Furthermore, the paper shows the connection between enumerating the PI-explanations of DTs and the enumeration of minimal hitting sets. Experimental results obtained on publicly available datasets, using state-of-the-art decision tree learners, show that in practice induced paths in decision trees may contain irrelevant literals, even when the decision tree is irreducible. For the decision trees considered in the experiments, the run times of the proposed algorithms are either negligible or comparable to tree learning times.

References

1. G. Aglin, S. Nijssen, and P. Schaus. Learning optimal decision trees using caching branch-and-bound search. In *AAAI*, pages 3146–3153, 2020.
2. G. Aglin, S. Nijssen, and P. Schaus. PyDL8.5: a library for learning optimal decision trees. In *IJCAI*, pages 5222–5224, 2020.
3. E. Alpaydin. *Machine Learning: The New AI*. MIT Press, 2016.
4. E. Angelino, N. Larus-Stone, D. Alabi, M. Seltzer, and C. Rudin. Learning certifiably optimal rule lists. In *KDD*, pages 35–44, 2017.
5. E. Angelino, N. Larus-Stone, D. Alabi, M. Seltzer, and C. Rudin. Learning certifiably optimal rule lists for categorical data. *J. Mach. Learn. Res.*, 18:234:1–234:78, 2017.
6. R. Appuswamy, M. Franceschetti, N. Karamchandani, and K. Zeger. Network coding for computing: Cut-set bounds. *IEEE Trans. Inf. Theory*, 57(2):1015–1030, 2011.
7. F. Avellaneda. Efficient inference of optimal decision trees. In *AAAI*, pages 3195–3202, 2020.

8. O. Bastani, C. Kim, and H. Bastani. Interpretability via model extraction. *CoRR*, abs/1706.09773, 2017.
9. O. Bastani, C. Kim, and H. Bastani. Interpreting blackbox models via model extraction. *CoRR*, abs/1705.08504, 2017.
10. D. Bertsimas and J. Dunn. Optimal classification trees. *Mach. Learn.*, 106(7):1039–1082, 2017.
11. C. Bessiere, E. Hebrard, and B. O'Sullivan. Minimising decision tree size as combinatorial optimisation. In *CP*, pages 173–187, 2009.
12. M. Bramer. *Principles of Data Mining, Third Edition*. Springer, 2016.
13. L. Breiman. Statistical modeling: The two cultures. *Statistical science*, 16(3):199–231, 2001.
14. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.
15. L. A. Breslow and D. W. Aha. Simplifying decision trees: A survey. *Knowledge Eng. Review*, 12(1):1–40, 1997.
16. C. E. Brodley and P. E. Utgoff. Multivariate decision trees. *Mach. Learn.*, 19(1):45–77, 1995.
17. C. Chen and C. Rudin. An optimization approach to learning falling rule lists. In *AISTATS*, pages 604–612, 2018.
18. A. Choi, A. Shih, A. Goyanka, and A. Darwiche. On symbolically encoding the behavior of random forests. *CoRR*, abs/2007.01493, 2020.
19. P. A. Flach. *Machine Learning - The Art and Science of Algorithms that Make Sense of Data*. CUP, 2012.
20. A. A. Freitas. Comprehensible classification models: a position paper. *SIGKDD Explorations*, 15(1):1–10, 2013.
21. N. Frosst and G. E. Hinton. Distilling a neural network into a soft decision tree. In *CExAIIA*, 2017.
22. M. R. Garey. Optimal binary identification procedures. *SIAM Journal on Applied Mathematics*, 23(2):173–186, 1972.
23. R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5):93:1–93:42, 2019.
24. A. Holzinger, G. Langs, H. Denk, K. Zatloukal, and H. Müller. Causability and explainability of artificial intelligence in medicine. *WIRE Data Min. Knowl. Discov.*, 9(4), 2019.
25. H. Hu, M. Siala, E. Hebrard, and M. Huguet. Learning optimal decision trees with MaxSAT and its integration in adaboost. In *IJCAI*, pages 1170–1176, 2020.
26. X. Hu, C. Rudin, and M. Seltzer. Optimal sparse decision trees. In *NeurIPS*, pages 7265–7273, 2019.
27. L. Hyafil and R. L. Rivest. Constructing optimal binary decision trees is np-complete. *Inf. Process. Lett.*, 5(1):15–17, 1976.
28. IAI. Interpretable AI. https://www.interpretable.ai/, 2020.
29. A. Ignatiev, N. Narodytska, and J. Marques-Silva. Abduction-based explanations for machine learning models. In *AAAI*, pages 1511–1519, 2019.
30. Incremental Decision Tree Induction. https://www-lrn.cs.umass.edu/iti/, 2020.
31. M. Janota and A. Morgado. SAT-based encodings for optimal decision trees with explicit paths. In *SAT*, pages 501–518, 2020.
32. S. B. Kotsiantis. Decision trees: a recent overview. *Artif. Intell. Rev.*, 39(4):261–283, 2013.
33. H. Lakkaraju, S. H. Bach, and J. Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *KDD*, pages 1675–1684, 2016.
34. H. Lakkaraju, E. Kamar, R. Caruana, and J. Leskovec. Interpretable & explorable approximations of black box models. *CoRR*, abs/1707.01154, 2017.
35. B. Letham, C. Rudin, T. H. McCormick, and D. Madigan. An interpretable stroke prediction model using rules and bayesian analysis. In *AAAI*, 2013.

36. Z. C. Lipton. The mythos of model interpretability. *Commun. ACM*, 61(10):36–43, 2018.
37. S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee. From local explanations to global understanding with explainable AI for trees. *Nature machine intelligence*, 2(1):2522–5839, 2020.
38. T. Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artif. Intell.*, 267:1–38, 2019.
39. T. M. Mitchell. *Machine learning*. McGraw-Hill, 1997.
40. C. Molnar. *Interpretable Machine Learning*. 2019. https://christophm.github.io/interpretable-ml-book/.
41. G. Montavon, W. Samek, and K. Müller. Methods for interpreting and understanding deep neural networks. *Digit. Signal Process.*, 73:1–15, 2018.
42. B. M. E. Moret. Decision trees and diagrams. *ACM Comput. Surv.*, 14(4):593–623, 1982.
43. N. Narodytska, A. Ignatiev, F. Pereira, and J. Marques-Silva. Learning optimal decision trees with SAT. In *IJCAI*, pages 1362–1368, 2018.
44. S. Nijssen and É. Fromont. Mining optimal decision trees from itemset lattices. In *KDD*, pages 530–539, 2007.
45. S. Nijssen and É. Fromont. Optimal constraint-based decision tree induction from itemset lattices. *Data Min. Knowl. Discov.*, 21(1):9–51, 2010.
46. OpenML: Machine learning, better, together. https://www.openml.org/, 2020.
47. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. VanderPlas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12:2825–2830, 2011.
48. Penn Machine Learning Benchmarks. https://github.com/EpistasisLab/penn-ml-benchmarks, 2020.
49. D. Poole and A. K. Mackworth. *Artificial Intelligence - Foundations of Computational Agents*. CUP, 2017.
50. J. R. Quinlan. Induction of decision trees. *Mach. Learn.*, 1(1):81–106, 1986.
51. M. T. Ribeiro, S. Singh, and C. Guestrin. Model-agnostic interpretability of machine learning. *CoRR*, abs/1606.05386, 2016.
52. R. L. Rivest. Learning decision lists. *Mach. Learn.*, 2(3):229–246, 1987.
53. L. Rokach and O. Maimon. Top-down induction of decision trees classifiers - a survey. *IEEE Trans. Syst. Man Cybern. Part C*, 35(4):476–487, 2005.
54. L. Rokach and O. Maimon. *Data Mining with Decision Trees - Theory and Applications*. WorldScientific, 2007.
55. A. M. Roth, N. Topin, P. Jamshidi, and M. Veloso. Conservative Q-improvement: Reinforcement learning for an interpretable decision-tree policy. *CoRR*, abs/1907.01180, 2019.
56. C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.
57. C. Rudin and S. Ertekin. Learning customized and optimized lists of rules with mathematical programming. *Math. Program. Comput.*, 10(4):659–702, 2018.
58. S. J. Russell and P. Norvig. *Artificial Intelligence - A Modern Approach*. Pearson Education, 2010.
59. W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, and K. Müller, editors. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 2019.
60. A. Shih, A. Choi, and A. Darwiche. A symbolic approach to explaining bayesian network classifiers. In *IJCAI*, pages 5103–5111, 2018.
61. A. Silva, M. C. Gombolay, T. W. Killian, I. D. J. Jimenez, and S. Son. Optimization methods for interpretable differentiable decision trees applied to reinforcement learning. In *AISTATS*, pages 1855–1865, 2020.

62. K. Sokol and P. A. Flach. Desiderata for interpretability: Explaining decision tree predictions with counterfactuals. In *AAAI*, pages 10035–10036, 2019.
63. UCI Machine Learning Repository. https://archive.ics.uci.edu/ml, 2020.
64. P. E. Utgoff, N. C. Berkman, and J. A. Clouse. Decision tree induction based on efficient tree restructuring. *Mach. Learn.*, 29(1):5–44, 1997.
65. G. Valdes, J. M. Luna, E. Eaton, C. B. Simone II, L. H. Ungar, and T. D. Solberg. Mediboost: a patient stratification tool for interpretable decision making in the era of precision medicine. *Nature Scientific reports*, 6:37854, 2016.
66. H. Verhaeghe, S. Nijssen, G. Pesant, C. Quimper, and P. Schaus. Learning optimal decision trees using constraint programming. In *BNAIC*, 2019.
67. H. Verhaeghe, S. Nijssen, G. Pesant, C. Quimper, and P. Schaus. Learning optimal decision trees using constraint programming. In *IJCAI*, pages 4765–4769, 2020.
68. S. Verwer and Y. Zhang. Learning decision trees with flexible constraints and objectives using integer optimization. In *CPAIOR*, volume 10335, pages 94–103, 2017.
69. S. Verwer and Y. Zhang. Learning optimal classification trees using a binary linear program formulation. In *AAAI*, pages 1625–1632, 2019.
70. F. Wang and C. Rudin. Falling rule lists. In *AISTATS*, 2015.
71. T. Wang, C. Rudin, F. Doshi-Velez, Y. Liu, E. Klampfl, and P. MacNeille. Bayesian rule sets for interpretable classification. In *ICDM*, pages 1269–1274, 2016.
72. T. Wang, C. Rudin, F. Doshi-Velez, Y. Liu, E. Klampfl, and P. MacNeille. A bayesian framework for learning rule sets for interpretable classification. *J. Mach. Learn. Res.*, 18:70:1–70:37, 2017.
73. M. Wu, M. C. Hughes, S. Parbhoo, M. Zazzi, V. Roth, and F. Doshi-Velez. Beyond sparsity: Tree regularization of deep models for interpretability. In *AAAI*, pages 1670–1678, 2018.
74. M. Wu, S. Parbhoo, M. C. Hughes, R. Kindle, L. A. Celi, M. Zazzi, V. Roth, and F. Doshi-Velez. Regional tree regularization for interpretability in deep neural networks. In *AAAI*, pages 6413–6421, 2020.
75. M. Wu, S. Parbhoo, M. C. Hughes, V. Roth, and F. Doshi-Velez. Optimizing for interpretability in deep neural networks with tree regularization. *CoRR*, abs/1908.05254, 2019.
76. F. Xu, H. Uszkoreit, Y. Du, W. Fan, D. Zhao, and J. Zhu. Explainable AI: A brief survey on history, research areas, approaches and challenges. In *NLPCC*, pages 563–574, 2019.
77. H. Yang, C. Rudin, and M. Seltzer. Scalable bayesian rule lists. In *ICML*, volume 70, pages 3921–3930, 2017.
78. Z.-H. Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012.

A Case Studies

A.1 Analysis of DT from Russel&Norvig's Book

Decision Tree This case study considers the decision tree (DT) shown in Figure 4, taken from [58, Ch. 18,page 702]. The example consists in deciding whether to wait for a table at a restaurant. Six features are used in the DT, namely:

- Alternate: whether there is a suitable alternative restaurant nearby.
- Bar: whether the restaurant has a comfortable bar area to wait in.
- Fri/Sat: true on Fridays and Saturdays.
- Hungry: whether the people are hungry.
- Patrons: how many people are in the restaurant (values are None, Some, and Full).
- Type: the kind of restaurant (French, Italian, Thai, or burger).

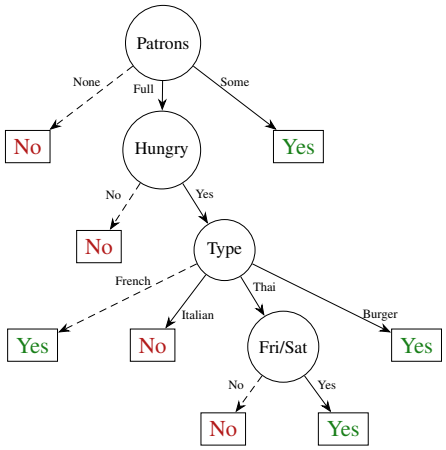


Fig. 4: Example of decision tree from Russel&Norvig's Book.

Redundancy Analysis Results Analysis of the paths in the DT shown in Figure 4 yields the following results.

- path (*Patrons=None*) is explanation-irredundant.
- path (*Patrons=Ful* and *Hungry=No*) is explanation-irredundant.
- path (*Patrons=Ful* and *Hungry=Yes* and *Type=Italian*) is explanation-redundant. If the values of *Patrons* and *Type* are fixed, then the value of *Hungry* is irrelevant for the prediction.
- path (*Patrons=Ful* and *Hungry=Yes* and *Type=Thai* and *Fri/Sat=No*) is explanation-redundant. If the values of *Patrons*, *Type* and *Fri/Sat* are fixed, then the value of *Hungry* is irrelevant for the prediction.
- path (*Patrons=Some*) is explanation-irredundant.
- path (*Patrons=Ful* and *Hungry=Yes* and *Type=Frencg*) is explanation-irredundant.
- path (*Patrons=Ful* and *Hungry=Yes* and *Type=Thai* and *Fri/Sat=Yes*) is explanation-irredundant.
- path (*Patrons=Ful* and *Hungry=Yes* and *Type=Burger*) is explanation-irredundant.

As result, 2 out of 8 paths exhibit explanation-redundancy. Thus, we conclude that the DT exhibits 25% of explanation-redundancy.

A.2 Analysis of DT from Poole&Mackworth's Book

Decision Tree This case study considers the decision tree shown in Figure 5, taken from [49, Ch. 07,page 298]. The example consists in predicting whether a person reads an article posted to a bulletin board given properties of the article. There are three features in the decision tree:

- Author: whether the author is known or not.
- Thread: whether the article started a new thread or was a follow-up.
- Length: the length of the article (short or long).

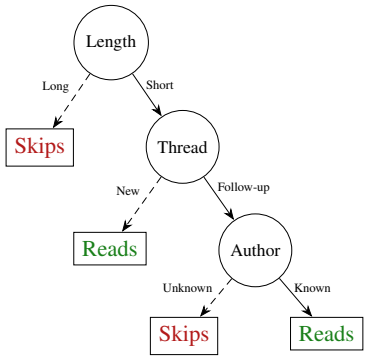


Fig. 5: Example of decision tree from Poole&Mackworth's Book.

Redundancy Analysis Results Analysis of the paths in the DT shown in Figure 5 yields the following results.

- path (*Length=long*) is explanation-irredundant.
- path (*Length=short* and *Thread=follow-up* and *Author=unknown*) is explanation-redundant. If values of *Thread* and *Author* are fixed, then the value of *Length* is irrelevant for the prediction.
- path (*Length=short* and *Thread=new*) is explanation-irredundant.
- path (*Length=short* and *Thread=follow-up* and *Author=known*) is explanation-redundant. If the values of *Length* and *Author* are fixed, then the value of *Thread* is irrelevant for the prediction.

Accordingly, 2 out of 4 paths exhibit explanation-redundancy. Therefore, we say that the DT exhibit 50% of explanation-redundancy.

A.3 Analysis of DT from Z.-H. Zhou's book

Decision Tree This case study considers the decision tree shown in Figure 6, taken from [78, Ch. 01,page 5]. The example consists in predicting the type of a drawing, to be chosen among the classes *cross* and *circle*. Two features are used, namely:

- $x > 0.64 \in \{\mathsf{Y}, \mathsf{N}\}$.
- $y > 0.73 \in \{\mathsf{Y}, \mathsf{N}\}$.

Redundancy Analysis Results Analysis of the paths in the DT shown in Figure 6 yields the following results.

- path ($y > 0.73$) is explanation-irredundant.
- path ($y \leq 0.73$ and $x > 0.64$) is explanation-redundant. If the value of y is fixed, then the value of x is irrelevant for the prediction.
- path ($y \leq 0.73$ and $x \leq 0.64$) is explanation-irredundant.

As a result, 1 out of 3 paths exhibit explanation-redundancy. Thus, we say that the DT exhibits 33.33% of explanation-redundancy.

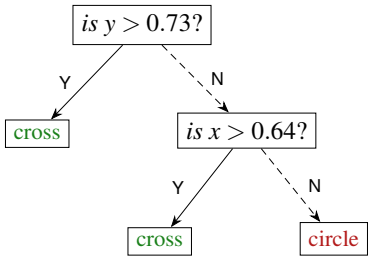


Fig. 6: Example of decision tree from Zhou's book [78].

A.4 Additional Examples

It is interesting to note that the DTs used in a number of books and surveys exhibit explanation-redundancy. A non-exhaustive list of references includes [42,15,53,54,58,19,32,3,12,65,49].

B Full Table of Results

Table 2 presents the experimental results obtained on an extended set of datasets.

Dataset	(#F	#S)	IAI										
			D	#N	%A	#P	%R	%C	%m	%M	%avg	D	#N
IndiansDiabetes	8	768	6	33	67	17	35	2	20	25	21	21	67
adult_data	12	6061	6	83	78	42	33	25	20	40	25	17	509
anneal	38	886	6	29	99	15	26	16	16	33	21	9	31
appendicitis	7	106	2	7	68	4	0	0	—	—	—	3	7
australian	14	690	6	45	61	23	17	8	20	33	23	7	33
auto	25	202	6	33	53	17	23	1	16	33	23	10	47
backache	32	180	4	17	72	9	33	39	25	33	30	3	9
balance	4	625	6	93	81	47	61	41	25	50	27	12	105
bank	19	36293	6	113	88	57	5	12	16	20	18	19	1467
banknote	4	1348	3	9	67	5	20	0	66	66	66	35	71
biodegradation	41	1052	5	19	65	10	30	1	25	50	33	8	71
biomed	8	209	3	9	66	5	20	1	50	50	50	8	33
breast-cancer	9	272	6	61	74	31	41	13	16	33	20	7	25
bupa	6	341	6	47	62	24	29	5	20	33	25	24	49
cancer	9	449	6	37	87	19	36	9	20	25	21	5	21
car	6	1728	6	43	96	22	86	89	20	80	45	11	57
cars	8	392	2	5	100	3	0	0	—	—	—	14	45
cleveland-nominal	7	187	6	81	31	41	24	15	16	33	20	7	65
cloud	7	108	3	9	31	5	0	0	—	—	—	10	21
colic	22	357	6	55	81	28	46	6	16	33	20	4	17
compas	11	1155	6	77	34	39	17	8	16	20	17	15	183

contraceptive	9	1425	6	99	49	50	8	2	20	60	37	17	385
corral	6	64	6	19	92	10	80	50	20	66	42	4	13
dermatology	34	366	6	33	90	17	23	3	16	33	21	7	17
divorce	54	150	5	15	90	8	50	19	20	33	24	2	5
ecoli	7	327	6	45	75	23	4	5	20	20	20	53	109
german_data	21	1000	6	25	61	13	38	10	20	40	29	10	99
glass	9	204	6	35	53	18	0	0	—	—	—	27	65
glass2	9	162	3	11	66	6	0	0	—	—	—	7	15
haberman	3	289	6	55	58	28	14	4	33	33	33	12	35
hayes-roth	4	93	6	23	78	12	25	32	50	66	61	6	17
heart-c	13	302	6	43	65	22	36	18	20	33	22	4	15
heart-h	13	293	6	37	59	19	31	4	20	40	24	8	25
heart-statlog	13	270	6	33	55	17	29	5	20	33	30	4	13
hepatitis	19	155	5	17	77	9	33	6	20	33	24	3	11
house-votes-84	16	298	6	49	91	25	68	67	16	50	27	3	9
hungarian	13	293	6	33	69	17	29	2	25	40	31	4	19
ionosphere	34	350	4	9	70	5	60	3	33	50	38	5	17
iris	4	149	5	23	90	12	41	25	25	33	30	10	21
irish	5	470	4	13	97	7	71	54	33	50	36	3	7
kr-vs-kp	36	3196	6	49	96	25	80	75	16	60	33	13	67
lending_data	9	5082	6	45	73	23	73	80	16	50	25	14	507
letter	16	18668	6	127	58	64	1	0	20	20	20	46	4857
lupus	3	87	2	7	44	4	0	0	—	—	—	1	3
lymphography	18	148	6	61	76	31	35	25	16	33	21	6	21
messidor	19	1146	3	7	50	4	50	1	50	66	58	22	107
meteo	4	14	5	13	33	7	42	33	33	50	38	1	3
molecular-biology_promoters	58	106	6	17	86	9	33	19	16	33	23	3	9
monk1	6	124	4	17	100	9	66	41	25	50	36	5	13
monk2	6	169	6	67	82	34	64	49	16	66	32	5	25
monk3	6	122	6	35	80	18	61	36	20	60	37	2	5
mortality	118	13442	6	111	74	56	8	14	16	20	17	26	865
mouse	5	57	3	9	83	5	20	0	33	33	33	2	5
mushroom	22	8124	6	39	100	20	80	44	16	33	24	5	23
mux6	6	64	6	55	61	28	85	78	20	50	37	4	15
new-thyroid	5	215	3	11	95	6	33	4	33	33	33	14	29
pendigits	16	10992	6	121	88	61	0	0	—	—	—	38	937
postoperative-patient-data	8	78	6	43	50	22	59	44	16	50	25	6	15
primary-tumor	15	228	6	55	71	28	35	21	16	33	21	6	21
promoters	58	106	1	3	90	2	0	0	—	—	—	3	9
recidivism_data	15	3998	6	105	61	53	28	22	16	33	18	15	611
schizo	14	340	6	17	55	9	55	5	50	66	58	13	51
segmentation	19	210	4	15	38	8	0	0	—	—	—	27	57
seismic_bumps	18	2578	6	37	89	19	42	19	20	33	24	8	39
shuttle	9	58000	6	63	99	32	28	7	20	33	23	23	159
soybean	35	623	6	63	88	32	9	5	25	25	25	16	71

spambase	57	4210	6	63	75	32	37	12	16	33	19	15	143
spect	22	228	6	45	82	23	60	51	20	50	35	6	15
splice	2	3178	3	7	50	4	0	0	—	—	—	88	177
student-mat	32	395	6	109	35	55	9	3	20	25	21	22	177
student-por	32	649	6	119	30	60	1	0	20	20	20	22	259
tae	5	110	6	43	40	22	9	6	25	33	29	8	23
titanic	3	24	2	5	40	3	33	25	50	50	50	2	5
tram_2000_side_16x16	256	2000	1	3	100	2	0	0	—	—	—	1	3
uci_mammo_data	13	126	6	53	11	27	51	43	16	40	24	9	23
vehicle	18	846	6	79	49	40	10	0	16	20	19	24	141
wdbc	30	569	2	7	87	4	0	0	—	—	—	57	115
wpbc	33	198	2	5	57	3	33	0	50	50	50	2	5
yeast	9	1462	6	45	49	23	4	0	25	25	25	64	493
zoo	16	59	6	23	91	12	33	7	16	33	21	6	13

Table 2: Explanation-redundancy in decision trees obtained with IAI and I