Formally Correct Search for Interpretable DNFs

Imane Bousdira, Martin Cooper, Aurélie Hurault

Formally Correct Search for Interpretable DNFs

Imane Bousdira[1], Martin Cooper[2], and Aurélie Hurault[1]

[1] IRIT, Toulouse INP, France
[2] IRIT, University of Toulouse, France
{imane.bousdira, cooper, aurelie.hurault}@irit.fr

Abstract. Interpretable models are a key aspect of explainable machine learning. A model can be considered to be interpretable if for each decision there is an explanation involving only k features, for some small constant k. For boolean functions κ this means that both κ and its complement $\overline{\kappa}$ are expressible as k-DNFs. Nested k-DNFs are one such family of interpretable models. We show how to find such models and provide software, based on a formally-verified SAT encoding, to do so. We report experiments indicating that nested DNFs are an interpretable alternative to random forests while retaining the same accuracy.

Keywords: Machine learning · Interpretable models

1 Introduction

Extensive deployment of Machine Learning (ML) must go hand-in-hand with explainable AI, in order to provide explanations to human users of decisions taken by ML models. Interpretable models are particularly user-friendly since they provide a guarantee of human comprehensibility of all decisions taken by the model.

Traditionally, decision trees (DTs) have been advocated as the interpretable alternative to black-box models. In recent work, it was pointed out that DTs are not the only family of interpretable models and a novel alternative (called nested DNFs) was proposed [11]. Whereas the learning and explaining of DTs have been well-studied [20,3,13], this is not the case for other interpretable models. The present paper helps to fill this gap by showing how SAT and MaxSAT solvers can be used to learn alternative interpretable models. Indeed, we provide software with formal guarantees for the investigation and construction of nested DNFs.

In recent years, a new domain of research has emerged concerning applying formal reasoning to explaining decisions of classifiers [16,38,26] with some works focusing on desirable properties of explanations [1,2] and others on the computational complexity of producing one or more explanations [7,6,12]. This is in contrast with popular and widely-used techniques, such as LIME [33], SHAP [24] and Anchors [34], which unfortunately do not provide any formal guarantees of correctness of explanations [25,27,23]. The formal reasoning approach is usually

E. Albert and C. Pasareanu (Eds.): FASE 2026, LNCS 16504, pp. 107–125, 2026.
https://doi.org/10.1007/978-3-032-22774-4_6

based on the notion of prime implicant or abductive explanation [37]. A binary boolean classifier κ is defined by its prime implicants and, for an instance x such that $\kappa(x) = 1$, any prime implicant consistent with x can be considered as an abductive explanation of the decision $\kappa(x) = 1$. This leads to the following definition of abductive explanation (also known as prime-impicant explanation or sufficient reason). We consider a classifier $\kappa : \{0,1\}^N \to \{0,1\}$ and an instance $x = (x_1, \ldots, x_N)$ composed of N boolean features.

Definition 1. *A* weak abductive explanation *(weak AXp) of the decision $\kappa(x) = c$ is a set $A \subseteq \{1, \ldots, N\}$ of features such that*

$$\forall y \in \{0,1\}^N \left(\left(\bigwedge_{i \in A} (x_i = y_i) \right) \longrightarrow \kappa(y) = c \right).$$

An abductive explanation *(AXp) is a subset-minimal weak AXp.*

Observe that the AXp's of a positive decision $\kappa(x) = 1$ are the prime implicants of κ consistent with x, whereas the AXp's of a negative decision $\kappa(x') = 0$ are the prime implicants of $\overline{\kappa}$ consistent with x'.

We now give a formal definition of interpretability, based on the size of explanations. This definition is parameterized by k, a small integer.

Definition 2. *A classifier κ is k-AXp interpretable if every decision $\kappa(x) = c$ has an AXp of size at most k.*

As an example, consider DTs of maximum depth k. They are k-AXp interpretable since each decision corresponds to a path of length $l \leq k$ and the set of l features tested along the path is a weak AXp. If a boolean function κ is given in the form of a k-DNF, then its positive decisions each have a weak AXp of size at most k corresponding to a term which evaluates to true. However, for such a k-DNF to be k-AXp interpretable, its complement $\overline{\kappa}$ must also be expressible as a k-DNF so that negative decisions also have AXp's of size at most k.

2 Nested k-DNFs and nested (k, k')-DNFs

In this section we revisit the notion of nested k-DNFs defined in [11]. This is a class of k-DNFs whose complement is also a k-DNF thus guaranteeing k-AXp interpretability. As a running example, consider the following 3-DNF with 9 terms:

$$\kappa_1 \;=\; abc + def + ghi + abd + deg + agh + abg + ade + dgh \qquad (1)$$

Remarkably, its complement also simplifies (after eliminating subsumed terms) to a 3-DNF:

$$\begin{aligned} \overline{\kappa_1} = \; & \overline{a}\,\overline{d}\,\overline{i} + \overline{a}\,\overline{d}\,\overline{g} + \overline{a}\,\overline{e}\,\overline{g} + \overline{b}\,\overline{d}\,\overline{g} + \overline{b}\,\overline{e}\,\overline{g} + \overline{c}\,\overline{d}\,\overline{g} \\ & + \overline{a}\,\overline{f}\,\overline{g} + + \overline{a}\,\overline{d}\,\overline{h} + \overline{a}\,\overline{e}\,\overline{h} + \overline{b}\,\overline{d}\,\overline{h} + \overline{b}\,\overline{e}\,\overline{h} \end{aligned}$$

This is a consequence of the fact that κ_1 is a nested 3-DNF (which we will define and explain in this section).

Consider k^2 literals $\ell_{i,j}$ $(1 \leq i, j \leq k)$. We can view $\{\ell_{i,j}\}$ as a $k \times k$ matrix:

$$\mathcal{L} = \begin{pmatrix} \ell_{1,1}\ \ell_{1,2} \ldots \ell_{1,k} \\ \vdots \\ \ell_{k,1}\ \ell_{k,2} \ldots \ell_{k,k} \end{pmatrix}$$

Using this matrix as a base, we can generate a large number of k-DNFs κ, composed of an arbitrary number n of terms, such that $\overline{\kappa}$ is also expressible as a k-DNF. For each $p = 1, \ldots, n$, let r_{pi} $(i = 1, \ldots, k)$ be k integers between 0 and k such that $\sum_{i=1}^{k} r_{pi} \leq k$. Then define κ as follows:

$$\kappa(x) \ = \ \bigvee_{p=1}^{n} \bigwedge_{i=1}^{k} \bigwedge_{j=1}^{r_{pi}} \ell_{i,j}$$

It is easy to see that the condition that $\sum_{i=1}^{k} r_{pi} \leq k$ for each $i = 1, \ldots, k$ ensures that κ is a k-DNF. Such a DNF is known as a *nested* k-DNF. For each $p = 1, \ldots, n$, the pth term of κ is the conjunction of the r_{pi} leftmost elements in row i of the matrix $\mathcal{L}$ (for $i = 1, \ldots, k$). It is this specific structure (the nested nature of the possible sets of literals from each row) that guarantees that the complement of κ is also a k-DNF [11].

We now generalize the notion of a nested k-DNF to a k-DNF whose complement is a k'-DNF, where k, k' are arbitrary and possible distinct positive integers, by considering $k' \times k$ matrices $\mathcal{L}$.

Definition 3. *A* nested (k, k')-DNF *is a k-DNF κ such that there exists a $k' \times k$ matrix of literals $\mathcal{L} = \{\ell_{i,j}\}$ and a set of integers $r_{pi} \in \{0, \ldots, k\}$ $(1 \leq p \leq n$, $1 \leq i \leq k')$ such that*

$$\kappa(x) \ = \ \bigvee_{p=1}^{n} \bigwedge_{i=1}^{k'} \bigwedge_{j=1}^{r_{pi}} \ell_{i,j}$$

In Definition 3, κ is a k-DNF, but note that we allow the possibility that $\sum_{i=1}^{k'} r_{pi}$ is greater than k: this can occur if there are repeated literals in the pth term.

Proposition 1. *The complement $\overline{\kappa}$ of a nested (k, k')-DNF is expressible as a k'-DNF.*

Proof. Clearly we have

$$\overline{\kappa}(x) \ = \ \bigwedge_{p=1}^{n} \bigvee_{i=1}^{k'} \bigvee_{j=1}^{r_{pi}} \overline{\ell_{i,j}} \tag{2}$$

We can expand this into a DNF: each term is obtained by selecting one literal from each of the n clauses in equation 2. An arbitrary term in the resulting DNF

will necessarily have the following form:

$$t = \bigwedge_{i=1}^{k'} \bigwedge_{j \in S_i} \overline{\ell_{i,j}}$$

where the S_i $(i = 1, \ldots, k')$ are subsets of $\{1, \ldots, k\}$ such that $\sum_{i=1}^{k'} |S_i| = n$. Let NE denote the set of $i \in \{1, \ldots, k'\}$ such that $S_i \neq \emptyset$. For each $i \in NE$, let $m(i)$ denote the minimum value in S_i. By construction of the terms of κ as the conjunction of the leftmost elements in rows of $\mathcal{L}$, if a clause in equation 2 contains $\overline{\ell_{i,j}}$, where $j > m(i)$, then it also contains $\overline{\ell_{i,m(i)}}$. It follows that the DNF necessarily contains another term

$$t' = \bigwedge_{i \in NE} \overline{\ell_{i,m(i)}}$$

obtained by selecting $\ell_{i,m(i)}$ instead of $\ell_{i,j}$ for all $j > m(i)$ and for all $i \in NE$. It is easy to see that t is subsumed by t' (since $m(i) \in S_i$). By elimination of all strictly subsumed terms, it follows that $\overline{\kappa}$ is equivalent to a DNF composed of terms of the form $\wedge_{i \in NE} \overline{\ell_{i,m(i)}}$, and hence $\overline{\kappa}$ can be expressed as a k'-DNF (since $NE \subseteq \{1, \ldots, k'\}$).

Returning to our running example, observe that the 3-DNF κ_1 of equation 1, is a nested (3,3)-DNF (i.e. nested 3-DNF) derived from the square matrix

$$\mathcal{L} = \begin{pmatrix} \ell_{1,1} & \ell_{1,2} & \ell_{1,3} \\ \ell_{2,1} & \ell_{2,2} & \ell_{2,3} \\ \ell_{3,1} & \ell_{3,2} & \ell_{3,3} \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

For κ_1, the number of terms $n = 9$ and, for example, $(r_{11}, r_{12}, r_{13}) = (3, 0, 0)$, meaning that the first term abc is the conjunction of the three literals in the first row of $\mathcal{L}$, whereas $(r_{91}, r_{92}, r_{93}) = (0, 1, 2)$, meaning that the ninth (and last) term dgh is the conjunction of the first literal in the second row and the first two literals in the third row of $\mathcal{L}$.

3 Verifying that a k-DNF is a nested (k, k')-DNF

We study the problem of determining whether a k-DNF given as input is a nested (k, k')-DNF. We propose a SAT encoding.

3.1 The SAT encoding

The encoding we propose is based on the definition of a nested (k, k')-DNF, with constraints that ensure the terms align with Definition 3. Specifically, we start with an initial list of terms of size at most k, and the SAT encoding is employed to infer a $k' \times k$ matrix, if it exists, from which all the terms can be generated. The resulting SAT instance is satisfiable iff such a matrix exists (i.e. the terms

Variable	Description
$X_{i,j,l}$	True if the literal $\ell_{i,j}$ is l
$R_{p,i,j}$	True if the literal $\ell_{i,j}$ is used in the term p
$T_{p,i,j,l}$	True if the literal $\ell_{i,j}$ is l and used in the term p

Table 1. Variables used in the SAT encoding.

form a nested (k, k')-DNF). Table 1 summarizes the variables introduced for the encoding.

The constraints are formulated by the following set of clauses where m and n represent the number of literals and the number of terms, respectively.

- **Each element of the matrix has at least one literal**
  For each $i \in [0, k'-1]$ and $j \in [0, k-1]$, we add the clauses:

$$\bigvee_{l=0}^{m-1} X_{i,j,l} \tag{3}$$

- **Each element of the matrix has at most one literal**
  For each $i \in [0, k'-1]$ and $j \in [0, k-1]$ and each two distinct literals a and b where $(0 \leq a, b \leq m-1)$, we add the clauses:

$$\neg X_{i,j,a} \vee \neg X_{i,j,b} \tag{4}$$

- **Nested : If a term contains $\ell_{i,j+1}$, then it also contains $\ell_{i,j}$**
  For each $i \in [0, k'-1]$, $j \in [0, k-2]$ and $p \in [0, n-1]$, we add the clauses:

$$\neg R_{p,i,j+1} \vee R_{p,i,j} \tag{5}$$

- **$T_{p,i,j,l}$ is true iff $X_{i,j,l}$ is true and $R_{p,i,j}$ is true**
  For each $i \in [0, k'-1]$, $j \in [0, k-1]$, $l \in [0, m-1]$ and $p \in [0, n-1]$, we add the clauses:

$$\begin{gathered} \neg T_{p,i,j,l} \vee X_{i,j,l} \\ \neg T_{p,i,j,l} \vee R_{p,i,j} \\ \neg X_{i,j,l} \vee \neg R_{p,i,j} \vee T_{p,i,j,l} \end{gathered} \tag{6}$$

- **Identification of the literals that are present or absent in a term**
  For each $l \in [0, m-1]$ and $p \in [0, n-1]$, we add the following clauses (where $literals(p)$ is the set of literals in the term p of the input):

$$\text{If } l \in literals(p) \ : \bigvee_{i=0}^{k'-1} \bigvee_{j=0}^{k-1} T_{p,i,j,l} \tag{7}$$

Otherwise, for each $i \in [0, k'-1]$ and $j \in [0, k-1]$: $\neg T_{p,i,j,l}$

3.2 Mechanized proof of correction

The SAT encoding is proven to be sound and complete using Why3 [10][3]. In other words, the encoding will return SAT if and only if the input k-DNF, represented by a list of terms, is a nested (k, k')-DNF.

Why3 Why3 is a platform based on the WhyML language, which combines imperative and functional aspects. It facilitates the specification and proof of programs by providing tools for specifying logical specifications (preconditions, postconditions, loop invariants, etc.) in first-order logic. It offers subgoal decomposition tactics and facilitates proof by connecting to various SMT solvers (automatic provers) as well as proof assistants such as Coq / Rocq (interactive provers). Why3 offers several libraries with lemmas and theorems that can be reused to facilitate the writing of proofs.

Why3 formalization The sources, including the Why3 formalization and the experiments, are available online : see page 16.

The SAT encoding relies on a k-DNF defined over a set of literals. To model this problem in Why3, we define a type `problem` using a record that includes a k-DNF `d`, a set `features` of features, the integer `k`, the number `n` of terms in the DNF, the number `m` of literals, and a bijection between the literals and integers (`num_to_literal`). It is notable that various constraints link these elements within the record (for instance, `m` is twice the cardinality of `features`). These constraints are defined in a predicate called `problem_valid`.

A witness proving that a k-DNF, with `n` terms, is a nested (k,k')-DNF is a pair consisting of a $k' \times k$ matrix of literals `lm` and a $n \times k'$ matrix `rpi` of integers representing the r_{pi} (of Definition 3). The type `nested_witness` represents such a witness using a record. To simplify manipulation in Why3, matrices have been encoded as sequences of sequences. A validity predicate (`nested_witness_valid (k':int) (input:problem) (nw:nested_witness)`) indicates that `lm` is a matrix of size $k' \times k$, `rpi` is a matrix of size $n \times k'$, all literals in `lm` are defined from the features of `features`, and all elements of `rpi` are positives. Finally, a predicate (`is_nested_witness (k':int) (input:problem) (nw:nested_witness)`) defines that the DNF (`input.d`) is a nested (k, k')-DNF, thanks to the witness `nw`. For `nw` to be a valid witness, each element of the matrix `rpi` must be less than `k`, and each term of the DNF `d` must be exactly defined by `lm` and `rpi` following Definition 3.

```
predicate is_nested_witness
          (k':int) (input:problem) (nw:nested_witness) =
   nested_witness_valid k' input nw
 ∧ (* All elements of the rpi matrix are at most k  *)
   (forall p: int. forall i:int.
   0 ≤ p < input.d.length → 0 ≤ i < k'
```

[3] https://www.why3.org

```
    → nw.rpi[p][i] ≤ input.k)
    (* Each term of the DNF d is exactly defined by lm and rpi *)
 ∧ (forall p: int. (0 ≤ p < input.d.length)
    → forall l:literal. (mem l input.d[p] ↔ exists i,j:int.
    0 ≤ i < k' ∧ 0 ≤j < nw.rpi[p][i] ∧ l = nw.lm[i][j]))
```

A k-DNF represented as an element of type `problem` is a nested (k, k')-DNF if there exists a witness that validates the predicate `is_nested_witness`.

```
predicate is_nested_k_kp_dnf (k':int) (pb:problem) =
   problem_valid pb
∧ exists nw:nested_witness. is_nested_witness k' pb nw
```

Similarly, the SAT encoding is described by a type `sat_witness` representing the variables of the encoding (see Table 1) : `x` a three-dimensional sequence, `r` a three-dimensional sequence and `t` a four-dimensional sequence. A predicate (`sat_witness_valid (k':int) (input:problem) (nw:nested_witness)`) signifies that `x` possesses a dimension of $k' \times k \times m$, whereas `r` has a dimension of $n \times k' \times k$ and `t` has a dimension of $n \times k' \times k \times m$. All clauses of the SAT encoding are described by predicates. For instance, the clauses described in (3) modelize that each element of the matrix has at least one literal. For the disjunctions to be true, at least one element must be true. So the clauses are represented as follows:

```
(* Each element of the matrix lm has at least one literal *)
predicate atLeastOneElement
          (k':int) (input:problem) (sw:sat_witness) =
  forall i,j : int. 0 ≤ i < k' → 0 ≤ j < input.k
  → (exists l:int. 0 ≤ l < input.m ∧ sw.x[i][j][l])
```

The clauses described in (4) are translated more directly as follows:

```
(* Each element of the matrix lm has at most one literal *)
predicate atMostOneElement
          (k':int) (input:problem) (sw:sat_witness) =
  forall i,j,a,b : int.
    0 ≤ i < k' → 0 ≤ j < input.k → 0 ≤ a < input.m
    → 0 ≤ b < input.m → a ≠ b
    → (¬ sw.x[i][j][a])  ∨  (¬ sw.x[i][j][b])
```

The same work is done for all clauses described in the section 3.1. Finally, a predicate (`is_sat_witness (k':int) (input:problem) (sw : sat_witness)`) defines that the `sw` witness is a solution of the SAT encoding of the original problem (`input`) as the conjunction of the six predicates derived from the SAT encoding.

```
predicate is_sat_witness
          (k':int) (input:problem) (sw:sat_witness) =
    sat_witness_valid k' input sw
  ∧ atLeastOneElement k' input sw
```

```
  ∧ atMostOneElement k' input sw
  ∧ nested k' input sw
  ∧ linkTXR k' input sw
  ∧ buildTMem k' input sw
  ∧ buildTNotMem k' input sw
```

The SAT encoding based on a k-DNF is satisfiable if there exists a witness that validates the predicate `is_sat_witness`.

```
predicate is_satisfiable (k':int) (pb:problem) =
    problem_valid pb
 ∧ exists sw:sat_witness. is_sat_witness k' pb sw
```

Indeed, if the initial problem is satisfiable, each of the clauses, and therefore each of the six predicates, will be satisfied.

Soundness The encoding is sound if when the SAT problem returns a solution, then the DNF is a nested (k, k')-DNF. This is expressed in Why3 as the following goal:

```
goal soundness : forall k':int. forall pb:problem.
  is_satisfiable k' pb → is_nested_k_kp_dnf k' pb
```

Since both parts of the implication are defined by the existence of a witness, the proof is based on the exhibition of a `nested_witness` from a `sat_witness`. A function `build_nested_witness` that creates a `nested_witness` from an integer `k'`, an input problem and a `sat_witness` is defined axiomatically. Two axioms define the size constraints of the `lm` and `rpi` matrices. One axiom defines the construction of the `lm` matrix. The combination of the `atLeastOneElement` and `atMostOneElement` ensures that, for all `i`,`j` there exists a unique integer `l`, such that `x[i][j][l]` is true. `lm[i][j]` is the literal associated with `l` via `num_to_literal`.

```
axiom buildLM :
 forall k':int. forall input:problem. forall sw:sat_witness.
    let nw = build_nested_witness k' input sw in
    forall i,j : int.  0 ≤ i < k' → 0 ≤ j < input.k
    → (exists l:int. 0 ≤ l < input.m ∧ sw.x[i][j][l]
        ∧ nw.lm[i][j] = (input.num_to_literal l))
```

A fourth axiom defines the construction of the `rpi` matrix from the `r` matrix. For all `p`,`i`, `rpi[p][i]` is the number of `j` such that `r[p][i][j]` is true. So `rpi[p][i]` $= \sum_{j=0}^{k-1}$ `(if r[p][i][j] then 1 else 0)`. The conditional is defined in a function `srpij` and the sum is the one defined in the module `Sum` of the integers (predefined in Why3).

```
use int.Sum as SI

function srpij (sw:sat_witness) : int → int → int → int=
```

```
  fun p → fun i → fun j → if sw.r[p][i][j] then 1 else 0

axiom buildRPI :
 forall k':int. forall input:problem. forall sw:sat_witness.
    let nw = build_nested_witness k' input sw in
    forall p,i: int. 0 ≤ p < input.n → 0 ≤ i < k'
    → nw.rpi[p][i] = SI.sum (srpij sw p i) 0 input.k
```

The proof of soundness therefore consists of proving that the witness constructed by the function `build_nested_witness` is indeed a nested witness.

```
lemma build_nested_witness_is_nested_witness :
  forall k':int. forall pb:problem. forall sw:sat_witness.
  problem_valid pb
  → is_sat_witness k' pb sw
  → is_nested_witness k' pb (build_nested_witness k' pb sw)
```

Unsurprisingly, the fact that all literals of the leftmost part (defined by `rpi`) of the matrix `lm` are in the term of the DNF `d` required the most lemmas. This proof is based on a lemma that is proven on the SAT encoding, namely that if the predicate associated with clauses (5) is true, then all lines of `rpi[p][i]` consist of 1 followed by 0. Thus, if `j < rpi[p][i]`, then `r[p][i][j]` is true, by definition of `rpi`. The existence of an `l` such that `x[i][j][l]` is true (clauses 3), leads to the existence of a `t[p][i][j][l]` and so to a literal of the DNF.

Completeness The encoding is complete if when the DNF is a nested (k, k')-DNF then the SAT problem returns a solution. This is expressed in Why3 as the following goal:

```
goal completness : forall k':int. forall pb:problem.
  is_nested_k_kp_dnf k' pb → is_satisfiable k' pb
```

Similarly to soundness, the proof is based on the exhibition of a `sat_witness` from a `nested_witness`. A function `build_satisfiable_witness` that creates a `sat_witness` from an integer `k'`, an input problem and a `nested_witness` is defined axiomatically. One axiom defines the size constraints of the `x`, `r` and `t` matrices. One axiom defines the construction of the `x` matrix from the `lm` matrix, i.e. `x[i][j][l]` is true iff `lm[i][j]` contains the literal associated to `l` (remember that a conversion literal to integer is needed).

```
axiom buildX :
 forall k':int. forall input:problem. forall nw:nested_witness.
    let sw = build_satisfiable_witness k' input nw in
    forall i,j,l : int.
    0 ≤ i < k' → 0 ≤ j < input.k → 0 ≤ l < input.m
    → sw.x[i][j][l]↔ nw.lm[i][j] = input.num_to_literal l
```

One axiom defines the construction of the `r` matrix from the `rpi` matrix, i.e. `r[p][i][j]` is true iff `j < rpi[p][i]`.

```
axiom buildR :
 forall k':int. forall input:problem. forall nw:nested_witness.
    let sw = build_satisfiable_witness k' input nw in
    forall i,j,p : int.
    0 ≤ i < k' → 0 ≤ j < input.k →  0 ≤ p < input.n
    → sw.r[p][i][j] ↔ j < nw.rpi[p][i]
```

One axiom defines the construction of the `t` matrix from the `x` and `r` matrices, i.e. `t[p][i][j][l]` is true iff `x[i][j][l]` $\wedge$ `r[p][i][j]`.

```
axiom buildT :
 forall k':int. forall input:problem. forall nw:nested_witness.
    let sw = build_satisfiable_witness k' input nw in
    forall i,j,p,l:int.
       0 ≤ i < k' → 0 ≤ j < input.k
       → 0 ≤ p < input.n → 0 ≤ l < input.m
       → sw.t[p][i][j][l] ↔ sw.x[i][j][l] ∧ sw.r[p][i][j]
```

The proof of completeness therefore consists of proving that the witness constructed by the function `build_satisfiable_witness` is indeed a sat witness. The completeness is then ensured by the following lemma :

```
lemma build_satisfiable_witness_is_sat_witness :
  forall k':int. forall pb:problem. forall nw:nested_witness.
  problem_valid pb
  → is_nested_witness k' pb nw
  → is_sat_witness k' pb (build_satisfiable_witness k' pb nw)
```

The proof is almost entirely done automatically by SMT solvers. The two lemmas that required manual proof were those that required manipulation of the bijection between literals and integers.

Proof effort All lemmas and goals are proven using Why3 (version 1.8.1) coupled with Z3[4] (version 4.15.2) [29], Alt-Ergo[5] (version 2.6.2), CVC5[6] (version 1.1.2) and Coq / Rocq[7] (version 8.20.1) [8]. The proof of the two goals requires 43 lemmas, of which 25 are proven automatically using SMT solvers, while 18 are proven manually with Coq / Rocq.

4 Learning nested (k, k')-DNFs

From a practical standpoint, it is interesting to construct a nested (k, k')-DNF starting with a given dataset. Therefore, we propose a MaxSAT encoding grounded on the previous SAT encoding, with the aim of satisfying the established constraints while maximizing accuracy.

[4] https://github.com/Z3Prover/z3
[5] https://alt-ergo.ocamlpro.com/
[6] https://cvc5.github.io/
[7] https://rocq-prover.org/

4.1 The MaxSAT encoding

In the previous section, the starting point was a k-DNF composed of a list of terms, with the task being to verify whether it is a nested (k, k')-DNF by identifying a generating matrix for all terms. We now study an approach that takes as input a dataset alongside candidate terms of size at most k, from which a subset of terms will be retained forming a nested (k, k')-DNF with the intention of maximizing accuracy. Hence, the goal is to select terms that maximize the coverage of target class examples in the dataset, while minimizing the coverage of the examples labelled by the other class. Therefore, in addition to the variables presented in Table 1, we introduce the two variables defined in Table 2.

Variable	Description
M_p	True if the term p is taken in the DNF
E_i	True if the example e_i is covered by the DNF

Table 2. Variables used in the MaxSAT encoding.

The MaxSAT encoding we propose consists of hard clauses that must be satisfied while aiming to maximize the satisfaction of soft clauses. As hard clauses, clauses (3), (4), (5) and (6) remains unchanged, whereas the literal $\neg M_p$ is added to clauses (7), indicating the condition that the corresponding term is taken in the final DNF. Moreover, in order to determine which terms cover the dataset examples, we add, for each example e_i, the following clauses, which come from $E_i \leftrightarrow \bigvee_{p \in P_i} M_p$, where P_i denotes the set of terms that cover e_i. In other words, if the example e_i is covered by the DNF, then at least one of the terms that covers it (i.e., one of the terms of P_i) is taken in the DNF. Furthermore, if a term $p \in P_i$ is taken in the DNF, then the example e_i is covered by the DNF. Finally, if P_i is empty, no term covers the example e_i, so it cannot be covered by the DNF.

$$\begin{gathered} \text{If } P_i \text{ is not empty :} \\ \neg E_i \vee \bigvee_{p \in P_i} M_p \\ \text{for each } p \in P_i \text{ , } \neg M_p \vee E_i \\ \text{Otherwise : } \neg E_i \end{gathered} \tag{8}$$

As soft clauses, given that the goal is to maximize accuracy, the clauses are defined as follows, for each example e_i in the dataset, with $\mathcal{E}_c$ referring to the set of examples of the target class c intended to be represented by the DNF.

$$\begin{gathered} \text{If } e_i \in \mathcal{E}_c \text{ : } E_i \\ \text{Otherwise : } \neg E_i \end{gathered} \tag{9}$$

The output of the MaxSAT is a nested (k, k')-DNF The proof that the SAT solver output generates a nested (k, k')-DNF is based on the creation of witnesses from the variables X, R, and T of the encoding. To show that the MaxSAT encoding also generates a nested (k, k')-DNF, it suffices to look at the clauses that manipulate these variables and that have been modified, namely clauses (7). The addition of $\neg M_p$ is logically equivalent to $M_p \rightarrow$ the SAT formula (7).

Let CT be the set of candidate terms used as input to the MaxSAT solver. When MaxSAT returns an answer, this set can be partitioned as follows: $CT = \{p \in CT | M_p\} \cup \{p \in CT | \neg M_p\}$ that is, the terms taken in the DNF and the terms not taken in the DNF. Let $n = |\{p \in CT | M_p\}|$ and $f : \{p \in CT | M_p\} \rightarrow \{0, .., n-1\}$ be a bijection that defines a reindexing of the terms included in the DNF such that the new indices $q < n$. The clauses in the SAT encoding, when these terms are given as inputs, are satisfied by X, R' and T' such that:

- R' is a $n \times k' \times k$ matrix such that $R'[q][i][j] = R[f^{-1}(q)][i][j]$
- T' is a $n \times k' \times k \times m$ matrix such that $T'[q][i][j][l] = T[f^{-1}(q)][i][j][l]$

Indeed, for all $q < n$, $M_{f^{-1}(q)}$ is true, so each clause (7) in the SAT encoding is satisfied since it is the right-hand side of an implication whose left-hand side is true. The DNF constructed from X, R' and T' is a nested (k, k')-DNF (Proof 3.2). This same DNF is obtained from X, R and T. Therefore, the DNF extracted from the MaxSAT answer is a nested (k, k')-DNF.

4.2 Description of the Experiments

In order to guarantee k-AXp interpretability (see Definition 2) of learned models, we investigated the construction of a nested (k, k)-DNF from a dataset.

The number of possible terms for a maximum term size k and a given number of literals m is $\sum_{i=1}^{k} \binom{m}{i}$, which includes all combinations of literals with cardinality up to k. Even when the search space is restricted to consistent terms (i.e. those that do not contain both a literal and its negation) and further filtering to retain only the terms that cover more examples of the target class (than of the other class), the number of candidate terms remains large, causing a notable increase in the overall number of clauses in the MaxSAT encoding. In order to address this, we suggest beginning by considering decision trees paths as potential terms. Specifically, we construct a random forest (i.e. a collection of DTs) and take the paths that lead to the target class as terms (see Figure 1).

We used the Random Forest Classifier with default parameters from the sklearn package in Python to generate the trees. A random forest with $10 \times k$ trees of maximum depth k is considered for every value of k. We report the accuracy of the RFs and the accuracy of the nested (k, k)-DNFs found by the MaxSAT solver, where the paths of the trees of depth from 2 to k are the candidate terms taken into consideration. We used the incomplete MaxSAT solver NuWLS-c-IBR from the MaxSAT Evaluation 2024[8] with a timeout. Note that

[8] https://maxsat-evaluations.github.io/2024/descriptions.html

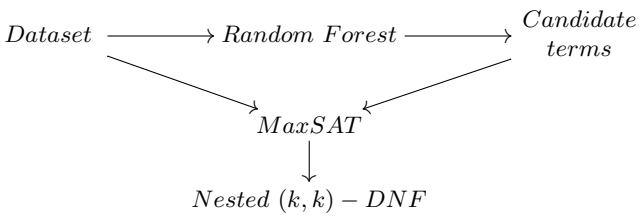

Fig. 1. General Architecture of the software

we searched for the nested (k, k)-DNFs for both classes, since the expressibility of a function as a nested (k, k)-DNF is not generally preserved under complementation (the complement of a nested (k, k)-DNF is guaranteed to be a k-DNF but not necessarily a nested (k, k)-DNF).

4.3 Datasets

We selected a collection of datasets from the UCI Repository[9] that contain categorical features in order to avoid the possibility of having a large number of literals, which would be the case if continuous features were used. All datasets consist of two classes, except for Balance Scale and Car Evaluation, which originally had 3 and 4 classes, respectively. To convert them to binary classification, the majority class was retained as one class, while the remaining classes were merged into a single second class. Additionally, label encoding was applied to convert some features into numeric values. An overview of these datasets is given in Table 3, with the last column indicating the average number of distinct values per feature. For each dataset, 80% of the examples was used for training while the rest was used for testing, except for the Monks and SPECT Heart datasets, where the train and test sets are provided separately.

Dataset	**# Examples**	**# Features**	**Avg. Cardinality**
Balance Scale	625	4	5.0
Car Evaluation	1728	6	3.5
Chess kr vs. kp	3196	36	2.0
Monks-1	432	6	2.8
Monks-2	432	6	2.8
Monks-3	432	6	2.8
SPECT Heart	267	22	2.0
Tic-Tac-Toe	958	9	3.0

Table 3. Characteristics of the datasets used in the experiments.

[9] https://archive.ics.uci.edu/datasets

4.4 Results

Our experimental results, presented in Table 4, indicate that the MaxSAT solver effectively finds nested (k, k)-DNFs with high accuracy, outperforming random forests (which do not have a guarantee of k-AXp interpretability) in certain datasets. For instance, in the Monks datasets, which consist of three artificial domains with specific structures, the nested (k, k)-DNFs formulas accurately represents the data. Note that the 5% noise in the Monks-3 training set causes overfitting as k increases beyond 3, although the nested $(3, 3)$-DNF model achieves 100% test accuracy. However, due to the restriction of only being able to contain up to k^2 distinct literals, the datasets used were relatively small to medium in size, particularly in terms of the number of features. In such cases, when the number of literals is not large, as in boolean domains, nested (k, k)-DNFs demonstrated good performance due to their ability to express a great variety of dependencies between literals. In contrast, the increasing number of candidate terms to be considered can become significant as the number of literals increases, especially when combined with a large value of k, which may cause the nested (k, k)-DNFs to underperform. More specifically, this results in an excessive increase in the number of clauses in the MaxSAT, making it difficult to find a model. Mitigating this issue demands reducing the number of clauses, though doing so requires considering fewer candidate terms, which may limit the DNF's ability to accurately represent the dataset.

From another viewpoint, the approach used in the experiments can be seen as extracting the knowledge from the trees into a smaller model, a nested (k, k)-DNF, thereby enhancing the model's interpretability. Notably, we observed that the nested (k, k)-DNFs for $k = 7$ do not exceed 20 terms in size. With regard to runtime, the solver successfully finds a model within a 2 minute timeout across all cases, except for a single representation of one class of the Chess dataset for $k = 7$, which requires increasing the timeout. Hence, we extended the timeout to 3 minutes for all datasets for $k = 7$, providing the solver more time to potentially find better models due to the larger size of the formulas.

5 State of the art

In [18], the authors provide an overview of recent progress in reasoning and constraint-based approaches to learning interpretable machine learning models, and discuss their advantages and limitations. They mainly consider as interpretable: decision trees, decision sets and decision lists. Depth-k decisions trees are known to be k-AXp interpretable and have been compared experimentally with nested k-DNFs [11]. However, unlike nested k-DNFs and depth-k decision trees, decision sets and decision lists do not provide guarantees on the size of explanations of default decisions, and hence cannot be considered k-AXp interpretable.

Definition 3 provides a generalisation of nested k-DNFs to nested (k, k')-DNFs. In the context of learning interpretable models, as in our experiments, it makes sense to choose $k = k'$, but there are potential applications in which

Dataset	Test accuracy (%)								
	$k=2$			$k=3$			$k=4$		
	RF	DNF_1	DNF_0	RF	DNF_1	DNF_0	RF	DNF_1	DNF_0
Balance Scale	**85.60**	72.80	72.00	**93.60**	83.20	83.20	**93.60**	87.20	82.40
Car Evaluation	78.61	**87.86**	**87.86**	**95.38**	94.51	94.51	98.27	**98.55**	96.24
Chess kr vs. kp	**90.00**	80.78	84.38	**93.12**	**93.12**	**93.12**	93.12	**94.38**	94.22
Monks-1	**75.46**	75.00	75.00	86.11	83.33	**91.67**	89.81	91.67	**100.0**
Monks-2	**67.13**	60.19	61.81	68.29	64.12	**71.76**	70.14	67.36	**70.83**
Monks-3	96.99	**97.22**	**97.22**	97.22	94.44	**100.0**	97.22	95.37	**99.07**
SPECT Heart	70.05	68.45	**71.12**	73.26	73.26	**73.80**	**73.80**	63.10	57.75
Tic-Tac-Toe	65.62	67.71	**68.75**	72.92	75.00	**80.73**	79.69	**84.38**	82.29

Dataset	$k=5$			$k=6$			$k=7$		
	RF	DNF_1	DNF_0	RF	DNF_1	DNF_0	RF	DNF_1	DNF_0
Balance Scale	**89.60**	83.20	84.80	**92.80**	80.00	86.40	**91.20**	77.60	75.20
Car Evaluation	97.98	98.27	**98.84**	**98.84**	**98.84**	98.27	**99.13**	**99.13**	**99.13**
Chess kr vs. kp	93.12	**94.53**	93.44	93.12	**95.47**	94.06	93.12	93.59	**94.22**
Monks-1	89.35	**100.0**	**100.0**	90.51	**100.0**	**100.0**	91.67	**100.0**	**100.0**
Monks-2	72.92	70.37	**75.00**	77.78	**90.05**	77.55	80.09	**83.80**	65.05
Monks-3	**97.45**	95.14	94.44	**97.22**	88.89	91.90	**95.60**	93.06	90.74
SPECT Heart	**74.33**	69.52	62.57	**74.33**	71.12	66.31	**75.94**	67.38	59.89
Tic-Tac-Toe	85.42	**90.10**	85.42	86.46	74.48	**86.98**	88.02	**92.19**	69.27

Table 4. Test accuracy of depth-k Random Forests and Nested (k,k)-DNFs. Columns RF, DNF_1 and DNF_0 denote Random Forest, DNF representation of class 1 and DNF representation of class 0, respectively.

$k \neq k'$. For example, if a given model κ is determined to be a nested (k,k')-DNF, where $k' < k$, then we may prefer to present its complement $\overline{\kappa}$ (which is a k'-DNF) to the user. A minor difference between nested k-DNFs and nested (k,k)-DNFs is that the latter imposes only that the number of *distinct* literals taken from the matrix $\mathcal{L}$ to build a term be at most k.

The use of (Max)SAT encodings for model learning has been explored in a variety of studies. Notably, several works have proposed SAT-based approaches for generating decision trees [9,30,21,4], as well as methods based on MaxSAT [15]. In [35], they combine the scalability of heuristic methods with the strength of encoding-based exact methods. A parallel can be drawn with our method: we use heuristics to identify candidate terms using a Random Forest, followed by MaxSAT which provides guarantees on the generated model. A (Max)SAT formulation has also been used in learning oblique decision trees (i.e. trees with linear combinations of features at each node) [5]. Furthermore, a (Max)SAT encoding has been introduced for training other models, namely decision sets and decision lists, as shown in [19,40,17]. For instance, [40] construct decision sets

and decision lists that achieve complete accuracy on the training data while being minimal in size, leveraging modern SAT-solving techniques to do so. Lastly, a novel MaxSAT encoding for learning Binary Decision Diagrams has been proposed in [36]. Clearly, models of different types impose structural requirements that must be met. In our work as well, we construct DNF formulas that satisfy a specific structural property. We go further by formally proving that this property is correctly enforced by the (Max)SAT encoding.

DNF formulas offer a simple and intuitive representation of boolean functions. Since their introduction, learning these models has been extensively studied [39,31,22,14]. Moreover, DNF simplification has been addressed in prior works, including the classical techniques introduced in [32,28]. Unlike studies that focus only on maximizing accuracy, our approach prioritizes maintaining the desired structure (i.e. nested DNF), even if it comes at the cost of some performance degradation. Doing so enables a gain in interpretability, given the guarantee that the size of the terms in the complement are bounded by a fixed constant.

6 Conclusion

Confidence in decisions taken by a machine-learning model requires explanations which are comprehensible by humans. We have investigated a recently-discovered class of interpretable models called nested DNFs. In order to ensure formal correctness of our experimental investigation of nested DNFs, we formally verified an encoding of the problem of deciding whether a DNF is expresssible as a nested DNF. This allowed us to use produce reliable software for learning a nested DNF from a dataset.

The subsequent experimental comparison of the learnt nested DNFs with random forests allowed us to confirm the viability of learning interpretable models whose accuracy is comparable with models which do not have a guarantee of interpretability. One avenue of possible future research is to develop heuristics for larger datasets and models. Another is the search for other interpretable models such as hybrid models combining decision trees and nested DNFs.

Data availability

The sources, including the Why3 formalization and the experiments, are available through DOI https://doi.org/10.5281/zenodo.18330397. Any modifications made after the publication of the paper will be visible at https://github.com/hurault/FASE26.

Acknowledgement

This work was funded by the French National Research Agency project ForML ANR-23-CE25-0009.

References

1. Amgoud, L., Ben-Naim, J.: Axiomatic foundations of explainability. In: Raedt, L.D. (ed.) IJCAI. pp. 636–642. ijcai.org (2022). https://doi.org/10.24963/ijcai.2022/90
2. Amgoud, L., Cooper, M.C., Debbaoui, S.: Axiomatic characterisations of sample-based explainers. In: ECAI. Frontiers in Artificial Intelligence and Applications, vol. 392, pp. 770–777. IOS Press (2024). https://doi.org/10.3233/FAIA240561
3. Audemard, G., Lagniez, J., Marquis, P., Szczepanski, N.: Deriving provably correct explanations for decision trees: The impact of domain theories. In: IJCAI. pp. 3688–3696. ijcai.org (2024), https://www.ijcai.org/proceedings/2024/408
4. Avellaneda, F.: Efficient inference of optimal decision trees. In: AAAI. pp. 3195–3202. AAAI Press (2020). https://doi.org/10.1609/AAAI.V34I04.5717
5. Avellaneda, F.: Learning optimal oblique decision trees with (max)sat. In: IJCAI 2025. pp. 2558–2565. ijcai.org (2025). https://doi.org/10.24963/IJCAI.2025/285
6. Barceló, P., Monet, M., Pérez, J., Subercaseaux, B.: Model interpretability through the lens of computational complexity. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) NeurIPS (2020), https://proceedings.neurips.cc/paper/2020/hash/b1adda14824f50ef24ff1c05bb66faf3-Abstract.html
7. Bassan, S., Amir, G., Katz, G.: Local vs. global interpretability: A computational complexity perspective. In: ICML. Proceedings of Machine Learning Research, vol. 235, pp. 3133–3167 (2024), https://openreview.net/forum?id=veEjiN2w9F
8. Bertot, Y., Castéran, P.: Interactive Theorem Proving and Program Development - Coq'Art: The Calculus of Inductive Constructions. Texts in Theoretical Computer Science. An EATCS Series, Springer (2004). https://doi.org/10.1007/978-3-662-07964-5
9. Bessiere, C., Hebrard, E., O'Sullivan, B.: Minimising decision tree size as combinatorial optimisation. In: Gent, I.P. (ed.) CP 2009. LNCS, vol. 5732, pp. 173–187. Springer (2009). https://doi.org/10.1007/978-3-642-04244-7_16
10. Bobot, F., Filliâtre, J.C., Marché, C., Paskevich, A.: Why3: Shepherd your herd of provers. In: Boogie 2011: First International Workshop on Intermediate Verification Languages. pp. 53–64 (2011)
11. Cooper, M.C., Bousdira, I., Carbonnel, C.: Interpretable DNFs. In: IJCAI. pp. 4985–4993 (2025). https://doi.org/10.24963/IJCAI.2025/555
12. Cooper, M.C., Marques-Silva, J.: Tractability of explaining classifier decisions. Artif. Intell. **316** (2023). https://doi.org/10.1016/J.ARTINT.2022.103841
13. Demirovic, E., Hebrard, E., Jean, L.: Blossom: an anytime algorithm for computing optimal decision trees. In: Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., Scarlett, J. (eds.) ICML. vol. 202, pp. 7533–7562. PMLR (2023)
14. Feldman, V.: Hardness of approximate two-level logic minimization and PAC learning with membership queries. J. Comput. Syst. Sci. **75**(1), 13–26 (2009). https://doi.org/10.1016/J.JCSS.2008.07.007
15. Hu, H., Siala, M., Hebrard, E., Huguet, M.: Learning optimal decision trees with MaxSAT and its integration in AdaBoost. In: Bessiere, C. (ed.) IJCAI. pp. 1170–1176 (2020). https://doi.org/10.24963/IJCAI.2020/163
16. Hurault, A., Marques-Silva, J.: Certified logic-based explainable AI - the case of monotonic classifiers. In: Prevosto, V., Seceleanu, C. (eds.) Tests and Proofs - 17th International Conference, TAP 2023. LNCS, vol. 14066, pp. 51–67. Springer (2023). https://doi.org/10.1007/978-3-031-38828-6_4
17. Ignatiev, A., Lam, E., Stuckey, P.J., Marques-Silva, J.: A scalable two stage approach to computing optimal decision sets. In: AAAI. pp. 3806–3814. AAAI Press (2021). https://doi.org/10.1609/AAAI.V35I5.16498

18. Ignatiev, A., Marques-Silva, J., Narodytska, N., Stuckey, P.J.: Reasoning-based learning of interpretable ML models. In: Zhou, Z. (ed.) IJCAI. pp. 4458–4465. ijcai.org (2021). https://doi.org/10.24963/IJCAI.2021/608
19. Ignatiev, A., Pereira, F., Narodytska, N., Marques-Silva, J.: A sat-based approach to learn explainable decision sets. In: Galmiche, D., Schulz, S., Sebastiani, R. (eds.) IJCAR 2018. LNCS, vol. 10900, pp. 627–645. Springer (2018). https://doi.org/10.1007/978-3-319-94205-6_41
20. Izza, Y., Ignatiev, A., Marques-Silva, J.: On tackling explanation redundancy in decision trees. J. Artif. Intell. Res. **75**, 261–321 (2022)
21. Janota, M., Morgado, A.: SAT-based encodings for optimal decision trees with explicit paths. In: Pulina, L., Seidl, M. (eds.) SAT 2020. LNCS, vol. 12178, pp. 501–518. Springer (2020). https://doi.org/10.1007/978-3-030-51825-7_35
22. Klivans, A.R., Servedio, R.A.: Learning DNF in time $2^{\tilde{O}(n^{1/3})}$. J. Comput. Syst. Sci. **68**(2), 303–318 (2004). https://doi.org/10.1016/J.JCSS.2003.07.007
23. Létoffé, O., Huang, X., Marques-Silva, J.: Towards trustable SHAP scores. In: Walsh, T., Shah, J., Kolter, Z. (eds.) AAAI-25. pp. 18198–18208. AAAI Press (2025). https://doi.org/10.1609/AAAI.V39I17.34002
24. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. Curran Associates, Inc. (2017), http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf
25. Marques-Silva, J.: Disproving XAI myths with formal methods - initial results. In: Aït-Ameur, Y., Khendek, F., Méry, D. (eds.) ICECCS 2023. pp. 12–21. IEEE (2023). https://doi.org/10.1109/ICECCS59891.2023.00012
26. Marques-Silva, J.: Logic-based explainability: Past, present & future. CoRR **abs/2406.11873** (2024), https://doi.org/10.48550/arXiv.2406.11873
27. Marques-Silva, J., Huang, X.: Explainability is *Not* a game. Commun. ACM **67**(7), 66–75 (2024). https://doi.org/10.1145/3635301
28. McCluskey, E.J.: Minimization of boolean functions. The Bell System Technical Journal **35**(6), 1417–1444 (1956). https://doi.org/10.1002/j.1538-7305.1956.tb03835.x
29. de Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 337–340. Springer (2008)
30. Narodytska, N., Ignatiev, A., Pereira, F., Marques-Silva, J.: Learning optimal decision trees with SAT. In: Lang, J. (ed.) IJCAI 2018. pp. 1362–1368. ijcai.org (2018). https://doi.org/10.24963/IJCAI.2018/189
31. Pitt, L., Valiant, L.G.: Computational limitations on learning from examples. J. ACM **35**(4), 965–984 (1988). https://doi.org/10.1145/48014.63140
32. Quine, W.V.: The problem of simplifying truth functions. The American Mathematical Monthly **59**(8), 521–531 (1952), http://www.jstor.org/stable/2308219
33. Ribeiro, M.T., Singh, S., Guestrin, C.: "Why should I trust you?": Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1135–1144 (2016)
34. Ribeiro, M.T., Singh, S., Guestrin, C.: Anchors: High-precision model-agnostic explanations. In: McIlraith, S.A., Weinberger, K.Q. (eds.) AAAI. pp. 1527–1535. AAAI Press (2018). https://doi.org/10.1609/AAAI.V32I1.11491
35. Schidler, A., Szeider, S.: SAT-based decision tree learning for large data sets. J. Artif. Intell. Res. **80**, 875–918 (2024). https://doi.org/10.1613/JAIR.1.15956

36. Shati, P., Cohen, E., McIlraith, S.A.: Sat-based learning of compact binary decision diagrams for classification. In: Yap, R.H.C. (ed.) CP 2023. LIPIcs, vol. 280, pp. 33:1–33:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2023). https://doi.org/10.4230/LIPICS.CP.2023.33
37. Shih, A., Choi, A., Darwiche, A.: A symbolic approach to explaining bayesian network classifiers. In: IJCAI. pp. 5103–5111 (2018)
38. Törnblom, J., Nadjm-Tehrani, S.: Formal verification of input-output mappings of tree ensembles. Sci. Comput. Program. **194**, 102450 (2020). https://doi.org/10.1016/J.SCICO.2020.102450
39. Valiant, L.G.: A theory of the learnable. Communications of the ACM **27**, 1134–1142 (1984)
40. Yu, J., Ignatiev, A., Stuckey, P.J., Bodic, P.L.: Learning optimal decision sets and lists with SAT. J. Artif. Intell. Res. **72**, 1251–1279 (2021). https://doi.org/10.1613/JAIR.1.12719