Solving Explainability Queries with Quantification: The Case of Feature Relevancy

Xuanxiang Huang[1], **Yacine Izza**[1, 3], **Joao Marques-Silva**[2]
[1] IRIT, University of Toulouse, France
[2] IRIT, CNRS, Toulouse, France
[3] CREATE, National University of Singapore, Singapore
xuanxiang.huang@univ-toulouse.fr, izza@com.nus.edu.sg, joao.marques-silva@irit.fr

Abstract

Trustable explanations of machine learning (ML) models are vital in high-risk uses of artificial intelligence (AI). Apart from the computation of trustable explanations, a number of explainability queries have been identified and studied in recent work. Some of these queries involve solving quantification problems, either in propositional or in more expressive logics. This paper investigates one of these quantification problems, namely the feature relevancy problem (FRP), i.e. to decide whether a (possibly sensitive) feature can occur in some explanation of a prediction. In contrast with earlier work, that studied FRP for specific classifiers, this paper proposes a novel algorithm for the FRP quantification problem which is applicable to any ML classifier that meets minor requirements. Furthermore, the paper shows that the novel algorithm is efficient in practice. The experimental results, obtained using random forests (RFs) induced from well-known publicly available datasets, demonstrate that the proposed solution outperforms existing state-of-the-art solvers for Quantified Boolean Formulas (QBF) by orders of magnitude. Finally, the paper also identifies a novel family of formulas that are challenging for currently state-of-the-art QBF solvers.

Introduction

The advances in ML over the years, and the fact that ML models are most often opaque, sparked the ongoing efforts on explainable artificial intelligence (XAI). Furthermore, the existing and expected uses of ML in high-risk applications of AI (European Commission 2021) motivate the need for explainability approaches that offer guarantees of rigor, and so can be trusted. Such need is underscored by the ample evidence of bias in ML models (Makortoff 2022). Unfortunately, the most visible XAI approaches (Ribeiro, Singh, and Guestrin 2016; Lundberg and Lee 2017; Ribeiro, Singh, and Guestrin 2018) offer no guarantees of rigor. For example, existing results have shown that such informal explanations can be consistent with points in feature space for which the prediction differs (Ignatiev, Narodytska, and Marques-Silva 2019c; Narodytska et al. 2019; Ignatiev 2020).

Pioneered by work on explaining boolean classifiers represented with restricted families of bayesian networks (Shih, Choi, and Darwiche 2018), there have been a stream of results on formal explainability, which are summarized in recent overviews include (Marques-Silva and Ignatiev 2022; Marques-Silva 2022a,b)[1] In addition to the problem of computing one explanation, recent work also studied a number of queries (Audemard, Koriche, and Marquis 2020; Huang et al. 2021; Audemard et al. 2021; Huang et al. 2022a), which can be addressed in the context of formal explainability, and which find numerous applications.

One example of an explainability query is *feature membership* (Huang et al. 2021), which corresponds to the problem of *relevancy* in logic-based abduction (Friedrich, Gottlob, and Nejdl 1990; Selman and Levesque 1990; Eiter and Gottlob 1995). (Aiming for naming consistency, this paper refers to the problem of feature membership as the *feature relevancy problem*.) Given a point $\mathbf{v}$ in feature space and its associated prediction c, the feature relevancy problem (FRP) is to decide whether a given target feature t can occur in some explanation of why the prediction is c given $\mathbf{v}$. For example, and motivated by existing regulations and guidelines (e.g. (European Commission 2016; European Commission's High-Level Expert Group on AI 2019; European Commission 2021)), t can be a sensitive feature, e.g. age, gender, ethnic origin, etc., and the existence of an explanation that includes t would represent a violation of such regulations. Earlier work proved Σ_2^{P}-hardness of FRP in the case of a

[1] Additional references include (Shih, Choi, and Darwiche 2018; Ignatiev, Narodytska, and Marques-Silva 2019a; Shih, Choi, and Darwiche 2019; Ignatiev, Narodytska, and Marques-Silva 2019b,c; Narodytska et al. 2019; Wolf, Galanti, and Hazan 2019; Darwiche 2020; Ignatiev 2020; Darwiche and Hirth 2020; Ignatiev et al. 2020a,b; Audemard, Koriche, and Marquis 2020; Izza, Ignatiev, and Marques-Silva 2020; Marques-Silva et al. 2020; Barceló et al. 2020; Marques-Silva et al. 2021; Izza and Marques-Silva 2021; Malfa et al. 2021; Ignatiev and Marques-Silva 2021; Cooper and Marques-Silva 2021; Huang et al. 2021; Audemard et al. 2021; Arenas et al. 2021; Blanc, Lange, and Tan 2021; Amgoud 2021; Wäldchen et al. 2021; Darwiche and Marquis 2021; Izza et al. 2021; Marques-Silva and Ignatiev 2022; Huang et al. 2022b; Ignatiev et al. 2022; Gorji and Rubin 2022; Darwiche and Ji 2022; Audemard et al. 2022b; Amgoud and Ben-Naim 2022; Audemard et al. 2022a; Ferreira et al. 2022; Liu and Lorini 2022; Izza, Ignatiev, and Marques-Silva 2022; Izza et al. 2022; Yu et al. 2022; Huang and Marques-Silva 2022; Izza and Marques-Silva 2022; Arenas et al. 2022).

DNF (disjunctive normal form) classifier.

Whereas earlier work considered specific families of classifiers (Huang et al. 2021; Huang and Marques-Silva 2022), e.g. decision trees, classifiers represented by DNF formulas or other propositional languages, this paper investigates instead generic FRP algorithms, which are independent of specific families of classifiers. Concretely, the paper shows that, for several families of classifiers, FRP is complete for Σ_2^{P}. Furthermore, the paper investigates solutions for solving FRP for those and other families of classifiers. Concretely, the paper develops quantified boolean formula (QBF) encodings for deciding FRP. Furthermore, the paper also proposes a novel algorithm for deciding FRP, which is based on counterexample-guided abstraction refinement (CEGAR) (Clarke et al. 2003). The proposed algorithm (FR-PCGR) can be used with any classifier that admits a logic representation, and so this includes most of the families of ML classifiers in common use. The paper focuses on random forests, simply because these can be encoded using QBF formulas, and so enable a direct comparison of the proposed CEGAR algorithm with QBF reasoners.

The experimental results demonstrate that the novel CEGAR-based algorithm substantially outperforms state-of-the-art QBF solvers, enabling to decide FRP for much larger random forests than what QBF solvers can handle. Thus, an indirect by-product of this work is a new family of challenging problem instances for benchmarking QBF solvers.

Preliminaries

Complexity classes, propositional logic & quantification. The paper assumes basic knowledge of computational complexity, namely the classes NP and Σ_2^{P}. The paper also assumes basic knowledge of propositional logic, including the Boolean satisfiability (SAT) problem for formulas in conjunctive normal form (CNF), the decision problem for quantified boolean formulas (QBF), and the use of SAT solvers as oracles for the complexity class NP. The interested reader is referred to existing bibliography on these topics (Arora and Barak 2009; Biere et al. 2021).

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

The *classifier decision problem* (CDP) is to decide whether the logic statement $\exists(\mathbf{x} \in \mathbb{F}).(\kappa(\mathbf{x}) = c)$, for $c \in \mathcal{K}$, is true. Given some target class $c \in \mathcal{K}$, the goal of CDP is to decide whether there exists some point $\mathbf{x}$ in feature space for which the prediction is c. For example, for a (plain) neural network or a random forest, it is easy to prove that CDP is NP-complete. However, for a linear classifier or a decision tree, CDP is in P.

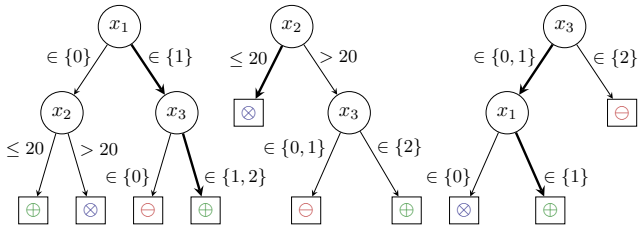

Figure 1: Random Forest Running Example.

Random Forests (RFs). Random Forests (RFs) (Breiman 2001; Yang et al. 2020; Zhang et al. 2019; Gao and Zhou 2020; Feng and Zhou 2018; Zhou and Feng 2017) are very popular and widely used tree ensemble ML models. Conceptually, an RF is collection of decision trees (DTs), where each tree $\mathcal{T}_i$, $i \in \{1, \ldots, T\}$ of the ensemble $\mathcal{T}$ is trained on a randomly selected subset of the training data so as the trees of the RF are not correlated. (In contrast to a single DT, RFs are less prone to over-fitting and so offer in general better accuracy on test data.) The predictions of a RF classifier are made by majority vote of trees, that is each tree predicts for a class and the class with largest score is picked. (Note that other versions of RFs using probabilities or weights are implemented by different learning tools, e.g., scikit-learn (Pedregosa and et al. 2011), XGBoost (Chen and Guestrin 2016), etc. However, and similarly to related work (Izza and Marques-Silva 2021), this paper considers the original proposal for RFs (Breiman 2001).)

Example 1. *Figure 1 shows the running example of a random forest classifier $\mathcal{T}$ containing 3 decision trees $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$. It represents a classification function defined on the set of features $\mathcal{F} = \{1, 2, 3\}$ and set of classes $\mathcal{K} = \{c_1, c_2, c_3\} = \{\ominus, \oplus, \otimes\}$. Moreover, the domain of the features are, respectively, $\mathbb{D}_1 = \{0, 1\}$, $\mathbb{D}_2 = [0, 50]$ and $\mathbb{D}_3 = \{0, 1, 2\}$. We consider the instance $\mathbf{v} = (1, 10, 1)$ which predicted to the class $\oplus$ (i.e. c_2), the highlighted edges indicate the prediction of each tree.*

Formal explainability. An abductive explanation (Ignatiev, Narodytska, and Marques-Silva 2019a) (AXp, also referred to as a prime implicant (PI) explanation (Shih, Choi, and Darwiche 2018)) represents a minimal set of literals (relating a feature value x_i and a constant $v_i \in \mathbb{D}_i$) that are logically sufficient for the prediction. As a result, AXp's provide guarantees of rigor that are not offered by other alternative explanation approaches. More recently, AXp's have been studied in terms of their computational complexity (Marques-Silva et al. 2020; Barceló et al. 2020; Marques-Silva et al. 2021; Izza and Marques-Silva 2021; Ignatiev and Marques-Silva 2021; Cooper and Marques-Silva 2021; Audemard et al. 2021). As highlighted earlier in the paper, there is a growing body of recent work on formal explanations.

Formally, given $\mathbf{v} = (v_1, \ldots, v_m) \in \mathbb{F}$, with $\kappa(\mathbf{v}) = c$, an AXp is any subset-minimal set $\mathcal{X} \subseteq \mathcal{F}$ such that,

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

If a set $\mathcal{X} \subseteq \mathcal{F}$ is not minimal but (1) holds, then $\mathcal{X}$ is referred to as a *weak* AXp. Thus, (1) represents a predicate $\mathsf{WAXp}(\mathcal{X})$, which holds true for $\mathcal{X} \subseteq \mathcal{F}$ iff (1) also holds true. Clearly, the predicate WAXp maps $2^{\mathcal{F}}$ into $\{\bot, \top\}$. Given $\mathbf{v} \in \mathbb{F}$, an AXp $\mathcal{X}$ represents an irreducible (or minimal) subset of the features which, if assigned the values dictated by $\mathbf{v}$, are sufficient for the prediction c, i.e. value changes to the features not in $\mathcal{X}$ will not change the prediction. We can use the definition of the predicate WAXp to formalize the definition of the predicate AXp, also defined on subsets $\mathcal{X}$ of $\mathcal{F}$:

$$\mathsf{AXp}(\mathcal{X}) := \mathsf{WAXp}(\mathcal{X}) \land \forall(\mathcal{X}' \subsetneq \mathcal{X}). \neg \mathsf{WAXp}(\mathcal{X}') \quad (2)$$

The definition of $\mathsf{WAXp}(\mathcal{X})$ makes this predicate *monotone*. Indeed, if $\mathcal{X} \subseteq \mathcal{X}' \subseteq \mathcal{F}$, and if $\mathcal{X}$ is a weak AXp, then $\mathcal{X}'$ is also a weak AXp, as the fixing of more features will not change the prediction. Hence, it is the case that,

Proposition 1. *If* $\mathsf{WAXp}(\mathcal{X})$ *holds for* $\mathcal{X} \subseteq \mathcal{F}$*, then* $\mathsf{WAXp}(\mathcal{X}')$ *also holds for any* $\mathcal{X} \subseteq \mathcal{X}' \subseteq \mathcal{F}$*.*

Using Proposition 1, monotonicity of WAXp allows simplifying the definition of AXp as follows, with $\mathcal{X} \subseteq \mathcal{F}$:

$$\mathsf{AXp}(\mathcal{X}) := \mathsf{WAXp}(\mathcal{X}) \land \forall(j \in \mathcal{X}). \neg \mathsf{WAXp}(\mathcal{X} \setminus \{j\}) \quad (3)$$

This simpler but equivalent definition of AXp has important practical significance, in that only a linear number of subsets needs to be checked for, as opposed to exponentially many subsets in (2). Unsurprisingly, most algorithms that compute one AXp are based on (3).

Feature relevancy problem. With the goal of deciding whether sensitive features can be used in minimal explanations of a prediction, recent work introduced the feature relevancy (or membership) problem (Huang et al. 2021, 2022a):

Definition 1 (Feature Relevancy Problem, FRP)**.** *Given an instance* $(\mathbf{v}, c)$ *and a target feature* $t \in \mathcal{F}$*, decide whether there exists one AXp* $\mathcal{X} \subseteq \mathcal{F}$ *that contains* t*.*

Complexity-wise, for a simple disjunctive normal form (DNF) boolean classifier, FRP is hard for Σ_2^{P} (Huang et al. 2021). This means that, for well-known families of classifiers, e.g. neural networks, tree ensembles, etc. FRP is at least as hard as solving a quantified boolean formula with two levels of quantification; this problem is widely believed to be harder than solving an NP-complete problem (Arora and Barak 2009).

Example 2. *For the RF in Figure 1, let the target feature be* $t = 3$*. Then, it can be shown that there is an AXp* $\{1, 3\}$ *that contains the target feature. Clearly, if features 1 and 3 are fixed, then the value of feature 2 is irrelevant for the prediction. To prove minimality, it suffices to analyze the consequence of removing either 1 or 3 from the AXp. This can be concluded by allowing either 1 or 3 to take any value from their domain, and inferring that the prediction of* $\oplus$ *is no longer guaranteed.*

Although RFs are generally regarded as black-box ML models, and non-interpretable, recent work (Izza and Marques-Silva 2021) demonstrates that rigorous explanations can be computed efficiently in practice for large RFs[2].

Properties of FRP

This section shows that FRP is complete for Σ_2^{P} when CDP (i.e. the classification decision problem) is in NP and FRP is hard for Σ_2^{P}. Furthermore, this section shows that the Σ_2^{P} membership result can be refined, which enables far more efficient algorithms in practice.

Proposition 2. *Given a classifier for which* (1) *can be decided with an NP oracle, then* FRP *is in* Σ_2^{P}*.*

Proof. By Proposition 1 and (3), to prove that a set $\mathcal{X}$ is an AXp, it suffices to prove that:
1. $\mathsf{WAXp}(\mathcal{X}) = \top$;
2. $\forall i \in \mathcal{X}. \mathsf{WAXp}(\mathcal{X} \setminus \{i\}) = \bot$, that is, $\mathcal{X}$ is subset-minimal.

Now, since by hypothesis, deciding whether a set of features $\mathcal{X}$ is a weak AXp is in NP, then we can decide, in polynomial-time, whether some guessed set $\mathcal{X} \subseteq \mathcal{F}$ containing feature t is an AXp, as follows. For step 1., check that $\mathcal{X}$ is a weak AXp, with an NP oracle. For step 2., iteratively check, for each feature $i \in \mathcal{X}$, $\mathcal{X} \setminus \{i\}$ is not a weak AXp, again with an NP oracle. Clearly, given $\mathcal{X}$, the overall procedure runs in non-deterministic polynomial time, given access to an NP oracle. Thus FRP is in Σ_2^{P}. □

Given the above, we can prove that deciding whether feature $t \in \mathcal{F}$ is included in some explanation for $(\mathbf{v} \in \mathbb{F}, c \in \mathcal{K})$, with $c = \kappa(\mathbf{v})$, corresponds to deciding the following 2QBF statement:

$$\exists(\mathcal{X} \subseteq \mathcal{F}). (t \in \mathcal{X}) \land \mathsf{WAXp}(\mathcal{X}) \land [\land_{j \in \mathcal{X}} \neg \mathsf{WAXp}(\mathcal{X} \setminus \{j\})] \quad (4)$$

It is plain to expand the previous expression into a logic formula with two levels of quantification.

For a given classifier, e.g. a random forest (RF), we need to provide propositional encodings for the statements $[\kappa(\mathbf{x}) = c]$ and $[\kappa(\mathbf{x}) \neq c]$. These encodings can be based on those that have been developed for computing explanations for RFs (Izza and Marques-Silva 2021). Although the proof of Proposition 2 offers a solution for solving FRP, the practical solutions derived from the proof reveal key inefficiencies, namely testing in the worst case the predicate WAXp a total of $m + 1$ times, with $m = |\mathcal{F}|$. Below, we propose a different proof argument, which involves far fewer tests of WAXp. As argued later in the paper, this reduction in the number of runs of WAXp has important practical impact. The proposed approach hinges on the following result:

Proposition 3. *Let* $\mathcal{X} \subseteq \mathcal{F}$ *represent a set of features. Let* $t \in \mathcal{X}$ *denote some target feature, such that,* $\mathsf{WAXp}(\mathcal{X})$ *holds and* $\mathsf{WAXp}(\mathcal{X} \setminus \{t\})$ *does not hold. Then, for any AXp* $\mathcal{Z} \subseteq \mathcal{X} \subseteq \mathcal{F}$*, it must be the case that* $t \in \mathcal{Z}$*.*

[2]More recent work also demonstrated similar results for other tree ensembles (Ignatiev et al. 2022).

Proof. Let $\mathcal{Z} \subseteq \mathcal{F}$ by any AXp such that $\mathcal{Z} \subseteq \mathcal{X}$. Clearly, by definition $\mathsf{WAXp}(\mathcal{Z})$ must hold. Moreover, from Proposition 1, it is also the case that $\mathsf{WAXp}(\mathcal{Z}')$ must hold, with $\mathcal{Z}' = \mathcal{Z} \cup (\mathcal{X} \setminus (\mathcal{Z} \cup \{t\}))$, since $\mathcal{Z} \subseteq \mathcal{Z}' \subseteq \mathcal{F}$. But, by hypothesis, $\mathsf{WAXp}(\mathcal{X} \setminus \{t\})$ does not hold; a contradiction. □

One consequence of Proposition 3 is that a more compact 2QBF encoding can be devised:

$$\begin{aligned}&\exists(\mathcal{X} \subseteq \mathcal{F}).(t \in \mathcal{X}) \land \\ &\quad \left[\forall(\mathbf{x} \in \mathbb{F}).\left(\bigwedge\nolimits_{i \in \mathcal{X}}(x_i = v_i)\right) \rightarrow (\kappa(\mathbf{x}) = c)\right] \land \\ &\quad \left[\exists(\mathbf{x} \in \mathbb{F}).\left(\bigwedge\nolimits_{i \in \mathcal{X} \setminus \{t\}}(x_i = v_i)\right) \land (\kappa(\mathbf{x}) \neq c)\right]\end{aligned} \tag{5}$$

where predicate WAXp (defined in (1)) is already expanded.

Proposition 3 reveals a condition for finding a set of features $\mathcal{X} \subseteq \mathcal{F}$ such that any AXp contained in $\mathcal{X}$ must also contain feature t. The next two sections describe two different approaches for computing such set $\mathcal{X}$. Given the key property of $\mathcal{X}$, it then suffices to extract any AXp to have a witness of t being included in some explanation.

Encoding FRP into QBF

The results of the previous section confirm the existence of 2QBF encodings for FRP. Clearly, a concrete family of classifiers needs to be considered, and so we propose a 2QBF encoding for RFs. First, we overview an existing propositional encoding for computing AXp's of RFs. We then build on this encoding to devise a 2QBF encoding. (It is important to note that the general-purpose algorithm for FRP described in the next section is also built on this propositional encoding.)

Propositional Encoding for RFs

We start by detailing how to encode the classification function κ of an RF $\mathcal{T}$. This paper exploits the propositional encoding proposed in recent work for computing AXp's of RFs (Izza and Marques-Silva 2021)[3]. The encoding comprises: 1) the structure of an RF $\mathcal{T}$, and 2) the majority votes.

Assumption 1. *1) Each $\mathbb{D}_i$ has n_i distinct values or disjoint intervals. 2) Values/intervals are ordered (from 1 to n_i).*

To present the encoding of $\mathcal{T}$, we introduce some auxiliary boolean variables and predicates:

1. $z_{i,j}, 1 \leq i \leq m, 1 \leq j \leq n_i$. $z_{i,j} = 1$ if feature i assigned with the j-th value/interval from its domain $\mathbb{D}_i$.
2. $p_{i,j}, 1 \leq i \leq T, 1 \leq j \leq K$, $p_{i,j} = 1$ if tree T_i predicts the class j.
3. $\mathsf{Class}(R)$ denotes the class (i.e. the label of a terminal node) of a root-to-leaf path R.
4. $\mathsf{L}(R)$ denotes the set of literals of a root-to-leaf path R.
5. $\mathsf{Votes}(c)$ denotes the number of trees picking the class $c \in \mathcal{K}$.

[3]One possible alternative was proposed in more recent work (Boumazouza et al. 2021). However, this encoding is less optimized and so it does not scale as well in practice. Other representations of RFs (Choi et al. 2020; Audemard, Koriche, and Marquis 2020; Parmentier and Vidal 2021) are not applicable in this context.

To encode the structure of an RF $\mathcal{T}$, one needs to encode each $\mathcal{T}_i$. Thus, encoding a tree is achieved by encoding all its paths. The set of paths $\mathcal{R}_i$ of $\mathcal{T}_i$ is encoded as follows:

$$\bigwedge\nolimits_{R \in \mathcal{R}_i}\left(\bigwedge\nolimits_{l \in \mathsf{L}(R)} l \rightarrow \mathsf{Class}(R)\right) \tag{6}$$

Besides, each feature i is assigned with exactly one value:

$$\bigwedge\nolimits_{1 \leq i \leq m} \sum\nolimits_{l=1}^{n_i} z_{i,l} = 1 \tag{7}$$

Each tree $\mathcal{T}_i$ predicts exactly one class:

$$\bigwedge\nolimits_{1 \leq i \leq T} \sum\nolimits_{l=1}^{K} p_{i,l} = 1 \tag{8}$$

Next, we detail how to use cardinality constraints to encode the majority votes. Suppose w.l.o.g. $\mathcal{K} = \{c_{j_1}, c_{j_2}, c_{j_3}\}$ such that $j_1 < j_2 < j_3$, and the prediction of the given instance is c_{j_2}. If $(\mathsf{Votes}(c_{j_1}) < \mathsf{Votes}(c_{j_2})) \land (\mathsf{Votes}(c_{j_2}) \geq \mathsf{Votes}(c_{j_3}))$ then the prediction of $\mathcal{T}$ remain unchanged. Otherwise, if $(\mathsf{Votes}(c_{j_1}) \geq \mathsf{Votes}(c_{j_2})) \lor (\mathsf{Votes}(c_{j_2}) < \mathsf{Votes}(c_{j_3}))$ then the prediction of $\mathcal{T}$ changed.

The case where the prediction of $\mathcal{T}$ remains unchanged can be encoded via the following constraints:

$$\sum\nolimits_{i=1}^{T} p_{i,j_2} + \sum\nolimits_{i=1}^{T} \neg p_{i,j_1} \geq 1 + T \tag{9}$$

$$\sum\nolimits_{i=1}^{T} p_{i,j_2} + \sum\nolimits_{i=1}^{T} \neg p_{i,j_3} \geq T \tag{10}$$

These require the use of $K - 1$ cardinality constraints, each comparing the $\mathsf{Votes}(c_{j_2})$ with the votes of some other class. Likewise, the case where the prediction of $\mathcal{T}$ changed can be encoded via the following constraints:

$$\sum\nolimits_{i=1}^{T} p_{i,j_1} + \sum\nolimits_{i=1}^{T} \neg p_{i,j_2} \geq T \tag{11}$$

$$\sum\nolimits_{i=1}^{T} p_{i,j_3} + \sum\nolimits_{i=1}^{T} \neg p_{i,j_2} \geq 1 + T \tag{12}$$

However, in this case, only 2 (instead of $K - 1$) cardinality constraints are needed (Izza and Marques-Silva 2021).

2QBF Encoding for RFs

Given Proposition 3, we just use two copies ($\mathcal{T}^0$ and $\mathcal{T}^t$) of the same RF $\mathcal{T}$ to construct a 2QBF encoding for FRP. $\mathcal{T}^0$ encodes $\mathsf{WAXp}(\mathcal{X})$ (i.e. the prediction of $\mathcal{T}$ remains unchanged, $[\kappa^0(\mathbf{x}) = c]$), and $\mathcal{T}^t$ encodes $\neg\mathsf{WAXp}(\mathcal{X} \setminus \{t\})$ (i.e. the prediction of $\mathcal{T}$ changed, $[\kappa^t(\mathbf{x}) \neq c]$). It should be noted that the QBF encoding for (5) uses *only* two levels of quantifiers (i.e. $\exists\forall$). However, one must introduce another level of quantification to account for the auxiliary variables used for representing the matrix in clausal form. Moreover, to present the constraints of the proposed 2QBF encoding, we need to introduce additional auxiliary boolean variables:

1. $s_i, 1 \leq i \leq m$. s_i is a selector such that $s_i = 1$ iff feature i is included in $\mathcal{X}$. Moreover, $s_i = 1$ also means that feature i must be fixed to its given value v_i, while $s_i = 0$ means that feature i can take any value from its domain.

2. $\mathbf{y}_i$, $1 \leq i \leq m$. $\mathbf{y}_i$ is a set of boolean variables (a bit vector) for $\mathbb{D}_i$ such that $|\mathbf{y}_i| = \log_2 n_i$. Since values/intervals of $\mathbb{D}_i$ are ordered, each value/interval has an index (from 1 to n_i) that can be represented by an assignment of $\mathbf{y}_i$. Let $g : \mathbb{B}^{|\mathbf{y}_i|} \to \mathbb{N}$ be a function mapping binary numbers to the indices of values/intervals of $\mathbb{D}_i$. The space of $\mathbf{y}_i$ is usually larger than $\mathbb{D}_i$, but due to constraint (7) that always pick a value from $\mathbb{D}_i$, we leave some $g(\mathbf{y}_i)$ undefined. More importantly, $\mathbf{y}_i$ is activated if $i \notin \mathcal{X}$ (i.e. $s_i = 0$), and $\mathbf{y}_i$ is deactivated if $i \in \mathcal{X}$ (i.e. $s_i = 1$).

Suppose, for the given instance $\mathbf{v} = (v_1, \ldots, v_m)$, that each value v_i corresponds to the first literal $z_{i,1}$ of its domain $\mathbb{D}_i$, so the instance is represented as $\mathbf{v} = (z_{1,1}, \ldots, z_{m,1})$. Let $\Omega(\kappa^0)$ (resp. $\Omega(\kappa^t)$) denote the set of variables of the encoding of $\mathcal{T}^0$ (resp. $\mathcal{T}^t$). The QBF encoding (quantifiers and constraints) is as follows:

1. $\exists(s_1, \ldots, s_m)$
2. $\forall(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m)$
3. $\exists(\Omega(\kappa^0) \cup \Omega(\kappa^t))$
4. $\bigwedge_{1 \leq i \leq m}(s_i \rightarrow z^0_{i,1})$
5. $\bigwedge_{1 \leq i \leq m, 1 \leq j \leq n_i}(\neg s_i \wedge (g(\mathbf{y}_i) = j - 1) \rightarrow z^0_{i,j})$ [4]
6. $\left[\kappa^0(z^0_{1,1}, \ldots, z^0_{1,n_1}, \ldots, z^0_{m,1}, \ldots, z^0_{m,n_m}) = c\right]$
7. $\bigwedge_{1 \leq i \leq m, i \neq t}(s_i \rightarrow z^t_{i,1})$
8. $\left[\kappa^t(z^t_{1,1}, \ldots, z^t_{1,n_1}, \ldots, z^t_{m,1}, \ldots, z^t_{m,n_m}) \neq c\right]$
9. s_t

The first existential quantifier picks a weak AXp candidate $\mathcal{X}$. The universal quantifier considers all possible values of $\mathbb{F}$. The second existential quantifier assigns values to the remaining variables. Line 4 states, for any feature i of $\mathcal{T}^0$, if $i \in \mathcal{X}$, then it is fixed to the given value. Line 5 states, for any feature i of $\mathcal{T}^0$, if $i \notin \mathcal{X}$, then if the value represented by the $\mathbf{y}_i$ equals to $j - 1$ then feature i is assigned with j-th values/intervals. Line 6 is the propositional encoding of $\mathcal{T}^0$ comprising constraints (6) to (10). Line 7 states that, for any feature i ($i \neq t$) of $\mathcal{T}^t$, if $i \in \mathcal{X}$, then it is fixed to the given value. Line 8 is the propositional encoding of $\mathcal{T}^t$ comprising constraints (6) to (8), and constraints (11) to (12). Line 9 states that the target feature t is included in $\mathcal{X}$.

Example 3. *For the RF in Figure 1, to encode "whether there is an AXp containing feature 3." For feature 1, define variables* $\{z_{1,1}, z_{1,2}\}$ *such that* $z_{1,1} = 1$ *iff* $x_1 = 0$, $z_{1,2} = 1$ *iff* $x_1 = 1$. *For feature 2, define variables* $\{z_{2,1}, z_{2,2}\}$ *such that* $z_{2,1} = 1$ *iff* $x_2 \leq 20$ *and* $z_{2,2} = 1$ *iff* $x_2 > 20$. *And for feature 3, define variables* $\{z_{3,1}, z_{3,2}, z_{3,3}\}$ *such that* $z_{3,1} = 1$ *iff* $x_3 = 0$, $z_{3,2} = 1$ *iff* $x_3 = 1$ *and* $z_{3,3} = 1$ *iff* $x_3 = 2$. *Moreover, define bit vector* $\mathbf{y}_1 = \{y_{1,0}\}$ *mapping* $\mathbf{y}_1 = 0$ *to* $z_{1,1}$ *and* $\mathbf{y}_1 = 1$ *to* $z_{1,2}$. *Bit vector* $\mathbf{y}_2 = \{y_{2,0}\}$ *mapping* $\mathbf{y}_2 = 0$ *to* $z_{2,1}$ *and* $\mathbf{y}_2 = 1$ *to* $z_{2,2}$. *Bit vector* $\mathbf{y}_3 = \{y_{3,0}, y_{3,1}\}$ *mapping* $\mathbf{y}_3 = (0, 0)$ *to* $z_{3,1}$, $\mathbf{y}_3 = (0, 1)$ *to* $z_{3,2}$, *and* $\mathbf{y}_3 = (1, 0)$ *to* $z_{3,3}$. *The given instance* $\mathbf{v} = (1, 10, 1)$ *is* $(z_{1,2}, z_{2,1}, z_{3,2})$ *and the QBF formula is as follows:*

1. $\exists(s_1, s_2, s_3).\forall(y_{1,0}, y_{2,0}, y_{3,0}, y_{3,1}).\exists(\Omega(\kappa^0) \cup \Omega(\kappa^t))$.
2. $(s_1 \rightarrow z^0_{1,2}) \wedge (s_2 \rightarrow z^0_{2,1}) \wedge (s_3 \rightarrow z^0_{3,2})$
3. $(\neg s_1 \wedge \neg y_{1,0} \rightarrow z^0_{1,1}) \wedge (\neg s_1 \wedge y_{1,0} \rightarrow z^0_{1,2}) \wedge$
   $(\neg s_2 \wedge \neg y_{2,0} \rightarrow z^0_{2,1}) \wedge (\neg s_1 \wedge y_{2,0} \rightarrow z^0_{2,2}) \wedge$
   $(\neg s_3 \wedge \neg y_{3,0} \wedge \neg y_{3,1} \rightarrow z^0_{3,1}) \wedge$
   $(\neg s_3 \wedge \neg y_{3,0} \wedge y_{3,1} \rightarrow z^0_{3,2}) \wedge (\neg s_3 \wedge y_{3,0} \wedge \neg y_{3,1} \rightarrow z^0_{3,3})$
4. $\left[\kappa^0(z^0_{1,1}, z^0_{1,2}, z^0_{2,1}, z^0_{2,2}, z^0_{3,1}, z^0_{3,2}, z^0_{3,3}) = c\right]$
5. $(s_1 \rightarrow z^t_{1,2}) \wedge (s_2 \rightarrow z^t_{2,1})$
6. $\left[\kappa^t(z^t_{1,1}, z^t_{1,2}, z^t_{2,1}, z^t_{2,2}, z^t_{3,1}, z^t_{3,2}, z^t_{3,3}) \neq c\right]$
7. (s_3)

Moreover, it should also be noted that there is indeed a pure 2QBF encoding for FRP. It suffices to i) negate (5), and ii) decide whether the resulting formula is *false*. (In this case, the existentially quantified auxiliary variables, used for converting the matrix to clausal form, do not change the number of levels of quantification.) For the modified formula, we are now checking whether *there is no AXp containing the target feature t.* When the answer is No, it confirms the existence of an AXp $\mathcal{X}$ such that $t \in \mathcal{X}$. Next, we detail this alternative QBF encoding (with $\forall\exists$ qualtifier alternation), where we use $\mathcal{T}^0$ to encode $\neg\mathsf{WAXp}(\mathcal{X})$, and $\mathcal{T}^t$ to encode $\mathsf{WAXp}(\mathcal{X} \setminus \{t\})$:

1. $\forall(s_1, \ldots, s_m, \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m)$
2. $\exists(\Omega(\kappa^0) \cup \Omega(\kappa^t))$
3. $\bigwedge_{1 \leq i \leq m}(s_i \rightarrow z^0_{i,1})$
4. $\sigma^0 \leftrightarrow \left[\kappa^0(z^0_{1,1}, \ldots, z^0_{1,n_1}, \ldots, z^0_{m,1}, \ldots, z^0_{m,n_m}) \neq c\right]$
5. $\bigwedge_{1 \leq i \leq m, i \neq t}(s_i \rightarrow z^t_{i,1})$
6. $\bigwedge_{1 \leq i \leq m, i \neq t, 1 \leq j \leq n_i}(\neg s_i \wedge (g(\mathbf{y}_i) = j - 1) \rightarrow z^t_{i,j})$
7. $\bigwedge_{1 \leq j \leq n_t}((g(\mathbf{y}_t) = j - 1) \rightarrow z^t_{t,j})$
8. $\sigma^t \leftrightarrow \left[\kappa^t(z^t_{1,1}, \ldots, z^t_{1,n_1}, \ldots, z^t_{m,1}, \ldots, z^t_{m,n_m}) = c\right]$
9. $\neg s_t \vee \sigma^0 \vee \sigma^t$

The universal quantifier picks all possible subsets of $\mathcal{F}$ as well as all possible values of $\mathbb{F}$. The existential quantifier assigns values to the remaining variables. Line 4 is the propositional encoding of $\mathcal{T}^0$, and we associate it with a variable σ^0. Line 7 states that, for the target feature t of $\mathcal{T}^t$, it is always not fixed. Line 8 is the propositional encoding of $\mathcal{T}^t$, also associated with a variable σ^t. Line 9 states that if $t \in \mathcal{X}$ and $\mathcal{X}$ is a weak AXp then $\mathcal{X} \setminus \{t\}$ is still a weak AXp, which means there is no explanation containing the target feature t. If this is not the case, then there is an AXp containing t.

Abstraction Refinement for FRP

This section details a general-purpose algorithm for FRP, that solely requires testing whether a set of features $\mathcal{X} \subseteq \mathcal{F}$ is (or is not) a weak AXp, i.e. it just requires the ability to decide (1). The novel FRP algorithm iteratively refines an over-approximation (or abstraction) of all the subsets $\mathcal{S}$ of $\mathcal{F}$ such that: i) $\mathcal{S}$ is a weak AXp, and ii) any AXp included in $\mathcal{S}$ also includes the target feature t. Formally, the set of subsets of $\mathcal{F}$ that we are interested in is defined as follows:

$$\mathbb{H} \triangleq \{\mathcal{S} \subseteq \mathcal{F} \,|\, \mathsf{WAXp}(\mathcal{S}) \wedge \forall(\mathcal{X} \subseteq \mathcal{S}).\, [\mathsf{AXp}(\mathcal{X}) \rightarrow (t \in \mathcal{X})]\} \quad (13)$$

Proposition 4. *Let $\mathcal{X} \subseteq \mathcal{F}$. $\mathcal{X} \in \mathbb{H}$ iff $\mathcal{X}$ respects the conditions of Proposition 3.*

[4] -1 serves as offset since indices range from 1 to n_i but binary numbers range from 0 to $n_i - 1$.

Algorithm 1 Deciding FRP for an arbitrary classifier

Input: Inst. $\mathbf{v}$, Target feat. t; Feat. set $\mathcal{F}$, Classifier κ

1: **function** $\mathsf{FRPCGR}(\mathbf{v}, t; \mathcal{F}, \kappa)$
2: $\quad\mathcal{H} \leftarrow \emptyset$
3: $\quad$**repeat**
4: $\quad\quad(\mathsf{outc}, \mathbf{s}) \leftarrow \mathsf{SAT}(\mathcal{H}, s_t)$
5: $\quad\quad$**if** $\mathsf{outc} =$ **true then**
6: $\quad\quad\quad\mathcal{P} \leftarrow \{i \in \mathcal{F} \,|\, s_i = 1\}$
7: $\quad\quad\quad\mathcal{D} \leftarrow \{i \in \mathcal{F} \,|\, s_i = 0\}$
8: $\quad\quad\quad$**if** $\neg\mathsf{WAXp}(\mathcal{P})$ **then**
9: $\quad\quad\quad\quad\mathcal{H} \leftarrow \mathcal{H} \cup \mathsf{newPosCl}(\mathcal{D}; t, \kappa)$
10: $\quad\quad\quad$**else**
11: $\quad\quad\quad\quad$**if** $\neg\mathsf{WAXp}(\mathcal{P} \setminus \{t\})$ **then**
12: $\quad\quad\quad\quad\quad\mathsf{reportWeakAXp}(\mathcal{P})$
13: $\quad\quad\quad\quad\quad$**return true**
14: $\quad\quad\quad\quad\mathcal{H} \leftarrow \mathcal{H} \cup \mathsf{newNegCl}(\mathcal{P}; t, \kappa)$
15: $\quad$**until** $\mathsf{outc} =$ **false**
16: $\quad$**return false**

Algorithm 2 Create new positive clause (example)

Input: Set $\mathcal{D}$; t, κ, ...

1: **function** $\mathsf{newPosCl}(\mathcal{D}; t, \kappa, \ldots)$
2: $\quad$**for all** $i \in \mathcal{D}$ **do**
3: $\quad\quad$**if** $\neg\mathsf{WAXp}(\mathcal{F} \setminus (\mathcal{D} \setminus \{i\}))$ **then**
4: $\quad\quad\quad\mathcal{D} \leftarrow \mathcal{D} \setminus \{i\}$
5: $\quad\omega \leftarrow (\vee_{i \in \mathcal{D}} s_i)$
6: $\quad$**return** ω

Algorithm 3 Create new negative clause (example)

Input: Set $\mathcal{P}$; t, κ, ...

1: **function** $\mathsf{newNegCl}(\mathcal{P}; t, \kappa, \ldots)$
2: $\quad$**for all** $i \in \mathcal{P} \setminus \{t\}$ **do**
3: $\quad\quad$**if** $\mathsf{WAXp}(\mathcal{P} \setminus \{t, i\})$ **then**
4: $\quad\quad\quad\mathcal{P} \leftarrow \mathcal{P} \setminus \{i\}$
5: $\quad\omega \leftarrow (\vee_{i \in \mathcal{P} \setminus \{t\}} \neg s_i)$
6: $\quad$**return** ω

The proposed algorithm iteratively refines the over-approximation of set $\mathbb{H}$ until one can decide with certainty whether t is included in some AXp. The refinement step involves exploiting counterexamples as these are identified.[5] In practice, it will in general be impractical to manipulate such over-approximation of set $\mathbb{H}$ directly. As a result, we use a propositional formula (in fact a CNF formula) $\mathcal{H}$, such that the models of $\mathcal{H}$ encode the subsets of features about which we have yet to decide whether each of those subsets only contains AXp's that include t. (Formula $\mathcal{H}$ is defined on a set of Boolean variables $\{s_1, \ldots, s_m\}$, where each s_i is associated with feature i, and assigning $s_i = 1$ denotes that feature i is included in a given set.) The algorithm then iteratively refines the over-approximation by filtering out sets of sets that have been shown not to be included in $\mathbb{H}$, i.e. the so-called counterexamples.

Algorithm 1 summarizes the proposed approach[6]. Algorithms 2 and 3 provide supporting functions. We now detail the key aspects of Algorithm 1. The algorithm iteratively uses an NP oracle (in fact a SAT solver) to pick (or *guess*) a subset $\mathcal{P}$ of $\mathcal{F}$, such that any previously picked set is not repeated. Since we are interested in feature t, we enforce that $t \in \mathcal{P}$. (This step is shown in lines 4 to 7.) Given a set $\mathcal{P}$ of picked features, that includes the target feature t, we can check the conditions of Proposition 3, namely:

1. $\mathcal{P}$ is a weak AXp; and
2. $\mathcal{P} \setminus \{t\}$ is not a weak AXp.

If the two conditions above hold, then we know that $\mathcal{P}$ belongs to set $\mathbb{H}$. Furthermore, $\mathcal{P}$ represents a witness that there must exist some AXp that contains t, and we know how to compute such an AXp by starting from $\mathcal{P}$. If the picked set $\mathcal{P}$ is not a weak AXp, then we can safely remove it from further consideration. This is achieved by enforcing that at least one of the non-picked elements is picked in the future. Why? Because we want to find a set that is at least a weak AXp, and the set we picked is not one. (As can be observed $\mathcal{H}$ is updated with a positive clause that captures this constraint, as shown in line 9.) After adding the new clause, the algorithm repeats the loop. Otherwise, the picked set $\mathcal{P}$ is a weak AXp (and so the first condition above holds). As a result, we now need to check whether removing t makes $\mathcal{P} \setminus \{t\}$ not to be a weak AXp. If $\mathcal{P} \setminus \{t\}$ is not a weak AXp, then we know that any weak AXp included in $\mathcal{P}$ must include t, and this also applies to any (subset-minimal weak) AXp. In this case, the algorithm reports $\mathcal{P}$ as a weak AXp that is *guaranteed* to be included in $\mathbb{H}$. (This is shown in line 12.) It should be noted that $\mathcal{P}$ is not necessarily an AXp. However, by Proposition 3, $\mathcal{P}$ is guaranteed to be a weak AXp such that *any* of the AXp's contained in $\mathcal{P}$ *must* include feature t. Furthermore, we know that we can extract an AXp from a weak AXp with a polynomial number of calls to an oracle that decides (1), and in this case we are guaranteed to always pick one that includes t. Finally, the last case corresponds to the situation when allowing t to take any value does not cause the prediction to change. This means we picked a set $\mathcal{P}$ that is a weak AXp, but not all AXp's in $\mathcal{P}$ include the target feature t (again due to Proposition 3). As a result, we must prevent the same weak AXp from being re-picked. This is achieved by requiring that at least one of the picked features not to be picked again in the feature. (This is shown in line 14. As can be observed, $\mathcal{H}$ is updated with a negative clause that captures this constraint.)

With respect to the clauses that are added to $\mathcal{H}$ at each step, as shown in Algorithms 2 and 3, one can envision optimizations (shown in lines 2 to 3 in both algorithms) that heuristically aim at removing features from the given sets, and so produce shorter (and so logically stronger) clauses. The insight is that any feature, which can be deemed irrelevant for the condition used for constructing the clause,

[5]The approach is referred to as counterexample-guided abstraction refinement (CEGAR) FRP, since the use of counterexamples in abstraction refinement can be traced to earlier work (with the same name) for model checking of software and hardware systems (Clarke et al. 2003).

[6]The algorithms are parametrized with the arguments shown after the semi-colon; their use is flexible.

can be safely removed from the set. For the experiments, we opted to use the simplest approach for constructing the clauses, and so opting to reduce the number of classification queries. Nevertheless, simple optimizations are easy to implement. For example, with respect to the last case (i.e. adding a negative clause in 14), $\mathcal{X} \setminus \{t\}$ must be a weak AXp. From (1), this test requires deciding the satisfiability of $\wedge_{i \in \mathcal{X} \setminus \{t\}}(x_i = v_i) \wedge (\kappa(\mathbf{x}) \neq c)$, and getting an unsatisfiability result. Hence, a simple refinement of $\mathcal{P}$ is given by the unsatisfiable core yielded by the satisfiability test.

Given the above discussion, we can conclude that the proposed algorithm is sound, complete and terminating for deciding FRP for arbitrary classifiers.

Proposition 5. *For a classifier $\mathbb{C}$, defined on set of features $\mathcal{F}$, with κ mapping $\mathbb{F}$ to $\mathcal{K}$, and an instance $(\mathbf{v}, c)$, $\mathbf{v} \in \mathbb{F}$, $c \in \mathcal{K}$, and a target feature $t \in \mathcal{F}$, Algorithm 1 returns a set $\mathcal{P} \subseteq \mathcal{F}$ iff $\mathcal{P}$ is a weak AXp for $(\mathbf{v}, c)$, with the property that any AXp $\mathcal{X} \subseteq \mathcal{P}$ is such that $t \in \mathcal{X}$.*

Proof. (Sketch) A set $\mathcal{P}$ respects set $\mathbb{H}$ if $\mathcal{S}$ is a weak AXp, and any of its subsets $\mathcal{X}$ that is an AXp is such that $t \in \mathcal{X}$.

1. Algorithm 1 is terminating.
   At each step, the algorithm adds a clause that guarantees that a picked assignment is not repeated. In total, $2^{|\mathcal{F}|}$ assignments can be made to the s_i variables. Hence, the main loop of Algorithm 1 executes at most $2^{|\mathcal{F}|}$ times.
2. Algorithm 1 is sound.
   Given the conditions used to report a picked $\mathcal{P}$, then by Proposition 3 we know that this picked set respects set $\mathbb{H}$, and so any AXp contained in $\mathcal{P}$ will include t.
3. Algorithm 1 is complete.
   It is plain that each clause added to $\mathcal{H}$ blocks only sets that ought not be included in $\mathbb{H}$. The SAT solver will enumerate assignments (i.e. and so a picked set) while that set is not yet blocked by clauses added to $\mathcal{H}$. If there exists a set $\mathcal{P}$ that respects $\mathcal{H}$, then it will eventually be picked. □

Experimental Results

This section presents experimental results on assessing the practical efficiency of the two methods proposed to solving FRP in the case study of RF classifiers.

Experimental set up. The evaluation comprises 27 datasets that originate from the Penn ML Benchmarks (Olson et al. 2017). The 27 datasets are split into two sets: the first one contains 9 small datasets that have at most 16 features (the average number of features is 8.4), and which is used to compare the performances of the two methods; the second set contains 18 datasets, with an average number of 21.5 features, which mainly serves to assess the scalability of the CEGAR-based approach (denoted by FRPCGR). The RF models are trained with varying the maximum depth from 4 to 6 and the number of trees from 20 to 100, so that we obtain the most accurate models. (These numbers are in line with RFs used in practice.) As a result, small RFs (i.e. with a number of trees less or equal 30) form the first set and the large RFs (with 100 trees) constitute the second benchmark set. For each dataset, a suite of 200 samples randomly picked is tested or all input data if there are less than 200 rows in the dataset. Moreover, the candidate feature set in the query is picked randomly for each test. (Hence, for each dataset, we generate 200 FRP queries.)

Furthermore, a prototype of FRPCGR was implemented in Python[7]. The PySAT toolkit (Ignatiev, Morgado, and Marques-Silva 2018) was used to implement the FRP encodings, and configured to run the Glucose 4 (Audemard and Simon 2018)[8] SAT solver. The QBF solvers we used are DepQBF (Lonsing and Egly 2017)[9] and CAQE (Rabe and Tentrup 2015)[10]. Moreover, we combined CAQE with preprocessor Bloqqer (Biere, Lonsing, and Seidl 2011)[11]. We ran both QBF solvers with their default configurations. Finally, the experiments were performed on a MacBook Pro with a 6-Core Intel Core i7 2.6 GHz processor with 16 GByte RAM, running macOS Monterey. The time limit for deciding one query was set to 1200 seconds, and we capped the time for finishing 200 queries by 5 hours.

QBF versus FRPCGR. Table 1 summarizes the comparison results of QBF and FRPCGR. Unsurprisingly, we observe that the resulting QBF encodings are somewhat larger that the SAT encodings. Indeed, the QBF formulation encodes two copies of the RF, i.e. $\mathcal{T}^0$ and $\mathcal{T}^t$, whereas FRPCGR encodes only one, i.e. $\mathcal{T}^t$. In addition, we note that encoding $\mathcal{T}^0$ requires $(K-1)$ cardinality constraints, therefore for a multi-class problem the encoding size of QBF can be larger than the SAT encoding used by FRPCGR. Table 1 also shows the average running times of both approaches for solving one FRP query. (Note that the reported average running times are computed on successful tests, and so the tests that time out are omitted.) Clearly, the results show that FRPCGR outperforms QBF solving on all datasets. More importantly, the running times for FRPCGR are most often negligible and at least one order of magnitude smaller than running times for QBF. Furthermore, we observe that in some datasets, QBF solvers were unable to terminate for some tests (e.g. 2 timeouts (resp. 1) for CAQE (resp. DepQBF) with *crx* dataset (resp. *glass2* dataset)) or all tests (e.g. DepQBF with *crx* dataset). Additionally, we evaluated the alternative $\forall\exists$ QBF encoding for the datasets reported in Table 1. It is still the case that runtime for QBF solvers solving these $\forall\exists$ instances is at least one order of magnitude larger than runtime for FRPCGR. Moreover, and in most cases, solving $\forall\exists$ (clausal) instances is significantly slower than solving $\exists\forall$ non-clausal (and so $\exists\forall\exists$ clausal) instances.

FRPCGR. Since the main goal is to assess the scalability of FRPCGR on large RFs (of sizes common in practical applications), instances obtained from RFs with 100 trees (as described earlier) were also considered. The number of

[7] All the materials for replicating the experiments are available at https://github.com/XuanxiangHuang/frpRF-experiments

[8] https://github.com/mi-ki/glucose-syrup

[9] https://github.com/lonsing/depqbf

[10] https://github.com/ltentrup/caqe

[11] http://fmv.jku.at/bloqqer/

Dataset	m	K	RF		Y%	QBF		DepQBF		CAQE		CNF		FRPCGR	
			#N	A%		vars	clauses	Yes	No	Yes	No	vars	clauses	Yes	No
crx	15	2	522	81.8	90	2579	5260	—	—	12.1*	36.70	1290	2719	0.03	0.05
ecoli	7	5	526	87.8	60	7448	12 376	1.40	29.38	3.91	2.25	2917	5103	0.04	0.02
glass2	9	2	348	84.8	87	2414	4744	18.5*	48.38	1.26	1.73	1202	2440	0.01	0.01
hayes_roth	4	3	336	84.2	71	5036	7832	0.07	0.08	2.30	2.94	2555	4017	0.02	0.01
votes_84	16	2	464	91.3	97	1266	2458	66.51	36.46	0.61	0.58	643	1256	0.01	0.05
iris	4	3	224	100	52	5079	8012	0.11	0.09	1.56	1.79	2564	4092	0.02	0.01
mofn	10	2	582	86.3	40	859	1851	0.03	0.04	0.22	0.28	440	931	0.01	0.01
monk3	6	2	472	94.3	22	937	1868	0.02	0.03	0.16	0.20	473	937	0.01	0.00
n_thyroid	5	3	284	100	83	5722	9386	4.24	0.97	2.23	2.18	2884	4807	0.02	0.01

Table 1: Comparison of QBF-based and CEGAR-based methods. Columns m, K report the characteristics of the dataset, namely number of features and classes, respectively. Sub-columns #N, A% in column RF report, resp., the number of nodes and the test accuracy of the trained models. Column Y% counts the number of FRP queries answered Yes (in percentage). Column QBF shows the average number of variables and clauses in the QBF encoding. Average runtimes for solving FRP queries with QBF solvers are reported (in seconds) in columns DepQBF and CAQE, s.t. times of resulting FRP answers Yes and No are reported separately. ("—" indicates that solver reached the fixed timeout for all tests; "*" indicates that timeouts are observed for some tests.) Column CNF shows the average number of variables and clauses of the SAT encoding generated by the FRPCGR. The last column reports the average times (in seconds) of the FRPCGR for solving one FRP query, again times of answers Yes and No are reported separately.

Dataset	m	K	RF		CNF		Y%	AXp	Time		#SAT calls	
			#N	A%	vars	clauses		avg. sz	Yes	No	Yes	No
agaricus_lepiota	22	2	1866	99.2	3343	6310	89	10	0.2	4.7	52	2538
allbp	29	3	2492	96.5	16 038	26 452	47	4	2.6	4.3	65	261
ann_thyroid	21	3	2192	98.9	16 802	27 509	26	6	1.0	1.0	33	75
appendicitis	7	2	1426	90.9	4674	8736	97	4	0.1	0.1	4	20
collins	23	13	2890	86.6	24 772	42 186	95	12	3.4	0.4	38	16
hypothyroid	25	2	2034	95.9	4768	9347	53	4	0.4	1.3	32	324
ionosphere	34	2	1566	87.1	5922	12 594	98	19	6.4	0.6	1272	232
kr_vs_kp	36	2	2268	94.2	2952	8102	71	11	0.6	20.5	285	11 261
magic	10	2	2990	81.9	10 631	22 403	86	6	0.2	0.1	14	36
mushroom	22	2	2078	99.0	3374	6386	90	11	0.2	2.8	46	1375
pendigits	16	10	3098	85.0	22 656	38 420	99	10	1.6	1.5	18	70
ring	20	2	2458	84.1	9113	18 815	68	15	0.2	0.5	20	130
segmentation	19	7	2288	92.8	20 822	35 114	91	9	1.6	4.5	45	290
shuttle	9	7	2618	99.8	19 543	31 942	78	4	0.9	0.9	14	31
texture	40	11	3040	81.4	27 018	47 325	97	23	6.9	62.0	210	5522
twonorm	20	2	3100	93.5	11 729	24 904	94	12	0.3	10.6	25	2606
vowel	13	11	10 176	90.4	44 530	88 700	98	9	4.1	5.7	19	56
waveform_21	21	3	3100	83.5	22 446	39 732	75	10	1.2	12.6	47	943

Table 2: Assessing FRPCGR on larger datasets and RFs trained with 100 trees. Column AXp reports the average size of computed AXp's for queries answered Yes. Column Time reports average runtimes (in seconds) for solving one FRP query. Column #SAT calls reports the average number of oracle guesses (counterexamples) performed in the CEGAR loop (i.e. number of iterations) to solve one FRP query. The remaining columns have the same meaning as described in the caption of Table 1.

nodes in these RFs ranges from 1426 to 10176. The results are shown in Table 2. As can be observed, the average running times of FRPCGR to decide FRP take from 0.1s to 6.9s (resp. 0.1s to 62s) for outputs "Yes" (resp. "No"). It should be underscored that FRPCGR computes in general more counterexamples to solve a negative decision (i.e. answer No), as this can be confirmed from the results, where the number of calls to the SAT oracle are substantially larger for decisions answered No on the majority of datasets (e.g. 11 261 calls for No against 285 for Yes, for *kr_vs_kp* dataset). As a result, and with a few exceptions, the running times of FRP tests of output No are larger than tests answered Yes. Also, we emphasize that in contrast to the QBF solvers, no timeouts were observed with FRPCGR, for the results

Datasets	DepQBF					CAQE				
	#Test(TO)	Yes Time		No Time		#Test(TO)	Yes Time		No Time	
		Total	Avg.	Total	Avg.		Total	Avg.	Total	Avg.
crx	21(14)	614.33	122.87	68.43	68.43	200(2)	2134.4	12.1	770.6	36.7
agaricus_lepiota	17(15)	0.1	0.1	0.0	0.0	68(13)	2138.7	42.8	36.4	9.1
allbp	17(15)	*	*	0.0	0.0	17(15)	*	*	0.0	0.0
ann_thyroid	23(14)	1.9	0.6	731.5	146.3	49(13)	1083.6	180.6	909.1	31.3
appendicitis	19(14)	1095.0	273.7	*	*	27(13)	1516.3	126.4	643.5	643.5
collins	16(15)	*	*	0.0	0.0	16(14)	1199.4	1199.4	0.0	0.0
hypothyroid	15(15)	*	*	*	*	18(13)	973.1	486.6	698.4	349.2
ionosphere	16(15)	0.2	0.2	*	*	15(15)	*	*	*	*
kr_vs_kp	21(14)	814.9	163.0	0.0	0.0	29(13)	1430.6	102.2	0.0	0.0
magic	22(14)	638.2	91.2	*	*	20(13)	1692.7	423.2	52.2	26.1
mushroom	18(14)	1018.1	1018.1	0.0	0.0	37(12)	3238.7	161.9	0.0	0.0
pendigits	16(15)	1.8	1.8	*	*	17(14)	647.9	323.9	*	*
ring	20(14)	1103.3	220.7	*	*	72(14)	707.6	14.7	573.6	57.4
segmentation	16(15)	1.2	1.2	*	*	17(14)	307.1	153.5	*	*
shuttle	16(15)	1.2	1.2	*	*	24(12)	2148.2	268.5	425.8	141.9
texture	15(15)	*	*	*	*	15(15)	*	*	*	*
twonorm	18(14)	15.9	8.0	383.2	383.2	16(14)	*	*	23.5	23.5
vowel	21(14)	715.6	119.3	*	*	17(12)	3500.2	875.1	*	*
waveform_21	17(15)	1.6	0.8	*	*	15(15)	*	*	*	*

Table 3: QBF method solves queries for dataset crx and datasets in Table 2. The first row shows the timeout information for dataset crx. The rest show the timeout information for datasets in Table 2. #Test(TO) shows the number of FRP queries tested in 5 hours, inside the parentheses is the number of timeout queries. If the query is solved, then its total and average time (in seconds) are reported in Column Yes Time and No Time. A '*' indicates out of time.

shown in both tables. Table 3 shows the timeout information when solving the queries for datasets of Table 2 with the QBF encoding. For example, when using DepQBF to solve queries of dataset *crx*, only 21 queries were tested, and only 14 queries were solved, 7 queries out of 21 cannot be solved in 1200 seconds time limit, and the rest queries were not tested due to the 5 hours time limit. Note that, for datasets *agaricus_lepiota*, *allbp*, *collins*, *kr_vs_kp* and *mushroom*, the total No time is 0.0, this is because the target feature t is not tested in the RF, which means no AXp containing feature t exists. One observation is that CAQE + Bloqqer can solve more queries (compared with DepQBF). But for dataset *texture*, both solvers fail to answer any query. Another observation is that FRPCGR is effective in practice and usable on large size RFs induced from realistic datasets. Furthermore, the results also indicate that FRPCGR substantially outperforms the encoding to QBF, being able to solve a vast number of FRP queries that QBF solvers are unable to.

Conclusions & Research Directions

This paper investigates the solution of quantification problems when answering explainability queries, focusing on the feature relevancy problem, FRP. The paper proves that for several families of classifiers, FRP is complete for Σ_2^{P}. Furthermore, the paper proposes two solutions for solving FRP. The first one builds on the proof of Σ_2^{P} completeness, and is based on encoding the problem into QBF, and then solving the QBF formulas with a state-of-the-art QBF solver. The second approach proposes a dedicated abstraction refinement algorithm for FRP. Random forest classifiers were considered for comparing the two approaches. The experiments show a clear performance edge of the dedicated algorithm when compared with the proposed QBF encodings, since the dedicated algorithm is the only capable of deciding FRP for large RFs, resulting from datasets with a realistic number of features. Research directions include fine-tuning all of the main steps of FRP, e.g. selecting the set to pick, but also the derivation of clauses when refining $\mathbb{H}$.

Acknowledgments

This work was supported by the AI Interdisciplinary Institute ANITI, funded by the French program "Investing for the Future – PIA3" under Grant agreement no. ANR-19-PI3A-0004, by the H2020-ICT38 project COALA "Cognitive Assisted agile manufacturing for a Labor force supported by trustworthy Artificial intelligence", and by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme. This work received comments and suggestions from several colleagues, including N. Asher, M. Cooper, A. Ignatiev, J. Planes, A. Morgado, and N. Narodytska. JMS also acknowledges the incentive provided by the ERC who, by not funding this research nor a handful of other grant applications between 2012 and 2022, has had a lasting impact in framing the research presented in this paper.

References

Amgoud, L. 2021. Non-monotonic Explanation Functions. In *ECSQARU*, 19–31.

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

Arenas, M.; Baez, D.; Barceló, P.; Pérez, J.; and Subercaseaux, B. 2021. Foundations of Symbolic Languages for Model Interpretability. In *NeurIPS*, 11690–11701.

Arenas, M.; Barceló, P.; Romero, M.; and Subercaseaux, B. 2022. On Computing Probabilistic Explanations for Decision Trees. In *NeurIPS*.

Arora, S.; and Barak, B. 2009. *Computational Complexity - A Modern Approach*. Cambridge University Press. ISBN 978-0-521-42426-4.

Audemard, G.; Bellart, S.; Bounia, L.; Koriche, F.; Lagniez, J.; and Marquis, P. 2021. On the Computational Intelligibility of Boolean Classifiers. In *KR*, 74–86.

Audemard, G.; Bellart, S.; Bounia, L.; Koriche, F.; Lagniez, J.; and Marquis, P. 2022a. On Preferred Abductive Explanations for Decision Trees and Random Forests. In *IJCAI*, 643–650.

Audemard, G.; Bellart, S.; Bounia, L.; Koriche, F.; Lagniez, J.; and Marquis, P. 2022b. Trading Complexity for Sparsity in Random Forest Explanations. In *AAAI*, 5461–5469.

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

Audemard, G.; and Simon, L. 2018. On the Glucose SAT Solver. *Int. J. Artif. Intell. Tools*, 27(1): 1840001:1–1840001:25.

Barceló, P.; Monet, M.; Pérez, J.; and Subercaseaux, B. 2020. Model Interpretability through the lens of Computational Complexity. In *NeurIPS*.

Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds. 2021. *Handbook of Satisfiability - Second Edition*. IOS Press.

Biere, A.; Lonsing, F.; and Seidl, M. 2011. Blocked Clause Elimination for QBF. In *CADE*, 101–115.

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

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

Breiman, L. 2001. Random Forests. *Mach. Learn.*, 45(1): 5–32.

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

Choi, A.; Shih, A.; Goyanka, A.; and Darwiche, A. 2020. On Symbolically Encoding the Behavior of Random Forests. *CoRR*, abs/2007.01493.

Clarke, E. M.; Grumberg, O.; Jha, S.; Lu, Y.; and Veith, H. 2003. Counterexample-guided abstraction refinement for symbolic model checking. *J. ACM*, 50(5): 752–794.

Cooper, M. C.; and Marques-Silva, J. 2021. On the Tractability of Explaining Decisions of Classifiers. In Michel, L. D., ed., *CP*, 21:1–21:18.

Darwiche, A. 2020. Three Modern Roles for Logic in AI. In *PODS*, 229–243.

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

Darwiche, A.; and Ji, C. 2022. On the Computation of Necessary and Sufficient Explanations. In *AAAI*, 5582–5591.

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

Eiter, T.; and Gottlob, G. 1995. The Complexity of Logic-Based Abduction. *J. ACM*, 42(1): 3–42.

European Commission. 2016. General Data Protection Regulation. http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679&from=en. Accessed: 2023-03-23.

European Commission. 2021. Artificial Intelligence Act. https://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1623335154975&uri=CELEX%3A52021PC0206. Accessed: 2023-03-23.

European Commission's High-Level Expert Group on AI. 2019. Ethics guidelines for trustworthy AI. https://ec.europa.eu/digital-single-market/en/news/ethics-guidelines-trustworthy-ai. Accessed: 2023-03-23.

Feng, J.; and Zhou, Z. 2018. AutoEncoder by Forest. In *AAAI*, 2967–2973.

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

Friedrich, G.; Gottlob, G.; and Nejdl, W. 1990. Hypothesis Classification, Abductive Diagnosis and Therapy. In *ESE*, 69–78.

Gao, W.; and Zhou, Z. 2020. Towards Convergence Rate Analysis of Random Forests for Classification. In *NeurIPS*.

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

Huang, X.; Cooper, M. C.; Morgado, A.; Planes, J.; and Marques-Silva, J. 2022a. Feature Necessity & Relevancy in ML Classifier Explanations. *CoRR*, abs/2210.15675.

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

Huang, X.; Izza, Y.; Ignatiev, A.; and Marques-Silva, J. 2021. On Efficiently Explaining Graph-Based Classifiers. In *KR*, 356–367.

Huang, X.; and Marques-Silva, J. 2022. On Deciding Feature Membership in Explanations of SDD & Related Classifiers. *CoRR*, abs/2202.07553.

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

Ignatiev, A.; Cooper, M. C.; Siala, M.; Hebrard, E.; and Marques-Silva, J. 2020a. Towards Formal Fairness in Machine Learning. In *CP*, 846–867.

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

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

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

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

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

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

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

Izza, Y.; Ignatiev, A.; and Marques-Silva, J. 2020. On Explaining Decision Trees. *CoRR*, abs/2010.11034.

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

Izza, Y.; Ignatiev, A.; Narodytska, N.; Cooper, M. C.; and Marques-Silva, J. 2021. Efficient Explanations With Relevant Sets. *CoRR*, abs/2106.00546.

Izza, Y.; Ignatiev, A.; Narodytska, N.; Cooper, M. C.; and Marques-Silva, J. 2022. Provably Precise, Succinct and Efficient Explanations for Decision Trees. *CoRR*, abs/2205.09569.

Izza, Y.; and Marques-Silva, J. 2021. On Explaining Random Forests with SAT. In *IJCAI*, 2584–2591.

Izza, Y.; and Marques-Silva, J. 2022. On Computing Relevant Features for Explaining NBCs. *CoRR*, abs/2207.04748.

Liu, X.; and Lorini, E. 2022. A Logic of "Black Box" Classifier Systems. In *WoLLIC*, 158–174.

Lonsing, F.; and Egly, U. 2017. DepQBF 6.0: A Search-Based QBF Solver Beyond Traditional QCDCL. In *CADE*, 371–384.

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

Makortoff, K. 2022. 'Risks posed by AI are real': EU moves to beat the algorithms that ruin lives. https://www.theguardian.com/technology/2022/aug/07/ai-eu-moves-to-beat-the-algorithms-that-ruin-lives. Accessed: 2023-03-23.

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

Marques-Silva, J. 2022a. Logic-Based Explainability in Machine Learning. *CoRR*, abs/2211.00541.

Marques-Silva, J. 2022b. Logic-Based Explainability in Machine Learning. In *Reasoning Web*. In Press.

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

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

Marques-Silva, J.; and Ignatiev, A. 2022. Delivering Trustworthy AI through Formal XAI. In *AAAI*, 12342–12350.

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

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

Parmentier, A.; and Vidal, T. 2021. Optimal Counterfactual Explanations in Tree Ensembles. In *ICML*, 8422–8431.

Pedregosa, F.; and et al. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830.

Rabe, M. N.; and Tentrup, L. 2015. CAQE: A Certifying QBF Solver. In *FMCAD*, 136–143.

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

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

Selman, B.; and Levesque, H. J. 1990. Abductive and Default Reasoning: A Computational Core. In *AAAI*, 343–348.

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

Shih, A.; Choi, A.; and Darwiche, A. 2019. Compiling Bayesian Network Classifiers into Decision Graphs. In *AAAI*, 7966–7974.

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

Wolf, L.; Galanti, T.; and Hazan, T. 2019. A Formal Approach to Explainability. In *AIES*, 255–261.

Yang, L.; Wu, X.; Jiang, Y.; and Zhou, Z. 2020. Multi-Label Learning with Deep Forest. In *ECAI*, volume 325, 1634–1641.

Yu, J.; Ignatiev, A.; Stuckey, P. J.; Narodytska, N.; and Marques-Silva, J. 2022. Eliminating The Impossible, Whatever Remains Must Be True. *CoRR*, abs/2206.09551.

Zhang, Y.; Zhou, J.; Zheng, W.; Feng, J.; Li, L.; Liu, Z.; Li, M.; Zhang, Z.; Chen, C.; Li, X.; Qi, Y. A.; and Zhou, Z. 2019. Distributed Deep Forest and its Application to Automatic Detection of Cash-Out Fraud. *ACM Trans. Intell. Syst. Technol.*, 10(5): 55:1–55:19.

Zhou, Z.; and Feng, J. 2017. Deep Forest: Towards An Alternative to Deep Neural Networks. In *IJCAI*, 3553–3559.