The Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI-23)

Explaining Random Forests Using Bipolar Argumentation and Markov Networks

Nico Potyka, Xiang Yin, Francesca Toni

Department of Computing, Imperial College London, London, UK
{n.potyka, x.yin20, f.toni}@imperial.ac.uk

Abstract

Random forests are decision tree ensembles that can be used to solve a variety of machine learning problems. However, as the number of trees and their individual size can be large, their decision making process is often incomprehensible. We show that their decision process can be naturally represented as an argumentation problem, which allows creating global explanations via argumentative reasoning. We generalize sufficient and necessary argumentative explanations using a Markov network encoding, discuss the relevance of these explanations and establish relationships to families of abductive explanations from the literature. As the complexity of the explanation problems is high, we present an efficient approximation algorithm with probabilistic approximation guarantees.

1 Introduction and Related Work

Random forests (RFs) (Breiman 2001) are machine learning models with various applications in areas like E-commerce, Finance and Medicine. They consist of multiple decision trees that use different subsets of the available features. Given an input, every tree makes an individual decision and the output of the random forest is obtained by a majority vote. They have low risk of overfitting; support both classification and regression tasks and come equipped with some feature importance measures (Breiman 2001). However, feature importance measures can be too simplistic as they can represent neither joint effects of features (e.g., multi-drug interactions) nor non-monotonicity (e.g., increasing the weight may be healthy for an underweight person, but not for an overweight person).

In recent years, a variety of other explanation methods has been proposed. Model-agnostic feature importance measures like LIME (Ribeiro, Singh, and Guestrin 2016), SHAP (Lundberg and Lee 2017) and MAPLE (Plumb, Molitor, and Talwalkar 2018) have similar limitations like the feature importance measures defined for random forests. Counterfactual explanations explain how an input can be modified to change the decision (Wachter, Mittelstadt, and Russell 2017), but mainly explain the model locally. Another interesting family of explanation methods are abductive explanations, also called prime implicant explanations (Shih, Choi, and Darwiche 2018; Izza and Marques-Silva 2021; Wäldchen et al. 2021). Roughly speaking, abductive explanations are sufficient reasons for a classification. Recently, SAT encodings have been applied to compute abductive explanations in tree ensembles (Izza and Marques-Silva 2021; Ignatiev et al. 2022) and many other logic-based explanation approaches have been investigated (Marques-Silva and Ignatiev 2022; Cyras et al. 2021; Vassiliades, Bassiliades, and Patkos 2021).

As random forests are essentially composed of rules, a natural question is if we can use logical tools to reason in more flexible ways about random forests. Since the rules can be mutually inconsistent, non-classical reasoning approaches are a natural choice. Here, we investigate abstract bipolar argumentation graphs (BAGs) (Amgoud et al. 2008; Oren and Norman 2008; Boella et al. 2010; Cayrol and Lagasquie-Schiex 2013) for this purpose. Intuitively, BAGs allow identifying consistent subsets (extensions) of contradicting arguments and to reason about them. We will show that the bi-stable semantics for BAGs (Potyka 2021) allows representing random forests as BAGs such that the possible decisions made by the forest correspond to extensions of the BAG. Finding sufficient and necessary reasons for the classification of a random forest can then be reduced to finding sufficient and necessary reasons in argumentation frameworks (Borg and Bex 2021). In order to solve the combinatorial reasoning problems, we consider Markov network encodings of the BAG (Potyka 2020), which also allow reasoning about almost sufficient and almost necessary reasons. As the computational complexity of the problems is high, we consider a probabilistic algorithm to approximate reasons and present first experimental results.

The proofs of all technical results can be found in the accompanying technical report (Potyka, Yin, and Toni 2022).

2 Random Forests and Classes of AXps

We will focus on Random forests for classification problems here. The goal of classification is to assign class labels y to inputs $\mathbf{x}$. Inputs are vectors $\mathbf{x} = (x_1, \ldots, x_k)$, where the i-th value belongs to some feature X_i with domain D_i. We let $\mathcal{D} = \bigtimes_{i=1}^{k} D_i$ denote the set of all inputs and $\mathcal{C}$ the set of *class labels*. Figure 1 shows two decision trees for a medical classification problem where patients are diagnosed based on their age and three symptoms A, B, C that can be present (1) or not (0). The diagnosis can

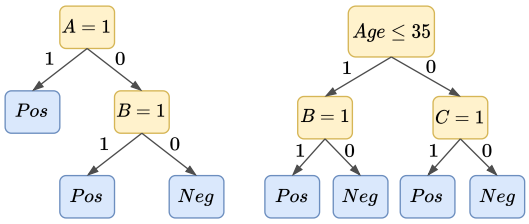

Figure 1: A simple random forest with two decision trees.

be positive (Pos) or negative (Neg). Formally, we understand trees as sets of rules $\mathcal{T} = \{r_1, \ldots, r_{|\mathcal{T}|}\}$. A rule r has the form $\text{prem}(r) \rightarrow \text{conc}(r)$, where the premise $\text{prem}(r)$ is a set of *feature literals* and the conclusion $\text{conc}(r) \in \mathcal{C}$ a class label. Feature literals are positive or negative *feature conditions*. Feature conditions (positive feature literals) have the form $X_i = v_i$ (categorical features) or $X_i \leq v_i$ (ordinal/numerical features), where $v_i \in D_i$. Negative feature literals are negated feature conditions. For example, the tree on the left in Figure 1 can be represented by the three rules $\{A = 1\} \rightarrow Pos$, $\{A \neq 1, B = 1\} \rightarrow Pos$, $\{A \neq 1, B \neq 1\} \rightarrow Neg$. Note that the rules are exhaustive and exclusive, that is, for every input $\mathbf{x} \in \mathcal{D}$, there is one and only one rule that applies. We call this rule the *active rule in $\mathcal{T}$ for $\mathbf{x}$*.

A random forest $\mathcal{F} = \{\mathcal{T}_1, \ldots, \mathcal{T}_t\}$ is a collection of decision trees. It processes an input $\mathbf{x}$ by computing the outputs $y_1, \ldots, y_t$ for $\mathbf{x}$ for all decision trees. Then, it returns the class label that was selected most frequently. We assume that $\bot$ is returned if multiple class labels receive the maximum number of votes (a *tie*). We let $\text{O}_\mathcal{F} : \mathcal{D} \rightarrow \mathcal{C} \cup \{\bot\}$ denote the *output function* for $\mathcal{F}$, where $\text{O}_\mathcal{F}(\mathbf{x}) = y$ if $\mathcal{F}$ outputs class y for $\mathbf{x}$ and $\text{O}_\mathcal{F}(\mathbf{x}) = \bot$ if there is a tie.

Example 1. *Consider the random forest composed of the two trees in Figure 1 and a patient aged 25 with symptoms A and B present. Then both decision trees will return Pos and the output of the random forest is Pos. If symtom B was not present, the decision tree on the left would return Pos, while the decision tree on the right would return Neg. In this case, the output is $\bot$.*

In order to understand the decision process of random forests, we can consider abductive explanations (Shih, Choi, and Darwiche 2018; Izza and Marques-Silva 2021). A *weak abductive explanation (wAXp)*, for a class label $y \in \mathcal{C}$ is a partial assignment to the features such that every completion $\mathbf{x}$ of this partial assignment satisfies $\text{O}(\mathbf{x}) = y$ (Huang et al. 2022). If a wAXp cannot be shortened, then it is called an *abductive explanation (AXp)* or prime implicant (Huang et al. 2022). For example, the partial assignment $(A = 1, B = 1, Age = 20)$ is a wAXp for Pos with respect to the random forest in Figure 1. However, it is not an AXp because it can be shortened to $(B = 1, Age = 20)$, which is, in fact, an AXp. (Wäldchen et al. 2021) recently generalized AXps. Roughly speaking, a partial assignment is called a *δ-relevant explanation for class y* if the probability that a completion satisfies $\text{O}(\mathbf{x}) = y$ is at least δ (Wäldchen et al. 2021) (where we consider a uniform distribution over the completions). For brevity, we will call them δ-AXps in the following. Note that 1-AXps are wAXps. Finding and even deciding if a partial assignment is an (δ-)AXp is difficult as complexity results in (Izza and Marques-Silva 2021) and (Wäldchen et al. 2021) show.

3 Ambiguous and Indistinguishable Inputs

Random forests may be unable to make a decision due to a tie in the individual tree decisions. For binary classification problems, we can always avoid a tie by creating a forest with an odd number of trees. However, if we have more than two classes, there is no simple workaround. We call the undecided inputs *ambiguous* and let $\text{Amb}(\mathcal{F}) = \{\mathbf{x} \in \mathcal{D} \mid \text{O}_\mathcal{F}(\mathbf{x}) = \bot\}$ denote the set of all ambiguous inputs.

The following proposition explains that analyzing ambiguity is a difficult problem even for simple random forests that contain only boolean features and have at most 4 leaves/rules. We call this special case *B4L random forests*.

Proposition 1.
- *Deciding if there exists an ambiguous input for a B4L random forest is NP-complete.*
- *Counting the number of ambiguous inputs for a B4L random forest is #P-complete.*

If $\mathcal{F}$ contains variables with infinite domains, the number of ambiguous inputs can be infinite. However, it is always possible to partition the inputs into a finite set of equivalence classes. To make this more precise, let us first note that every random forest yields a natural partition of the input domains based on the feature conditions that occur in the forest.

Definition 1 (Domain Partition)**.** The *domain partition associated with $\mathcal{F}$* partitions every domain D_i into disjoint subsets $S_{i,1}, \ldots, S_{i,n_i}$ such that $D_i = \uplus_{j=1}^{n_i} S_{i,j}$. If $D_i = \{v_1, \ldots, v_{|D_i|}\}$ is finite, then $n_i = |D_i|$ and $S_{i,j} = \{v_j\}$. If D_i is continuous, let $X_i \leq v_1, \ldots, X_i \leq v_{|D_i|}$ denote the feature conditions that occur in $\mathcal{F}$ for X_i and assume w.l.o.g. that $v_1 \leq \cdots \leq v_{|D_i|}$. Then $n_i = |D_i| + 1$, $S_{i,j} = (v_{j-1}, v_j] = \{v \in D_i \mid v_{j-1} < v \leq v_j\}$ where $v_0 = \inf D_i$, and $S_{|D_i|+1} = [v_{|D_i|}, \sup D_i)$.

Example 2. *For the random forest in Figure 1, the domains of A, B and C are partitioned into $\{0\}$ and $\{1\}$. For Age, the domain is partitioned into $(-\infty, 35]$ and $(35, \infty)$.*

Note that the number of partitioning sets is always finite because random forests are finite. Furthermore, when we chose one partition index i_j for every feature X_i, then all inputs in $S_{i_1} \times \cdots \times S_{i_k} \subseteq \mathcal{D}$ are indistinguishable for the trees and, therefore, are all classified in the same way. To capture these indistinguishable inputs, we define the *characteristic function of $\mathcal{F}$* as the mapping $\chi_\mathcal{F} : \mathcal{D} \rightarrow \mathbb{N}^k$ that maps every input $\mathbf{x}$ to a k-dimensional vector $v = \chi_\mathcal{F}(\mathbf{x})$ such that $x_i \in S_{i,v_i}$ for all $i = 1, \ldots, k$.

Definition 2 (Indistinguishability Relation)**.** Two inputs $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{D}$ are *indistinguishable with respect to $\mathcal{F}$* iff $\chi_\mathcal{F}(\mathbf{x}_1) = \chi_\mathcal{F}(\mathbf{x}_2)$. We denote this by $\mathbf{x}_1 \equiv_\mathcal{F} \mathbf{x}_2$.

It is easy to check that indistinguishability is an equivalence relation and that the equivalence classes $E \in \mathcal{D}/\equiv_\mathcal{F}$

correspond to the sets $S_{i_1} \times \cdots \times S_{i_k}$ that we obtain by choosing one partition index i_j for every feature.

Let us note that while $\mathrm{Amb}(\mathcal{F})$ can be infinite, the set of equivalence classes of ambiguous inputs $\mathrm{Amb}(\mathcal{F})/\equiv_{\mathcal{F}}$ is always finite. Hence, we can now ask, what is the number of ambiguous equivalence classes? If all domains are finite, this is equivalent to counting the number of ambiguous inputs because, in this case, every equivalence class contains exactly one input. Hence, Proposition 1 implies that counting the ambiguous equivalence classes is $\#P$-hard as well.

4 Representing Random Forests as BAGs

In order to reason about the decision process of random forests, we represent it as a *bipolar argumentation graph (BAG)*. Formally, a BAG is a tuple $\mathcal{B} = (\mathcal{A}, \mathrm{Att}, \mathrm{Sup})$, where $\mathcal{A}$ is a finite set of arguments, $\mathrm{Att} \subseteq \mathcal{A} \times \mathcal{A}$ is the *attack relation* and $\mathrm{Sup} \subseteq \mathcal{A} \times \mathcal{A}$ is the support relation (Cayrol and Lagasquie-Schiex 2013). We let $\mathrm{Att}(A) = \{B \mid (B, A) \in \mathrm{Att}\}$ denote the attackers of A and, analogically, $\mathrm{Sup}(A)$ its supporters.

Various semantics have been proposed for BAGs. We use the bi-complete semantics from (Potyka 2021) here, which generalizes the complete semantics (Dung 1995) and resolves conflicts between attackers and supporters by means of majority votes. It is based on labellings $L : \mathcal{A} \to \{\mathrm{in}, \mathrm{out}, \mathrm{und}\}$ that assign a label in (accept), out (reject) or und (undecided) to every argument. Given a labelling L, we say that the attackers of an argument *dominate* its supporters if $|\{B \in \mathrm{Att}(A) \mid \mathrm{L}(B) = \mathrm{in}\}| > |\{B \in \mathrm{Sup}(A) \mid \mathrm{L}(B) \neq \mathrm{out}\}|$. That is, for every supporter that is not out, there is an attacker that is in and there is at least one additional attacker that is in. Intuitively, every non-rejected pro-argument is balanced out by an accepted counterargument and there is an additional counterargument that breaks a potential tie. Symmetrically, the supporters of an argument *dominate* its attackers if $|\{B \in \mathrm{Sup}(A) \mid \mathrm{L}(B) = \mathrm{in}\}| > |\{B \in \mathrm{Att}(A) \mid \mathrm{L}(B) \neq \mathrm{out}\}|$. Given a BAF $(\mathcal{A}, \mathrm{Att}, \mathrm{Sup})$, we call a labelling $\mathrm{L} : \mathcal{A} \to \{\mathrm{in}, \mathrm{out}, \mathrm{und}\}$

Bi-complete (Potyka 2021): if L satisfies

1. $\mathrm{L}(A) = \mathrm{in}$ if and only if $\mathrm{L}(B) = \mathrm{out}$ for all $B \in \mathrm{Att}(A)$ or A's supporters dominate its attackers.
2. $\mathrm{L}(A) = \mathrm{out}$ if and only if A's attackers dominate its supporters.

A bi-complete labelling is called *bi-stable* if it does not label any argument undecided. We let $\mathcal{L}^c(\mathcal{B})$ and $\mathcal{L}^s(\mathcal{B})$ denote the bi-complete and bi-stable labellings of the BAG $\mathcal{B}$.

Given a random forest $\mathcal{F} = \{\mathcal{T}_1, \ldots, \mathcal{T}_t\}$, we want to represent it as a BAG $\mathcal{B}_{\mathcal{F},\mathbf{x}}$ such that the labellings of $\mathcal{B}_{\mathcal{F},\mathbf{x}}$ correspond to the possible inputs and decisions of $\mathcal{F}$. To do so, we first associate a collection of arguments with $\mathcal{F}$.

Definition 3 (Explanation Arguments)**.** The *explanation arguments* $\mathcal{A}_{\mathcal{F}} = \mathcal{A}_{\mathcal{C}} \cup \mathcal{A}_R \cup \mathcal{A}_F$ associated with the random forest $\mathcal{F}$ are defined as follows:

- $\mathcal{A}_{\mathcal{C}} = \{A_y \mid y \in \mathcal{C}\}$ contains one *class arguments* for every class,
- $\mathcal{A}_R = \bigcup_{\mathcal{T} \in \mathcal{F}} \mathcal{A}_{\mathcal{T}}$, where $\mathcal{A}_{\mathcal{T}} = \{A_{\mathcal{T},r} \mid r \in \mathcal{T}\}$ contains a *rule argument* $A_{\mathcal{T},r}$ for every tree $\mathcal{T}$ in $\mathcal{F}$ and every rule $r \in \mathcal{T}$,
- $\mathcal{A}_F = \bigcup_{i=1}^{n} \mathcal{A}_{X_i}$, contains one *feature argument* for every partitioning set of the feature domain (Def. 1), that is, $\mathcal{A}_{X_i} = \{A_{X_i \in S_{i,1}}, \ldots, A_{X_i \in S_{i,n_i}}\}$.

Next, we explain the attack and support relations in $\mathcal{B}_{\mathcal{F}}$. Intuitively, $\mathcal{B}_{\mathcal{F}}$ is a layered graph with the feature arguments $\mathcal{A}_F$ at the bottom, the rule arguments $\mathcal{A}_R$ in the middle and the class arguments $\mathcal{A}_{\mathcal{C}}$ at the top. Attack edges occur only within the feature layer, from the feature to the rule and from the rule to the class layer. Support edges occur only from the rule to the class layer.

Definition 4 (Explanation Argument Relationships)**.** The attack and support relationships $\mathrm{Att}_{\mathcal{F}} = \mathrm{Att}_{F,F} \cup \mathrm{Att}_{F,R} \cup \mathrm{Att}_{R,\mathcal{C}}$ and $\mathrm{Sup}_{\mathcal{F}} = \mathrm{Sup}_{R,\mathcal{C}}$ associated with the random forest $\mathcal{F}$ are defined as follows:

- $\mathrm{Att}_{F,F}$ contains a *feature-feature-attack* between all feature arguments that belong to the same feature. That is, $(A_{f_1}, A_{f_2}) \in \mathrm{Att}_{F,F}$ if and only if $A_{f_1}, A_{f_2} \in \mathcal{A}_{X_i}$,
- $\mathrm{Att}_{F,R}$ contains a *feature-rule-attack* $(A_{X_i \in S_{i,j}}, A_{\mathcal{T},r})$ if the feature constraint $X_i \in S_{i,j}$ is inconsistent with a feature literal $L \in \mathrm{prem}(r)$ (e.g., the feature constraint $X \in (1, 3]$ is inconsistent with the feature literal $X > 6$),
- $\mathrm{Att}_{R,\mathcal{C}}$ contains a *rule-class-attack* $(A_{\mathcal{T},r}, A_y)$ for every rule argument $A_{\mathcal{T},r}$ with $\mathrm{conc}(r) \neq y$,
- $\mathrm{Sup}_{R,\mathcal{C}}$ contains a *rule-class-support* $(A_{\mathcal{T},r}, A_y)$ for every rule argument $A_{\mathcal{T},r}$ with $\mathrm{conc}(r) = y$.

Intuitively, feature-feature attacks guarantee that only one feature argument per feature can be accepted (because they refer to distinct feature values/ranges). Feature-rule attacks deactivate rules that are inconsistent with the currently accepted feature configuration. The rule-class relationships support/attack classes according to their claim. The Explanation BAG associated with a random forest is then constructed from the explanation arguments and the attack and support relationships between them.

Definition 5 (Explanation BAG)**.** Given a random forest $\mathcal{F}$, the *explanation BAG* for $\mathcal{F}$ is the BAG $\mathcal{B}_{\mathcal{F}} = (\mathcal{A}_{\mathcal{F}}, \mathrm{Att}_{\mathcal{F}}, \mathrm{Sup}_{\mathcal{F}})$.

We note that $\mathcal{B}_{\mathcal{F}}$ can be constructed in quadratic time. The reason for the quadratic blowup is that we have pairwise attacks between feature arguments for the same feature.

Proposition 2. *$\mathcal{B}_{\mathcal{F}}$ can be generated from $\mathcal{F}$ in quadratic time.*

4.1 Faithfulness of the Explanation BAG

As we show next, the explanation BAG $\mathcal{B}_{\mathcal{F}}$ is a faithful representation of $\mathcal{F}$ in the following sense: every bi-stable labelling of $\mathcal{B}_{\mathcal{F}}$ represents a possible decision made by $\mathcal{F}$ (*correctness*) and for every possible decision that $\mathcal{F}$ can make, there is a bi-stable labelling of $\mathcal{B}_{\mathcal{F}}$ that represents it (*completeness*). The following lemma motivates the use of bi-stable labellings.

Lemma 1. *Let L be a labelling for $\mathcal{B}_{\mathcal{F}}$.*

1. If $L \in \mathcal{L}^c(\mathcal{B}_{\mathcal{F}})$, then for all features X_i, either

- $L(A_{X_i \in S_{i,j}}) = \text{und}$ *for all* $A_{X_i \in S_{i,j}} \in \mathcal{A}_{X_i}$ *or*
- $L(A_{X_i \in S_{i,j}}) = \text{in}$ *for exactly one* $A_{X_i \in S_{i,j}} \in \mathcal{A}_{X_i}$ *and* $L(A_{X_i \in S_{i,j'}}) = \text{out}$ *for all other* $A_{X_i \in S_{i,j'}} \in \mathcal{A}_{X_i} \setminus \{A_{X_i \in S_{i,j}}\}$.

2. *If* $L \in \mathcal{L}^s(\mathcal{B}_\mathcal{F})$*, then for all features* X_i*,* $L(A_{X_i \in S_{i,j}}) = \text{in}$ *for exactly one* $A_{X_i \in S_{i,j}} \in \mathcal{A}_{X_i}$ *and* $L(A_{X_i \in S_{i,j'}}) = \text{out}$ *for all other* $A_{X_i \in S_{i,j'}} \in \mathcal{A}_{X_i} \setminus \{A_{X_i \in S_{i,j}}\}$.

Lemma 1 states that bi-complete labellings either accept exactly one feature constraint per feature or remain undecided. Since the undecided case is not interesting for our purposes, we focus on bi-stable labellings. The fact that bi-stable labellings accept exactly one constraint per feature allows us to associate every bi-stable labelling L of $\mathcal{B}_\mathcal{F}$ with an equivalence class $S_L \in \mathcal{D}/\equiv_\mathcal{F}$ of inputs with respect to the indistinguishability relation (Def. 2).

As we show next, every bi-stable labelling L accepts exactly one rule argument per tree. This rule argument corresponds to the active path in the tree for all inputs $\mathbf{x} \in S_L$ in the corresponding equivalence class $S_L \in \mathcal{D}/\equiv_\mathcal{F}$.

Lemma 2. *If* $L \in \mathcal{L}^s(\mathcal{B}_\mathcal{F})$*, then for all trees* $\mathcal{T} \in \mathcal{F}$*,* $A_{\mathcal{T},r} \in \mathcal{A}_\mathcal{T}$ *is labelled* in *if and only if* r *is the active rule in* $\mathcal{T}$ *for all inputs* $\mathbf{x} \in S_L$*. Furthermore, all other rule arguments in* $\mathcal{A}_\mathcal{T}$ *are labelled out.*

We can now show that our enconding is correct in the sense that a class argument A_y can be labelled in by L if and only if $\text{O}_\mathcal{F}(\mathbf{x}) = y$ for all $\mathbf{x} \in S_L$.

Proposition 3 (Correctness)**.** *If* $L \in \mathcal{L}^s(\mathcal{B}_\mathcal{F})$*, then for all class arguments* $A_y \in \mathcal{A}_\mathcal{C}$*,* $L(A_y) = \text{in}$ *if and only if* $\text{O}_\mathcal{F}(\mathbf{x}) = y$ *for all* $\mathbf{x} \in S_L$*. Furthermore, if* $L(A_y) = \text{in}$*, then* $L(A_{y'}) = \text{out}$ *for all* $A_{y'} \in \mathcal{A}_\mathcal{C}$*.*

Proposition 3 guarantees that every bi-stable labelling L represents a collection of inputs from the equivalence class $S_L \in \mathcal{D}/\equiv_\mathcal{F}$ and the accepted class-arguments corresponds to their classification. However, it is also possible that L does not accept any class argument. As we show next, this is only possible if the inputs in $S_L \in \mathcal{D}/\equiv_\mathcal{F}$ are ambiguous. Since all inputs in an equivalence class are classified equally, this is equivalent to showing that for every input with $\text{O}_\mathcal{F}(\mathbf{x}) \neq \bot$, there is a corresponding bi-stable labelling $L_\mathbf{x}$ that represents it. $L_\mathbf{x}$ is defined as follows:

1. a feature argument $A_{X_i \in S_{i,j}} \in \mathcal{A}_F$ is labelled in if $X_i \in S_{i,j}$ and labelled out otherwise,
2. a rule argument $A_{\mathcal{T},r} \in \mathcal{A}_R$ is labelled in if r is the active rule in $\mathcal{T}$ for $\mathbf{x}$ and labelled out otherwise,
3. a class argument is labelled in if its supporters dominate its attackers, out if its attackers dominate its supporters, and und otherwise.

The following lemma explains that $L_\mathbf{x}$ is always a bi-complete labelling (Item 1) and accepts at most one class argument (Item 2).

Lemma 3. *1. For all* $\mathbf{x} \in \mathcal{D}$*,* $L_\mathbf{x} \in \mathcal{L}^c(\mathcal{B}_\mathcal{F})$ *.*

2. *There is at most one* $y \in \mathcal{C}$ *such that* $L_\mathbf{x}(A_y) = \text{in}$*. Furthermore, if* $L_\mathbf{x}(A_y) = \text{in}$ *for some* $y \in \mathcal{C}$*, then* $L_\mathbf{x}(A_{y'}) = \text{out}$ *for all* $y' \in \mathcal{C} \setminus \{y\}$*.*

We can now show that our encoding is complete in the sense that $L_\mathbf{x}$ is a bi-stable labelling ($\mathbf{x}$ is represented by a bi-stable labelling) if and only if $\text{O}_\mathcal{F}(\mathbf{x}) \neq \bot$.

Proposition 4 (Completeness)**.** *For all inputs* $\mathbf{x} \in \mathcal{D}$*,* $\text{O}_\mathcal{F}(\mathbf{x}) \neq \bot$ *if and only if* $L_\mathbf{x} \in \mathcal{L}^s(\mathcal{B}_\mathcal{F})$*.*

4.2 Applications of the Explanation BAG

Now that we established the formal relationship between $\mathcal{B}_\mathcal{F}$ and $\mathcal{F}$, we can use it to reduce questions about $\mathcal{F}$ to argumentation problems in $\mathcal{B}_\mathcal{F}$. To begin with, we note that counting the ambiguous equivalence classes of $\mathcal{F}$ can be reduced to counting the bi-stable labellings of $\mathcal{B}_\mathcal{F}$.

Proposition 5. $|\text{Amb}(\mathcal{F})/\equiv_\mathcal{F}| = |\mathcal{D}| - |\mathcal{L}^s(\mathcal{B}_\mathcal{F})|$.

Two interesting argumentative reasoning problems that are relevant for explainable AI are finding sufficient and necessary reasons for the acceptance of arguments (Borg and Bex 2021). A set of arguments S is a *sufficient reason* for an argument A if for all labellings L, A is accepted by L whenever S is accepted by L. A set of arguments N is a *necessary reason* for A if L accepts A only if it also accepts N. We will consider sufficient and necessary reasons with respect to bi-stable labellings here. Note that a set of feature arguments $\{A_{X_{i_1} \in S_{i_1,j_1}}, \ldots, A_{X_{i_k} \in S_{i_k,j_k}}\}$ is a (minimal) sufficient reason for a class argument A_y in $\mathcal{B}_\mathcal{F}$ if and only if every partial assignment from $S_{i_1,j_1} \times \cdots \times S_{i_k,j_k}$ is a wAXp (AXp) for y in $\mathcal{F}$.

Example 3. *For the explanation BAG corresponding to Figure 1, the set of feature arguments* $\{A_{B \in \{1\}}, A_{Age \in (-\infty,35]}\}$ *is a minimal sufficient reason for* A_{Pos}*. This means that every partial assignment of the form* $(B = 1, Age = x)$*, where* $x \leq 35$*, is an AXp for the random forest.*

Similarly, if $\{A_{X_{i_1} \in S_{i_1,j_1}}, \ldots, A_{X_{i_k} \in S_{i_k,j_k}}\}$ is a necessary reason for A_y, then $\mathcal{F}$ can only classify an input as y if the input is an extension of one of the partial assignments from $S_{i_1,j_1} \times \cdots \times S_{i_k,j_k}$.

Example 4. *For Figure 1, the feature argument* $A_{A \in \{0\}}$ *is necessary for* A_{Neg} *because if* $A = 1$*, the first tree will vote for* Pos*, so that the output of* $\mathcal{F}$ *is either* Pos *or* $\bot$*.*

The following proposition allows us to construct necessary feature arguments bottom-up.

Proposition 6. *If* $N \subseteq \mathcal{A}_\mathcal{F}$ *is necessary for* A_y*, then all* $A \in N$ *are necessary for* A_y*.*

This suggests the following algorithm for finding all necessary feature arguments. For every class argument A_y and every feature argument $A_{X_i \in S_{i,j}}$, test if $A_{X_i \in S_{i,j}}$ is necessary for A_y. The union of all these feature arguments is then the maximal necessary reason among the feature arguments and we can find it with a linear number of atomic necessity checks. However, deciding if a feature argument is necessary for a class argument, may be a difficult problem itself. The problem is in $CoNP$ because a counterexample for the necessity of a candidate can be verified efficiently, but we currently do not know a lower bound for the complexity.

5 Markov Network Representation

We can reduce many combinatorial tasks in argumentation graphs to probabilistic queries in Markov networks (Potyka 2020). The reduction also allows us to generalize the idea of necessary and sufficient reasons to δ-sufficient and δ-necessary reasons similar to the idea of δ-AXps.

Intuitively, Markov networks decompose a large probabilistic model P into smaller local models (Koller and Friedman 2009). We denote random variables by capital letters U, V, W and values of these random variables by small letters u, v, w. Bold capital letters $\mathbf{U}, \mathbf{V}, \mathbf{W}$ denote ordered sequences of random variables and bold small letters $\mathbf{u}, \mathbf{v}, \mathbf{w}$ denote assignments to these random variables. For example, if $\mathbf{U} = (U_1, U_2, U_3)$ and $\mathbf{u} = (u_1, u_2, u_3)$, then $\mathbf{U} = \mathbf{u}$ denotes the assignment $(U_1 = u_1, U_2 = u_2, U_3 = u_3)$. We write $\mathbf{V} \subseteq \mathbf{U}$ if the random variables in $\mathbf{V}$ form a subset of the random variables in $\mathbf{U}$. If $\mathbf{V} \subseteq \mathbf{U}$, we denote by $\mathbf{U}|_{\mathbf{V}}$ and $\mathbf{u}|_{\mathbf{V}}$ the restriction of $\mathbf{U}$ and $\mathbf{u}$ to the random variables in $\mathbf{V}$. For example, if $\mathbf{V} = (U_1, U_3)$, we have $\mathbf{U}|_{\mathbf{V}} = (U_1, U_3)$ and $\mathbf{u}|_{\mathbf{V}} = (u_1, u_3)$. We consider three types of random variables in our application.

Definition 6 (Explanation Random Variables)**.** The random variables associated with $\mathcal{F}$ are defined as follows:

- For every feature X_i, we introduce a *feature variable* U_i that can take values from $\{S_{i,j_1}, \ldots, S_{i,n_i}\}$ (the partitioning sets of the feature domain from Def. 1).
- For every tree $\mathcal{T} = \{r_1, \ldots, r_k\}$, we introduce a *tree variable* $U_{\mathcal{T}}$ that can take values from $\{r_1, \ldots, r_k\}$.
- We introduce a *class variable* $U_{\mathcal{C}}$ that can take values from $\mathcal{C}$.

A *factor* with scope $\mathbf{V} \subseteq \mathbf{U}$ is a function $\phi(\mathbf{V})$ that maps every assignment $\mathbf{v}$ to $\mathbf{V}$ to a non-negative real number. Intuitively, factors can increase or decrease the probability of variable assignments. Given a set of factors $\Phi = \{\phi_1(\mathbf{U}_1), \ldots, \phi_k(\mathbf{U}_k)\}$, $\mathbf{U}_i \subseteq \mathbf{U}$, we define the *plausibility of a state* of $\mathbf{U}$ via

$$\mathrm{Pl}_\Phi(\mathbf{U}) = \prod_{i=1}^{k} \phi_i(\mathbf{U}|_{\mathbf{U}_i}).$$

By normalizing the plausibility, we obtain a probability distribution that is called the *Gibbs distribution* over $\mathbf{U}$:

$$P_\Phi(\mathbf{U}) = \frac{1}{Z}\mathrm{Pl}_\Phi(\mathbf{U}),$$

where the normalization constant $Z = \sum_{\mathbf{u}} \mathrm{Pl}_\Phi(\mathbf{u})$ guarantees that the probabilities add up to 1. Z is also called the *partition function*.

In our application, we build up the Gibbs distribution from two types of factors. Intuitively, the first one simulates the individual tree decisions based on the state of the feature constraints and the second one simulates the decision making process of the random forest based on the tree decisions.

Definition 7 (Explanation Factors)**.** The factors associated with $\mathcal{F}$ are defined as follows:

- For every tree $\mathcal{T} \in \mathcal{F}$, there is a tree factor $\phi_{\mathcal{T}}(\mathbf{U}_{\mathcal{T}})$, where $\mathbf{U}_{\mathcal{T}}$ contains the tree variable $U_{\mathcal{T}}$ and for each feature X_i used in $\mathcal{T}$, the corresponding feature variable U_i. $\phi_{\mathcal{T}}(\mathbf{U}_{\mathcal{T}})$ is a tree-factor (Koller and Friedman 2009) defined as follows: given a variable assignment $\mathbf{u}_{\mathcal{T}}$, $\phi_{\mathcal{T}}(\mathbf{U}_{\mathcal{T}})$ computes the active rule r for the assignment of the feature variables and returns 1 if r is assigned to $U_{\mathcal{T}}$ and 0 otherwise.
- There is one class factor $\phi_{\mathcal{C}}(\mathbf{U}_{\mathcal{C}})$, where $\mathbf{U}_{\mathcal{C}}$ contains the class variable $U_{\mathcal{C}}$ and all tree variables. $\phi_{\mathcal{C}}(\mathbf{U}_{\mathcal{C}})$ is defined as a deterministic factor (Koller and Friedman 2009) defined as follows: Given a variable assignment $\mathbf{u}_{\mathcal{C}}$, $\phi_{\mathcal{C}}(\mathbf{U}_{\mathcal{C}})$ iterates over the tree variables and counts for every class the number of rules that vote for the class. It then returns 1 if the class assigned to $U_{\mathcal{C}}$ has a larger number of votes than all other classes and 0 otherwise.

The factors define the explanation plausibility distribution and the corresponding Gibbs distribution for $\mathcal{F}$.

Definition 8. Given a random forest $\mathcal{F}$, the associated *explanation plausibility distribution* for $\mathcal{F}$ is

$$\mathrm{Pl}_{\mathcal{F}}(\mathbf{U}) = \phi_{\mathcal{C}}(\mathbf{U}|_{\mathbf{U}_{\mathcal{C}}}) \cdot \prod_{\mathcal{T} \in \mathcal{F}} \phi_{\mathcal{T}}(\mathbf{U}|_{\mathbf{U}_{\mathcal{T}}})$$

and the *explanation Gibbs distribution* is

$$P_{\mathcal{F}}(\mathbf{U}) = \frac{1}{Z}\mathrm{Pl}_{\mathcal{F}}(\mathbf{U}).$$

Although $P_{\mathcal{F}}(\mathbf{U})$ is motivated by the explanation BAG, we can construct it immediately from $\mathcal{F}$. To do this, we traverse all trees to create the domains of the random variables, translate the decision trees into tree factors and create the class factor. This can almost be done in linear time, but as we need to order the threshold values of continuous features for the domain partition, there can be a log-linear blowup. As the explanation plausibility distribution is just the product of the factors, we can generate it in log-linear time.

Proposition 7. *The explanation plausibility distribution* $\mathrm{Pl}_{\mathcal{F}}(\mathbf{U})$ *can be generated from* $\mathcal{F}$ *in log-linear time.*

Building up the Gibbs distribution probably requires exponential time as it involves computing the normalization constant Z. However, we will exploit the fact that the plausibility distribution can be used to design sampling algorithms to approximate Z and queries to the Gibbs distribution.

5.1 Explanation Queries

Before going into the sampling algorithms, let us explain what we can learn from the normalization constant and probabilities from the Gibbs distribution. We keep exploiting the fact that bi-stable labellings correspond to non-ambiguous inputs for $\mathcal{F}$ (Proposition 4). To do so, we associate every input $\mathbf{u}$ for $P_{\mathcal{F}}(\mathbf{U})$ with a labelling $L_{\mathbf{u}}$ as follows:

1. a feature argument $A_{X_i \in S_j^i} \in \mathcal{A}_F$ is labelled in if $U_i = S_j^i$ and labelled out otherwise,
2. a rule argument $A_{\mathcal{T},r} \in \mathcal{A}_R$ is labelled in if $U_{\mathcal{T}} = r$ and labelled out otherwise,
3. a class argument A_y is labelled in if $U_{\mathcal{C}} = y$ and labelled out otherwise.

Let us first observe that the plausibility of every input for $P_{\mathcal{F}}(\mathbf{U})$ is either 0 or 1 and it is non-zero if and only if it represents a bi-stable labelling of the explanation BAG.

Proposition 8. *For every assignment* $\mathbf{u}$ *to* $\mathrm{Pl}_{\mathcal{F}}(\mathbf{U})$*, we have* $\mathrm{Pl}_{\mathcal{F}}(\mathbf{u}) \in \{0, 1\}$*. Furthermore,* $\mathrm{Pl}_{\mathcal{F}}(\mathbf{u}) \neq 0$ *if and only if* $L_{\mathbf{u}}$ *is a bi-stable labelling of the explanation BAG.*

This relationship allows us to connect the partition function Z to the number of bi-stable labellings (non-ambiguous inputs) and probabilistic queries to generalizations of sufficient and necessary reasons. We say that a set of arguments S is a *δ-sufficient reason* for an argument A if among the labellings that accept S, $\delta \cdot 100$ % also accept A. Similarly, N is a *δ-necessary reason* for A if among the labellings that accept A, $\delta \cdot 100$ % also accept S. Note that 1-sufficient (1-necessary) reasons are just sufficient (necessary) reasons. Furthermore, if all features are categorical, then $\{A_{X_{i_1} \in S_{i_1,j_1}}, \ldots, A_{X_{i_k} \in S_{i_k,j_k}}\}$ is a δ-sufficient reason for a class argument A_y in $\mathcal{B}_{\mathcal{F}}$ if and only if every partial assignment from $S_{i_1,j_1} \times \cdots \times S_{i_k,j_k}$ is a δ-AXp for y in $\mathcal{F}$. If we have continuous features, this may not be the case for $\delta \neq 1$ because the indistinguishability relation does not necessarily partition the domains of continuous features into equivalence classes of equal size.

In the next proposition, we use the following notation: Given an assignment $\mathbf{v}_F$ to a subsequence of feature random variables $\mathbf{V}_F$, we let $S_{\mathbf{v}_F}$ denote the corresponding set of feature arguments that contains the feature argument $A_{X_i \in S^i_j}$ if and only if $\mathbf{v}_F$ assigns $U_i = S^i_j$.

Proposition 9. *1. If all features are categorical, then* $Z = |\mathcal{D}| - |\mathrm{Amb}(\mathcal{F})| = |\mathcal{L}^s(\mathcal{B}_{\mathcal{F}})|$ *is the number of equivalence classes of non-ambiguous inputs for* $\mathcal{F}$*.*

2. *Let* $\mathbf{V}_F$ *be a subsequence of feature random variables. Then* $\mathrm{Pl}_{\mathcal{F}}(C = y, \mathbf{v}_F) = \frac{N_{(C=y,\mathbf{v}_F)}}{Z}$*, where* $N_{(C=y,\mathbf{v}_F)}$ *is the number of bi-stable labellings that accept all arguments in* $S_{\mathbf{v}_F} \cup \{A_y\}$*.*
3. $\mathrm{P}_{\mathcal{F}}(C = y \mid \mathbf{v}_F) = \delta$ *if and only if* $S_{\mathbf{v}_F}$ *is a δ-sufficient reason for* A_y*.*
4. $\mathrm{P}_{\mathcal{F}}(\mathbf{v}_F \mid C = y) = \delta$ *if and only if* $S_{\mathbf{v}_F}$ *is a δ-necessary reason for* A_y*.*

5.2 A Probabilistic Approximation Algorithm

Proposition 1 and the complexity results for deciding AXps and δ-AXps from (Izza and Marques-Silva 2021) and (Wäldchen et al. 2021) make it unlikely that there is an efficient exact algorithm for computing the partition function and the probabilities in Proposition 9. We therefore consider a probabilistic algorithm that approximates the probabilities. Readers familiar with Bayesian networks may notice that $P_{\mathcal{F}}$ is almost a Bayesian network: The variable factors are independent of all other factors, the tree factors depend only on the variable factors and the class factor only on the tree factors. The dependency structure of the factors in $P_{\mathcal{F}}$ is therefore acyclic like in a Bayesian network. However, the class factor cannot be interpreted as a conditional probability distribution because it does not define a probability distribution when the configuration of the tree factors corresponds to an ambiguous input. Nevertheless, the acyclic structure allows us to use forward sampling ideas for Bayesian networks (Koller and Friedman 2009) to approximate sufficient and necessary queries.

Input: *rand. forest* $\mathcal{F}$*, queries* $(\mathbf{w}_1 \mid \mathbf{v}_1), \ldots, (\mathbf{w}_l \mid \mathbf{v}_l)$
Output: *estimates for* $\mathrm{P}_{\mathcal{F}}(\mathbf{w}_1 \mid \mathbf{v}_1), \ldots, \mathrm{P}_{\mathcal{F}}(\mathbf{w}_l \mid \mathbf{v}_l)$

DO:
| $E \leftarrow sampleEquivalenceClass(\mathcal{F})$
| **IF** $\mathrm{O}_{\mathcal{F}}(E) \neq \bot$:
| | $countNonambiguous()$
| | $\mathbf{u} \leftarrow computeAssignment(E)$
| | **FOR** $i = 1$ **TO** k :
| | | **IF** $\mathbf{u}|_{\mathbf{V}_i} = \mathbf{v}_i$*:*
| | | | **IF** $\mathbf{u}|_{\mathbf{W}_i} = \mathbf{w}_i$*:* $countPos(\mathbf{w}_i, \mathbf{v}_i)$
| | | | **ELSE:** $countNeg(\mathbf{w}_i, \mathbf{v}_i)$
| **ELSE:** $countAmbiguous()$
WHILE *termination condition not met*
RETURN $estimates()$

Figure 2: Probabilistic approximation algorithm for estimating the percentage of non-ambiguous inputs, and the probabilities of sufficient and necessary queries.

Figure 2 shows the template of our algorithm. It expects as input a random forest and the probabilistic queries that are to be approximated. The queries consist of sufficient queries (item 2) or necessary queries (item 3) in Proposition 1. The algorithm uses forward sampling (from the feature variables to the class variable). It repeatedly samples equivalence classes of inputs for $\mathcal{F}$. Since the tree and class factors are deterministic, the state of the tree and class variables is already determined by this sample and their state is only computed if needed. With a slight abuse of notation, we write $\mathrm{O}_{\mathcal{F}}(E)$ for $\mathrm{O}_{\mathcal{F}}(e)$, where $e \in E$ is an arbitrary input from the equivalence class E (recall that all inputs in E are indistinguishable for $\mathcal{F}$). Ambiguous samples are rejected immediately, but we keep track of their number ($countAmbiguous()$). For non-ambiguous samples, we also iterate a counter ($countNonambiguous()$) and complete the variable assignment. The completed samples are used to approximate the queries by relative frequencies. More precisely, $countPos(\mathbf{w}_i, \mathbf{v}_i)$ and $countNeg(\mathbf{w}_i, \mathbf{v}_i)$ increment counters $N^+_{(\mathbf{w}_i,\mathbf{v}_i)}$ or $N^-_{(\mathbf{w}_i,\mathbf{v}_i)}$ that count how often the target $\mathbf{w}_i$ was satisfied or not when the condition $\mathbf{v}_i$ was satisfied. The estimate for the conditional probability $P_{\mathcal{F}}(\mathbf{w}_i \mid \mathbf{v}_i)$ is $\frac{N^+_{(\mathbf{w}_i,\mathbf{v}_i)}}{N^+_{(\mathbf{w}_i,\mathbf{v}_i)}+N^-_{(\mathbf{w}_i,\mathbf{v}_i)}}$. The estimate for the percentage of non-ambiguos input equivalence classes is $\frac{N_n}{N_n+N_a}$, where N_a (N_n) is the numbers of (non-)ambiguous equivalence classes that we sampled. Multiplying this fraction by the number of all equivalence classes results in an

estimate for the number of non-ambiguos input equivalence classes, but we restrict to reporting the percentage as it is easier to comprehend. We have the following guarantees, where *convergence in probability* means that the probability that the estimates deviate from the target by more than an arbitrarily small ϵ goes to 0 as the number of samples goes to ∞.

Proposition 10. *When sampling inputs uniformly and independently in the algorithm in Figure 2, then $N_n/(N_n + N_a)$ converges in probability to the percentage of non-ambiguous equivalence classes and $e_{(\mathbf{w}_i|\mathbf{v}_i)} = N^+_{(\mathbf{w}_i,\mathbf{v}_i)}/(N^+_{(\mathbf{w}_i,\mathbf{v}_i)} + N^-_{(\mathbf{w}_i,\mathbf{v}_i)})$ to $\mathrm{P}_{\mathcal{F}}(\mathbf{w}_i \mid \mathbf{v}_i)$ for $1 \leq i \leq l$. Every iteration runs in linear time with respect to $\mathcal{F}$ and the number of queries l. Furthermore, if we have $M \geq \frac{3\ln(2/\delta)}{P(\mathbf{w}_i|\mathbf{v}_i)\cdot\epsilon^2}$ samples for $e_{(\mathbf{w}_i|\mathbf{v}_i)}$, then $P(e_{(\mathbf{w}_i|\mathbf{v}_i)} \in \mathrm{P}_{\mathcal{F}}(\mathbf{w}_i \mid \mathbf{v}_i) \cdot (1 \pm \epsilon)) \geq 1 - \delta$.*

Let us note that even though every iteration of our algorithm runs in linear time, we may require a large number of iterations until the estimates converge. The probabilistic error bound at the end of Proposition 10 shows that the convergence speed depends on the number of samples generated for $e_{(\mathbf{w}_i|\mathbf{v}_i)}$. If $P_{\mathcal{F}}(\mathbf{v}_i)$ is small, this will take longer. Typically, the estimates for necessary queries (conditioned on a class label) and sufficient queries for shorter abductive explanations will converge faster.

Formally, we simultaneously approximate the percentage of non-ambiguous inputs using Monte-Carlo sampling and the probabilities of the queries using rejection sampling (Koller and Friedman 2009). Every sample that we create in our algorithm can be used for the Monte-Carlo approximation, but only a fraction for individual rejection samples. It can be wasteful not to use the Monte-Carlo samples for the queries. However, once a sufficiently large number of samples has been created for the Monte-Carlo approximation, we can switch from rejection sampling to conditional forward sampling. That is, if $e_{(\mathbf{w}_i|\mathbf{v}_i)}$ requires additional samples, we fix the state of the variables $\mathbf{v}_i$ and sample only the remaining feature variables, which is justified by the acyclic dependency structure of the factors in $P_{\mathcal{F}}$ that we explained at the beginning of this section.

5.3 Implementation and Experiments

As a first proof of concept, we implemented a simple variant of the algorithm in Figure 2 in Python[1]. We consider a reason $\mathbf{v}$ *almost sufficient* for $C = y$ if it is δ-sufficient and $P_{\mathcal{F}}(C = y \mid \mathbf{v}) > 1.1 \cdot P_{\mathcal{F}}(C = y)$, that is, $\mathbf{v}$ results in a relative increase of the probability of at least 10 %. We consider $\mathbf{v}$ *almost necessary* if it is δ-necessary. In this case, we do not need to take the prior into account because we sample uniformly from the partition domains (the prior is therefore always at most 0.5). We chose $\delta = 0.9$.

Our implementation works in two stages. The first stage is analogous to Figure 2 and the queries are the atomic sufficient and necessary queries of the form $(U_y \mid U_i)$ and $(U_i \mid U_y)$ for all combinations of feature arguments U_i and class arguments U_y. At the end of stage 1, we report the estimated percentage of non-ambiguous inputs and all almost sufficient and necessary reasons that were found. We can combine all almost necessary reasons to a single big necessary reason for reasons similar to Proposition 6. However, there may be many more almost sufficient reasons. Therefore, in the second stage, the algorithm tries to find almost sufficient reasons of size 2. To this end, for all pairs of features (U_i, U_j), and all possible assignments (u_i, u_j) of equivalence classes to these features, we perform forward sampling conditioned on the feature assignment (u_i, u_j) and report the estimate if the probability exceeds the δ-threshold. For every pair (u_i, u_j), the probability can be estimated quickly. However, since there can be a large number of pairs, the overall runtime can be long and the almost sufficient reasons of size 2 are reported continuously while the sampling procedure is running.

We tested our algorithm on three datasets. The Iris and PIMA dataset are continuous datasets that have been considered for counterfactual explanations (White and d'Avila Garcez 2020). In addition, we consider the Mushroom dataset that contains discrete features. For reproducibility, the datasets are contained in the source folder. For the random forest trained on the Iris dataset, the estimated percentage of non-ambiguous input equivalence classes is 98 % and we found several almost sufficient reasons. These included 1-sufficient reasons. The estimates are based on several hundred examples, but since there is uncertainty in the sampling process, we should be careful and assume that these are δ-sufficient for δ close to 1, but not necessarily equal to 1. $\mathrm{petallength} \in (5.0, 5.14]$ is an example of an almost sufficient reason of length 1 for the class $\mathrm{Virginica}$. The pair $(\mathrm{sepallength} \in (5.45, 5.5], \mathrm{petallength} \in (2.64, 2.75])$ is an almost sufficient reason of length 2 for the class $\mathrm{Versicolor}$. We generated 10,000 samples for the first stage in less than one minute on a Windows laptop with i7-11800H CPU and 16 GB RAM. The second stage produced a variety of other sufficient reasons within seconds, but many redundant reasons are reported in the current version. For the Mushroom dataset, we found that our random forest learnt that $\mathrm{Odor\_Foul} = 1$ is 0.99-sufficient for $\mathrm{Poisonous}$ and $\mathrm{Odor\_Foul} = 0$ is 0.98-necessary for Edible. We provide more details and examples about the experiments and how to reproduce them in the appendix of (Potyka, Yin, and Toni 2022).

6 Conclusions

We showed that the decision process of random forests can be naturally encoded as a bipolar argumentation problem. This allows reducing counting the number of ambiguous inputs and finding sufficient and necessary reasons to reasoning tasks over argumentation problems. The argumentation problems are often solved using reductions to SAT (Dvorák et al. 2012; Beierle, Brons, and Potyka 2015; Alviano 2018), CSP (Lagniez, Lonca, and Mailly 2015) or Markov networks (Potyka 2020). We used a Markov network reduction as it naturally leads to almost sufficient and almost necessary reasons and a variety of algorithms with probabilistic approximation guarantees.

[1]https://github.com/nicopotyka/Uncertainpy, folder *examples/explanations/randomForests*. See appendix in (Potyka, Yin, and Toni 2022) for more details.

Acknowledgments

This research was partially funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 101020934, ADIX) and by J.P. Morgan and by the Royal Academy of Engineering under the Research Chairs and Senior Research Fellowships scheme. Any views or opinions expressed herein are solely those of the authors.

References

Alviano, M. 2018. The pyglaf argumentation reasoner. In *International Conference on Logic Programming (ICLP)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Amgoud, L.; Cayrol, C.; Lagasquie-Schiex, M.-C.; and Livet, P. 2008. On bipolarity in argumentation frameworks. *International Journal of Intelligent Systems*, 23(10): 1062–1093.

Beierle, C.; Brons, F.; and Potyka, N. 2015. A software system using a SAT solver for reasoning under complete, stable, preferred, and grounded argumentation semantics. In *Joint German/Austrian Conference on Artificial Intelligence (KI)*, 241–248. Springer.

Boella, G.; Gabbay, D. M.; van der Torre, L.; and Villata, S. 2010. Support in abstract argumentation. In *International Conference on Computational Models of Argument (COMMA)*, 40–51. Frontiers in Artificial Intelligence and Applications, IOS Press.

Borg, A.; and Bex, F. 2021. Necessary and Sufficient Explanations for Argumentation-Based Conclusions. In Vejnarová, J.; and Wilson, N., eds., *European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, volume 12897 of *LNCS*, 45–58. Springer.

Breiman, L. 2001. Random forests. *Machine learning*, 45(1): 5–32.

Cayrol, C.; and Lagasquie-Schiex, M.-C. 2013. Bipolarity in argumentation graphs: Towards a better understanding. *International Journal of Approximate Reasoning*, 54(7): 876–899. Publisher: Elsevier.

Cyras, K.; Rago, A.; Albini, E.; Baroni, P.; and Toni, F. 2021. Argumentative XAI: A Survey. In Zhou, Z., ed., *International Joint Conference on Artificial Intelligence (IJCAI)*, 4392–4399. ijcai.org.

Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial intelligence*, 77(2): 321–357. Publisher: Elsevier.

Dvorák, W.; Järvisalo, M.; Wallner, J. P.; and Woltran, S. 2012. Cegartix: A sat-based argumentation system. In *Pragmatics of SAT Workshop (POS)*.

Huang, X.; Izza, Y.; Ignatiev, A.; Cooper, M. C.; Asher, N.; and Marques-Silva, J. 2022. Tractable Explanations for d-DNNF Classifiers. In *AAAI Conference on Artificial Intelligence (AAAI)*, 5719–5728. AAAI Press.

Ignatiev, A.; Izza, Y.; Stuckey, P. J.; and Marques-Silva, J. 2022. Using MaxSAT for Efficient Explanations of Tree Ensembles. In *AAAI Conference on Artificial Intelligence (AAAI)*, 3776–3785. AAAI Press.

Izza, Y.; and Marques-Silva, J. 2021. On Explaining Random Forests with SAT. In Zhou, Z., ed., *International Joint Conference on Artificial Intelligence, (IJCAI)*, 2584–2591.

Koller, D.; and Friedman, N. 2009. *Probabilistic graphical models: principles and techniques*. MIT press.

Lagniez, J.-M.; Lonca, E.; and Mailly, J.-G. 2015. Coquiaas: A constraint-based quick abstract argumentation solver. In *International Conference on Tools with Artificial Intelligence (ICTAI)*, 928–935. IEEE.

Lundberg, S. M.; and Lee, S.-I. 2017. A unified approach to interpreting model predictions. In *International Conference on Neural Information Processing Systems (NeurIPS)*, 4768–4777.

Marques-Silva, J.; and Ignatiev, A. 2022. Delivering Trustworthy AI through Formal XAI. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Oren, N.; and Norman, T. J. 2008. Semantics for Evidence-Based Argumentation. In *International Conference on Computational Models of Argument (COMMA)*, 276–284. IOS Press.

Plumb, G.; Molitor, D.; and Talwalkar, A. 2018. Model agnostic supervised local explanations. In *International Conference on Neural Information Processing Systems (NeurIPS)*, 2520–2529.

Potyka, N. 2020. Abstract Argumentation with Markov Networks. In *European Conference on Artificial Intelligence (ECAI)*, 865–872.

Potyka, N. 2021. Generalizing Complete Semantics to Bipolar Argumentation Frameworks. In *European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2021)*, Lecture Notes in Computer Science, 130–143. Springer.

Potyka, N.; Yin, X.; and Toni, F. 2022. Explaining Random Forests using Bipolar Argumentation and Markov Networks (Technical Report). *CoRR*, abs/2211.11699.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. " Why should i trust you?" Explaining the predictions of any classifier. In *ACM SIGKDD international conference on knowledge discovery and data mining*, 1135–1144.

Shih, A.; Choi, A.; and Darwiche, A. 2018. A Symbolic Approach to Explaining Bayesian Network Classifiers. In Lang, J., ed., *International Joint Conference on Artificial Intelligence, IJCAI*, 5103–5111. ijcai.org.

Vassiliades, A.; Bassiliades, N.; and Patkos, T. 2021. Argumentation and explainable artificial intelligence: a survey. *The Knowledge Engineering Review*, 36.

Wachter, S.; Mittelstadt, B.; and Russell, C. 2017. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harv. JL & Tech.*, 31: 841.

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.

White, A.; and d'Avila Garcez, A. S. 2020. Measurable Counterfactual Local Explanations for Any Classifier. In *European Conference on Artificial Intelligence (ECAI))*.