Explainable Global Fairness Verification of Tree-Based Classifiers

Stefano Calzavara*, Lorenzo Cazzaro*, Claudio Lucchese* and Federico Marcuzzi*
*Università Ca' Foscari Venezia, Italy
Email: {name.surname}@unive.it

***Abstract*—We present a new approach to the global fairness verification of tree-based classifiers. Given a tree-based classifier and a set of sensitive features potentially leading to discrimination, our analysis synthesizes sufficient conditions for fairness, expressed as a set of traditional propositional logic formulas, which are readily understandable by human experts. The verified fairness guarantees are global, in that the formulas predicate over all the possible inputs of the classifier, rather than just a few specific test instances. Our analysis is formally proved both sound and complete. Experimental results on public datasets show that the analysis is precise, explainable to human experts and efficient enough for practical adoption.**

I. Introduction

The ever-increasing success of Machine Learning (ML) led to its massive deployment in a range of different settings over the last years. Unfortunately, classic approaches to assess the performance of ML models do not always provide a reliable picture of their effectiveness in so many varied practical deployments. For example, traditional deep learning models with state-of-the-art accuracy may be vulnerable to evasion attacks, i.e., small perturbations of test inputs designed to force prediction errors, thus making their deployment in adversarial settings unfeasible [1], [2]. Similarly, ML models may turn out to be *unfair* for automated decision-making. For example, a commercial recidivism-risk assessment algorithm was found to be racially biased [3] and an existing algorithm adopted in the US falsely determined that black patients were healthier than other patients with similar conditions [4]. These incidents led to a proliferation of research on fair ML in recent times, as summarized in different surveys [5], [6], [7].

Fairness in ML has been analyzed from different angles and can be broadly categorized in two main research lines. The first one includes the development of new ML algorithms that are able to mitigate the bias that is directly or indirectly present in the training data [8], [9], [10]. The second, complementary research line investigates techniques to estimate or even formally verify the fairness guarantees provided by existing ML models [11], [12], [13]. This paper contributes to the latter line of work, which is still at an early stage of development and suffers from relevant shortcomings.

A very popular approach to assess the fairness guarantees of ML models is based on *testing* [14], [15], [16], [17]. The key common intuition underlying any fairness testing strategy is straightforward: generating a number of test inputs designed to automatically identify individuals who may suffer from discrimination by the ML model. Unfortunately, as for any testing approach, this type of analysis is under-approximated: these proposals can identify room for unfair treatment, but cannot establish formal fairness proofs. This is sub-optimal because it does not allow one to prove that unfair behavior can never affect specific classes of individuals. For this reason, recent papers advocated the adoption of formal *fairness verification* techniques to prove lack of discrimination [10], [11], [12], [13], [18]. Most of the work in the area, however, just focuses on deep neural networks and disregards tree-based ML models, such as decision trees [19] and random forests [20], which are still exceptionally popular, in particular for non-perceptual classification tasks. The only notable fairness verification approach designed for tree-based classifiers leverages abstract interpretation to verify *local fairness* properties [9]. Unfortunately, local fairness is now recognized as a rather weak property predicating just on specific test instances, while *global fairness* predicates over all the possible inputs of the classifier and is thus more reliable to assess the actual fairness guarantees that it provides [18].

A. Contributions

We present a new approach to the global fairness verification of tree-based classifiers. Given a tree-based classifier and a set of sensitive features potentially leading to discrimination, our analysis synthesizes sufficient conditions for fairness, expressed as a set of traditional propositional logic formulas F predicating over the entire feature space, rather than just on a specific test set, thus providing global fairness guarantees.

Our fairness verification approach is formally proved both *sound* and *complete*, i.e., fairness is certified for any instance satisfying some formula in F, and the formulas in F can characterize all the instances where the classifier is fair. Moreover, our approach is *explainable*, i.e., it is readily understandable by human experts, being based on traditional logic formulas. In particular, we empirically show that a small set of simple logic formulas suffices to largely characterize the fairness guarantees provided by the classifier in practice. This makes our approach particularly appealing for problems like algorithmic hiring, where automated decisions must be carefully audited [21].

B. Structure of the Paper

The paper is organized as follows:

- Section II reviews the background and introduces the necessary ingredients to appreciate the technical contribution;

- Section III presents our fairness verification approach and establishes the formal guarantees provided by our analysis. Moreover, it describes the implementation of the analysis in C++. The source code of our analyzer is publicly available on Github[1];
- Section IV reports on our experimental evaluation on public datasets. Our experiments assess the precision of the analysis, its explainability, and its performance;
- Section V presents and compares with related work, thus clarifying the distinctive features of our proposal;
- Section VI briefly concludes the paper and describes possible future work in the area.

II. Background

We here introduce the key technical ingredients required to appreciate the contribution of the paper.

A. Tree-Based Classifiers

Let $\mathcal{X} \subseteq \mathbb{R}^d$ be a d-dimensional vector space of features and let $\mathcal{Y}$ be a finite set of class labels. We assume each element of the feature space $\vec{x} = \langle x_1, \ldots, x_d \rangle$, called *instance*, to be assigned a correct class y by an unknown *target* function $g : \mathcal{X} \to \mathcal{Y}$. A *classifier* is a function $f : \mathcal{X} \to \mathcal{Y}$ intended to approximate the target function g as accurately as possible. Normally, f is automatically trained by a supervised learning algorithm, using a *training set* $\mathcal{D}_{train} = \{(\vec{x}_i, g(\vec{x}_i))\}_i$ of correctly labeled instances. The performance of f is then assessed on a separate *test set* $\mathcal{D}_{test} = \{(\vec{z}_i, g(\vec{z}_i))\}_i$, sampled from the same distribution of the training set.

In this paper, we focus on *decision tree* classifiers [19]. Decision trees can be inductively defined as follows: a decision tree t is either a leaf $\lambda(y)$ for some label $y \in \mathcal{Y}$ or an internal node $\sigma(f, v, t_l, t_r)$, where $f \in \{1, \ldots, d\}$ identifies a feature, $v \in \mathbb{R}$ is a threshold for the feature, and t_l, t_r are decision trees (left and right respectively). At test time, the instance $\vec{x}$ traverses the tree t until it reaches a leaf $\lambda(y)$, which returns the prediction y, denoted by $t(\vec{x}) = y$. Specifically, for each traversed tree node $\sigma(f, v, t_l, t_r)$, $\vec{x}$ falls into the left sub-tree t_l if $x_f \leq v$, and into the right sub-tree t_r otherwise. Figure 1 represents an example decision tree of depth two, which assigns the label $+1$ to the instance $\langle 10, 6 \rangle$ and the label -1 to the instance $\langle 6, 9 \rangle$. Decision trees are normally combined into an *ensemble* $T = \{t_1, \ldots, t_n\}$ to improve their predictive power [20]: in this case, the ensemble prediction $T(\vec{x})$ is computed by combining together the individual tree predictions $t_i(\vec{x})$, e.g., by performing majority voting on the individually predicted classes.

B. Fairness in ML

Many definitions of fairness have been proposed in the literature, each with its pros and cons [22]. Existing fairness definitions can be broadly categorized in two main classes: *individual fairness* definitions, requiring that similar inputs must result in similar outputs, and *group fairness* definitions,

[1]https://github.com/LorenzoCazzaro/explainable-global-fairness-verification

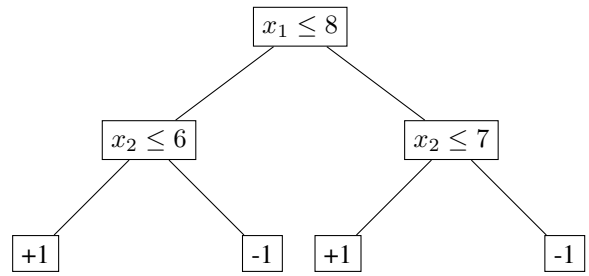


Fig. 1: Example of decision tree

requiring that a particular group of inputs taken as a whole must be treated like a different group. It is now acknowledged that individual fairness and group fairness are both prominent and useful concepts, however, they may be conflicting and are normally studied separately [23]. In this paper, we focus on a specific individual fairness definition known as lack of *causal discrimination*, which was introduced in a seminal paper on fairness testing [14].

We consider this property for several reasons: its intuitive flavour, its popularity in the literature, and its independence from the distribution of the class labels (i.e., the target function), which simplifies its practical application. Moreover, lack of causal discrimination is a powerful foundation for fairness, because it does not rely on the choice of a specific test set, but rather predicates over all the possible instances in (a subset of) the feature space $\mathcal{X}$, much in line with recent proposals on the verification of *global robustness* properties of machine learning models [24], [25], [26]. This is important in the fairness setting, because fairness is particularly relevant for minorities for which it might be hard to collect representative data in the test set. Indeed, the need for global fairness verification has been recently advocated for neural networks [18].

Intuitively, lack of causal discrimination requires that any two similar inputs must lead the classifier to the same outputs, thus capturing the intuition of individual fairness; more specifically, lack of causal discrimination requires that the classifier returns the same prediction on any two instances differing just for the value of a set of *sensitive* features $S \subseteq \{1, \ldots, d\}$. Given an instance $\vec{x}$, we let $\delta(\vec{x}, S)$ stand for the set of the instances differing from $\vec{x}$ just for a (possibly empty) subset of the sensitive features S. For example, if S included just two binary features, then $\delta(\vec{x}, S)$ would include the four instances obtained from $\vec{x}$ by setting each of the sensitive features in S to one of their two possible values, while keeping the other features unchanged. Formally, lack of causal discrimination is then defined as follows.

Definition 1 (Causal Discrimination)**.** Let $f : \mathcal{X} \to \mathcal{Y}$ be a classifier and let S be a set of sensitive features. We say that f does not perform *causal discrimination* on $\mathcal{X}' \subseteq \mathcal{X}$ if and only if, for every instance $\vec{x} \in \mathcal{X}'$, we have that $\forall \vec{z} \in \delta(\vec{x}, S) : f(\vec{z}) = f(\vec{x})$.

To exemplify the definition at work, suppose a classifier is used to evaluate the legitimacy of loan requests and that

the set of sensitive features S just includes the customer's gender. Lack of causal discrimination on $\mathcal{X}$ requires that any two identical customers differing just for their gender are guaranteed to get the same response to their loan requests. Focusing on a specific $\mathcal{X}' \subseteq \mathcal{X}$ allows one to make the fairness guarantees *conditional* and thus more practically useful, e.g., by requiring that any two identical customers with a monthly salary higher than $4,000 are guaranteed to receive the same response to their loan requests, irrespective of their gender.

III. Global Fairness Verification

We here present our new approach to the global fairness verification of tree-based classifiers.

A. Overview

Verifying the lack of causal discrimination is challenging, because it involves a universal quantification over a set of instances, possibly drawn from a continuous, unbounded feature space. Prior work on causal discrimination circumvented this problem by restricting its focus to a finite feature space and assessing fairness by means of a testing approach [14]. In particular, this restriction enables the computation of a *causal discrimination score*, defined as the fraction of instances in the feature space suffering from causal discrimination, a measure which is only meaningful as long as the feature space is finite and instances therein can be exhaustively enumerated. While one can play around this limitation by using binning to discretize the feature space, the testing approach in [14] is not exhaustive for scalability reasons, hence it can only identify counter-examples suffering from causal discrimination, but it cannot prove a lack of causal discrimination. Similar criticisms apply to other more recent proposals on fairness testing [15], [16], [17].

In the present work, we improve over the state of the art by proposing a new verification technique to formally verify the fairness guarantees supported by tree-based models. In particular, our technique allows one to automatically identify subsets of the feature space where lack of causal discrimination is ensured, rather than just counter-examples suffering from causal discrimination. Concretely, we first discuss how one can verify lack of causal discrimination for tree ensembles given a continuous, unbounded subset of the feature space $\mathcal{X}' \subseteq \mathcal{X}$ as input (Section III-B). We then build on this idea to design an effective algorithm to automatically synthesize sufficient conditions ensuring lack of causal discrimination, expressed as traditional propositional logic formulas, which are readily understandable by human experts (Section III-C). Our algorithm is iterative and its execution can be safely stopped before convergence to improve performance, without sacrificing soundness, i.e., only sufficient conditions are returned by the algorithm. Moreover, the algorithm is proved complete upon termination, i.e., the logic formulas returned by the algorithm may eventually provide a complete characterization of the fairness guarantees given by the classifier.

B. Verification Algorithm

The first problem we investigate can be formulated as follows: given a decision tree ensemble T, a set of sensitive features S and a subset of the feature space $\mathcal{X}' \subseteq \mathcal{X}$, verify that T does not perform causal discrimination on $\mathcal{X}'$.

To answer this question, we leverage the *stability* notion from the adversarial machine learning literature [27]. The idea underlying stability is that classifiers deployed in adversarial settings should not be fooled by adversarial manipulations of test instances, designed to force the classifier to return wrong predictions (so-called evasion attacks). This can be enforced by requiring the classifier to be "stable" with respect to adversarial manipulations, i.e., to stick to its original predictions despite such manipulations. For example, malware detectors must not change their predictions as the result of semantically-preserving adversarial manipulations of malicious software, e.g., if malware replaces a given API call with an equivalent set of instructions, it should still be classified as malware. Formally, stability is defined as follows.

Definition 2 (Stability)**.** Given a classifier $f : \mathcal{X} \to \mathcal{Y}$, an instance $\vec{x} \in \mathcal{X}$ and a set of adversarial manipulations $A(\vec{x})$, f is *stable* on $\vec{x}$ if and only if $\forall \vec{z} \in A(\vec{x}) : f(\vec{z}) = f(\vec{x})$.

The idea of stability generalizes from the adversarial setting to the fairness setting, because lack of causal discrimination requires that arbitrary changes to the sensitive features S must not affect the classifier predictions. A similar observation for a different definition of fairness has been performed in recent independent work [9]. The important point to note, though, is that while stability predicates on a specific instance $\vec{x} \in \mathcal{X}$, lack of causal discrimination predicates over a potentially unbounded set of instances $\mathcal{X}' \subseteq \mathcal{X}$, hence traditional approaches to stability/fairness verification such as [27] cannot be directly applied to verify lack of causal discrimination on $\mathcal{X}'$.

We thus propose to leverage recent solutions for the *data-independent* verification of decision tree ensembles [26], [28] to verify lack of causal discrimination in a continuous, unbounded subset of the feature space $\mathcal{X}'$. These verification approaches are data-independent because they do not analyze the behavior of the ensemble on specific test instances, but they rather analyze the structure of the ensemble to allow the verification of properties predicating over all the instances in (a subset of) the feature space. Concretely, data-independent analyses operate in terms of a set of *hyper-rectangles* $H \subseteq \mathcal{X}$ such that all instances $\vec{x} \in H$ satisfy a property of interest, e.g., lead to the same prediction or enjoy stability. In particular, we build on the following definition of data-independent stability analysis for our verification purposes.

Definition 3 (Data-Independent Stability Analysis)**.** Let $f : \mathcal{X} \to \mathcal{Y}$ be a classifier and let S be a set of sensitive features. A *data-independent stability analysis* is an algorithm which takes as input f and S to return as output a set of hyper-rectangles U satisfying the following property: for every instance $\vec{x} \in \mathcal{X}$, if there exists $\vec{z} \in \delta(\vec{x}, S)$ such that $f(\vec{z}) \neq f(\vec{x})$, then there exists $H \in U$ such that $\vec{x} \in H$.

Intuitively, the output of a data-independent stability analysis over-approximates the subset of the feature space where the classifier violates stability for some instances therein, i.e., if $\vec{x}$ suffers from causal discrimination, then the result of the analysis must include a hyper-rectangle H such that $\vec{x} \in H$. In the fairness setting, this means that the union of the hyper-rectangles returned by the analysis over-approximates the set of the counter-examples suffering from causal discrimination. Note that we only require an over-approximation rather than an exact characterization, because this is a minimal assumption for sound fairness verification and data-independent analyses might be approximated by design for scalability reasons. For example, the stability analysis in [26] identifies a superset of the portion of the feature space where the classifier is unstable, because robustness verification for tree ensembles (an NP-hard problem [29]) can be reduced to stability analysis.

If a data-independent stability analysis is available, solving our problem of interest is straightforward. In particular, given a decision tree ensemble T and a set of sensitive features S, we can run the data-independent stability analysis on T and S to produce the set of hyper-rectangles U. To prove that T does not perform causal discrimination on $\mathcal{X}' \subseteq \mathcal{X}$, it suffices to show that there exists no $H \in U$ such that $H \cap \mathcal{X}' \neq \emptyset$.

We now discuss how the verification approaches in [26], [28] can be used to implement a data-independent stability analysis as required by our definition, thus allowing us to take advantage of existing state-of-the-art verification approaches.

1) Resilience Verifier [26]: Recent work on adversarial machine learning introduced the *resilience* notion to characterize the security of classifiers deployed in adversarial settings. At the core of its resilience verification algorithm, there is a data-independent stability analysis for decision tree ensembles, which can be readily leveraged for our purposes by modeling an attacker such that, for all instances $\vec{x} \in \mathcal{X}$, $A(\vec{x}) = \delta(\vec{x}, S)$. Intuitively, this attacker has full control of the value of the sensitive features S, so stability against such an attacker ensures that causal discrimination is not possible. The output of the analysis is a set of hyper-rectangles over-approximating the subset of the feature space where the decision tree ensemble is unstable, thus satisfying the conditions of Definition 3.

2) VoTE Checker [28]: The VoTE Checker takes as input a decision tree ensemble and produces as output a partitioning of the feature space $\mathcal{X}$ in terms of the equivalence classes induced by the ensemble. Each equivalence class is represented as a pair (H, y), meaning that all the instances $\vec{x} \in H$ traverse the same path combination in the ensemble leading to prediction y. If there exist two equivalence classes $(H_1, y_1), (H_2, y_2)$ such that: (i) $y_1 \neq y_2$, (ii) H_1, H_2 have a non-empty intersection for all the features $j \notin S$, and (iii) H_1, H_2 have different intervals for some feature $k \in S$, then $H_1 \cup H_2$ includes a counter-example suffering from causal discrimination, because it contains two instances differing just for some sensitive feature k where the classifier returns two different predictions. It is thus possible to add both H_1 and H_2 to the set of hyper-rectangles to be returned by the analysis, thus satisfying the conditions of Definition 3.

C. Synthesis Algorithm

We now consider a different, more interesting problem: given a decision tree ensemble T and a set of sensitive features S, characterize the subset of the feature space $\mathcal{X}' \subseteq \mathcal{X}$ such that T does not perform causal discrimination on $\mathcal{X}'$.

Intuitively, this problem can be conservatively solved by subtracting from $\mathcal{X}$ all the hyper-rectangles $H \in U$ returned by the data-independent stability analysis assumed in the previous section, thus under-approximating the subset of the feature space where T is stable. This approach would be sound, but also computationally inefficient, because subtracting hyper-rectangles suffers from an exponential blowup with respect to the dimensionality of the feature space. Indeed, given two hyper-rectangles with d features, their subtraction might generate $O(d)$ hyper-rectangles in the general case, leading to $O(d^{|U|})$ hyper-rectangles in the worst case at the end of the subtraction process. This limitation can be circumvented by avoiding the computation of the subtraction and by directly reasoning in terms of instances falling out from the hyper-rectangles U. However, this would make it hard to characterize $\mathcal{X}'$ in a human-understandable way, due to both the sheer number of the hyper-rectangles and the potentially large dimensionality of the feature space.

We thus propose an iterative algorithm designed to incrementally generate increasingly complex sufficient conditions ensuring lack of causal discrimination, expressed as traditional logic formulas. This way, the first iterations of the algorithm can efficiently generate conditions that are short and easy to understand for human experts, while being arguably the most useful to analysts; the more analysis time and computational resources are available, the more complex conditions can be identified, thus detecting other subsets of the feature space where lack of causal discrimination is ensured. In other words, each iteration of the algorithm extends the sound approximation identified by the previous iteration by accounting for more complicated sufficient conditions for fairness, until the whole relevant subset of the feature space is covered by the conditions (or an early stopping criterion is met).

1) Overview: Our algorithm is inspired by the classic Apriori algorithm for frequent itemset mining [30]. We define an *item* i as a formula of the form $x_f \leq v$ or $x_f > v$, where $f \in \{1, \ldots, d\}$ identifies a feature and $v \in \mathbb{R}$. An *itemset* I is a set of formulas $\{i_1, \ldots, i_n\}$, interpreted in conjunctive form. For example, the itemset $I = \{x_1 > 1, x_1 \leq 3, x_2 > 5\}$ identifies all the instances such that the first feature is in the interval $(1, 3]$ and the second feature is in the interval $(5, +\infty)$. Formally, we write $[\![i]\!]$ for the set of the instances identified by item i and we let $[\![I]\!] = \bigcap_{i \in I} [\![i]\!]$ stand for the set of the instances identified by the itemset I. Given a hyper-rectangle H, we denote with $to_itemset(H)$ its itemset representation, i.e., the itemset I such that $[\![I]\!] = H$.

An itemset identifies a subset of the feature space which is assessed for lack of causal discrimination, thus allowing us to leverage the following *monotonicity* property to prune the search space: if I enjoys lack of causal discrimination, then

any other $I' \supseteq I$ does that as well, because $[\![I']\!] \subseteq [\![I]\!]$; hence, I' itself does not need to be analyzed. This property allows one to assess itemsets for lack of causal discrimination by starting from those with smaller cardinality.

To exemplify how the algorithm works at a high level, consider a two-dimensional feature space and assume U contains just two hyper-rectangles $H_1 = \langle(1,5],(3,8]\rangle$ and $H_2 = \langle(4,7],(2,6]\rangle$. Our goal is characterizing those instances falling *outside* both H_1 and H_2. Instances laying outside H_1 can be described as the union of the following four hyper-rectangles:

- $H_{11} = \langle(-\infty,1],(-\infty,+\infty)\rangle$, represented as the itemset $I_{11} = to_itemset(H_{11}) = \{x_1 \leq 1\}$
- $H_{12} = \langle(5,+\infty),(-\infty,+\infty)\rangle$, represented as the itemset $I_{12} = to_itemset(H_{12}) = \{x_1 > 5\}$
- $H_{13} = \langle(-\infty,+\infty),(-\infty,3]\rangle$, represented as the itemset $I_{13} = to_itemset(H_{13}) = \{x_2 \leq 3\}$
- $H_{14} = \langle(-\infty,+\infty),(8,+\infty)\rangle$, represented as the itemset $I_{14} = to_itemset(H_{14}) = \{x_2 > 8\}$

Instances laying outside H_2 can instead be described as the union of the following four hyper-rectangles:

- $H_{21} = \langle(-\infty,4],(-\infty,+\infty)\rangle$, represented as the itemset $I_{21} = to_itemset(H_{21}) = \{x_1 \leq 4\}$
- $H_{22} = \langle(7,+\infty),(-\infty,+\infty)\rangle$, represented as the itemset $I_{22} = to_itemset(H_{22}) = \{x_1 > 7\}$
- $H_{23} = \langle(-\infty,+\infty),(-\infty,2]\rangle$, represented as the itemset $I_{23} = to_itemset(H_{23}) = \{x_2 \leq 2\}$
- $H_{24} = \langle(-\infty,+\infty),(6,+\infty)\rangle$, represented as the itemset $I_{24} = to_itemset(H_{24}) = \{x_2 > 6\}$

To identify instances laying outside both H_1 and H_2, we first inspect all the itemsets above and we check whether they identify subsets of the feature space intersecting H_1 or H_2. If we do not find overlaps, the itemsets already represent instances falling out both H_1 and H_2, hence causal discrimination cannot happen there. In our example, we identify the itemsets I_{11}, I_{14}, I_{22} and I_{23} as sufficient conditions for lack of causal discrimination; note that the first and the third itemsets only involve the feature x_1, while the second and the fourth itemsets only involve the feature x_2. The other itemsets I_{12}, I_{13}, I_{21} and I_{24}, instead, identify subsets of the feature space where causal discrimination might potentially happen. These itemsets are thus combined together to generate additional itemsets to check, each possibly using both features x_1 and x_2, leading to conditions of higher complexity. For example, I_{12} and I_{24} generate the new itemset $\{x_1 > 5, x_2 > 6\}$, which represents again instances falling out both H_1 and H_2, thus identifying sufficient conditions for lack of causal discrimination. Instead, I_{12} and I_{13} generate the new itemset $\{x_1 > 5, x_2 \leq 3\}$, which overlaps with H_2, hence it identifies a subset of the feature space where causal discrimination may happen. This itemset may thus be combined with other itemsets to undergo further refinements over x_1 and x_2, leading to smaller intervals on them, possibly identifying additional sufficient conditions for proving lack of causal discrimination.

2) Algorithm: Having defined the key intuitions of our proposal, we are now ready to present the details of the synthesis algorithm (Algorithm 1). The algorithm takes as input a tree ensemble T and a set of sensitive features S to return as output a set of sufficient conditions on the feature space ensuring lack of causal discrimination. The algorithm starts by invoking the ANALYZE function over T and S, which implements a data-independent stability analysis (cf. Definition 3) returning a set of hyper-rectangles U where T may be unstable (line 2). The algorithm then initializes an empty set of fairness conditions F and generates a set of candidates C, initialized with itemsets involving just a single feature, as described in our example; this step is implemented by the GEN_ITEMSETS function (lines 3-4). The definition of GEN_ITEMSETS is also reported in Algorithm 1; it implements the simple intuition from our previous example to first construct a hyper-rectangle H' constraining just a single feature i and then convert it into an equivalent itemset. In the first loop of the algorithm, each itemset $I \in C$ is then checked against U: if $[\![I]\!]$ does not intersect any hyper-rectangle $H \in U$, the itemset identifies a sufficient condition for lack of causal discrimination, hence it is added to the set of fairness conditions F (lines 5-7). The itemsets which do not immediately contribute to extending F are instead used in the main loop of the algorithm (lines 8-19). In particular, all such itemsets are combined with each other through a *meet* operator $\sqcap$ to produce new itemsets to analyze; such itemsets can either be proved fair or undergo additional refinements at later iterations, as long as there are candidates to process. The meet operator is defined and commented below.

Definition 4 (Meet Operator)**.** Given two itemsets I_1, I_2 such that $|I_1| = |I_2| = k$ and $|I_1 \cap I_2| = k-1$, we define their *meet* $I_1 \sqcap I_2$ as the itemset $I = I_1 \cup I_2$, provided that the following conditions hold:

1) $[\![I]\!] \neq \emptyset$, i.e., the itemset I identifies a non-empty subset of instances;
2) $[\![I]\!] \subset [\![I_1]\!]$ and $[\![I]\!] \subset [\![I_2]\!]$, i.e., the itemset I identifies less instances than both I_1 and I_2.

If any of the aforementioned conditions do not hold, the meet I does not exist.

Observe that, given two itemsets I_1, I_2 of cardinality k sharing $k-1$ elements, their meet $I_1 \sqcap I_2$ produces a new itemset $I_1 \cup I_2$ of cardinality $k+1$, i.e., itemsets are generated in increasing order of cardinality to leverage the discussed monotonicity property. The two technical conditions of Definition 4 just ensure that testing the newly generated itemset might be useful, i.e., the new itemset is non-empty and differs from the previously generated itemsets I_1, I_2. Although these conditions are formulated in a declarative style to simplify their understanding, the meet operator $\sqcap$ is straightforward to implement in practice. In particular, let i^* be the (only) item such that $i^* \in I_2 \setminus I_1$ and let f^* be the feature predicated upon by i^*, then we let $I_1 \sqcap I_2 = I_1 \cup \{i^*\}$ provided that the two conditions of the definition are satisfied. The implementation of the first condition checks whether all the items $i \in I_1$

Algorithm 1 Synthesizing Fairness Conditions

```
1:  function SYNTHESIZE(T, S)
2:      U ← ANALYZE(T, S)
3:      F ← ∅
4:      C ← GEN_ITEMSETS(U)
5:      for I ∈ C do
6:          if ∀H ∈ U : ⟦I⟧ ∩ H = ∅ then
7:              F ← F ∪ {I}
8:      C ← C \ F
9:      while C ≠ ∅ do
10:         C′ ← ∅
11:         for I1 ∈ C do
12:             for I2 ∈ C do
13:                 I ← I1 ⊓ I2
14:                 if I exists ∧ ∄I′ ∈ F : ⟦I⟧ ⊆ ⟦I′⟧ then
15:                     if ∀H ∈ U : ⟦I⟧ ∩ H = ∅ then
16:                         F ← F ∪ {I}
17:                     else
18:                         C′ ← C′ ∪ {I}
19:         C ← C′
20:     return F
21:
22: function GEN_ITEMSETS(U)
23:     C ← ∅
24:     for H = ⟨H1, ..., Hd⟩ ∈ U do
25:         for i ∈ {1, ..., d} do
26:             H̄i ← (−∞, +∞) \ Hi
27:             for intv ∈ H̄i do
28:                 H′ ← ⟨(−∞, +∞)^(i−1), intv, (−∞, +∞)^(d−i)⟩
29:                 C ← C ∪ {to_itemset(H′)}
30:     return C
```

predicating on f^* have a non-empty intersection with i^*. The implementation of the second condition amounts to verifying that adding i^* to I_1 identifies a smaller interval for feature f^*.

Finally, we report the key insight that not all the itemsets must undergo the potentially expensive intersection test against U. In particular, if $[\![I]\!] \subseteq [\![I']\!]$ for some I' which we already proved fair, I can be ignored, because it does not identify new sufficient conditions for fairness (cf. the check at line 14). Note that it is easy to move from the declarative style of this check to its implementation, because checking $[\![I]\!] \subseteq [\![I']\!]$ amounts to checking that, for all features f, the interval on f identified by I is included in the interval on f identified by I'.

3) Formal Results: We can prove that our algorithm is both sound and complete, as formalized below. Soundness ensures that any itemset I returned by the synthesis algorithm is a sufficient condition for fairness, i.e., instances in $[\![I]\!]$ cannot suffer from causal discrimination.

Theorem 1 (Soundness)**.** For any decision tree ensemble T and set of sensitive features S, the call SYNTHESIZE(T, S) returns a set of itemsets F such that, for every $I \in F$, T does not perform causal discrimination on $[\![I]\!]$.

Proof. The call ANALYZE(T, S) returns a set of hyper-rectangles U satisfying the following property: for every instance $\vec{x} \in \mathcal{X}$, if there exists $\vec{z} \in \delta(\vec{x}, S)$ such that $f(\vec{z}) \neq f(\vec{x})$, then there exists $H \in U$ such that $\vec{x} \in H$. This means that T cannot perform causal discrimination on any $\mathcal{X}' \subseteq \mathcal{X}$ such that $\forall H \in U : \mathcal{X}' \cap H = \emptyset$. The conclusion follows by observing that any itemset I which is added to F must satisfy this property (cf. lines 6-7 and lines 15-16). □

Completeness, instead, ensures that the combination of all the itemsets F returned by the synthesis algorithm coincides with the subset of the feature space disjoint from the result of the data-independent stability analysis, i.e., it represents the ideal outcome of the synthesis algorithm. Note that, if the data-independent stability analysis is not over-approximated but exact, this ensures that F covers all the instances where the classifier is fair.

Theorem 2 (Completeness)**.** For any decision tree ensemble T and set of sensitive features S, the call SYNTHESIZE(T, S) returns a set of itemsets F such that $\bigcup_{I \in F} [\![I]\!] = \mathcal{X} \setminus \bigcup_{H \in U} H$, where U is the output of ANALYZE(T, S).

Proof. We prove the equality of the two sets by showing that the first is included in the second and vice-versa. Specifically:

- Consider any instance $\vec{x} \in \bigcup_{I \in F} [\![I]\!]$, we show that for all $H \in U$ we have $\vec{x} \notin H$, which implies $\vec{x} \in \mathcal{X} \setminus \bigcup_{H \in U} H$. Indeed, the algorithm ensures that for all $I \in F$ we have that $\forall H \in U : [\![I]\!] \cap H = \emptyset$ (cf. lines 6-7 and lines 15-16).
- Consider any instance $\vec{x} \in \mathcal{X} \setminus \bigcup_{H \in U} H$, we show that there exists $I \in F$ such that $\vec{x} \in [\![I]\!]$, which implies $\vec{x} \in \bigcup_{I \in F} [\![I]\!]$. We first observe that, for all $H_j \in U$, the call GEN_ITEMSETS(U) returns a set of at most $2d$ itemsets:
$$C_j = \{\{i_1\}, \ldots, \{i_{2d}\}\},$$
such that $\vec{x} \in [\![i_k]\!]$ for some i_k; we refer to such item i_k as the *witness* for H_j. The itemset I includes all the witnesses for each $H_j \in U$, thus ensures that $\vec{x} \in [\![I]\!] = \bigcap_{i \in I} [\![i]\!]$. The conclusion follows by observing that either the itemset I or another itemset I' such that $[\![I]\!] \subseteq [\![I']\!]$ is eventually enumerated by the algorithm.

□

4) Implementation: We implemented the synthesis algorithm presented in this section in C++. Our implementation leverages the data-independent stability analysis used by the resilience verifier presented in [26], which we simply leverage as a black box to implement the ANALYZE function. The stability analysis is based on an iterative algorithm, whose computational cost may show an exponential trend, but can be constrained by fixing the number of iterations, without significantly affecting the precision of the results [26]. Although the rest of the implementation is a rather direct translation of the pseudocode in Algorithm 1, a few important details are worth discussing. A first point to note is that our implementation supports a user-specified early stopping criterion in terms of

TABLE I: Dataset statistics (we report in parentheses the number of categorical features after one-hot-encoding).

Dataset	#Num features	#Cat features	#Instances	%Positive
Adult	6	7 (81)	45,222	25%
German	7	13 (49)	1,000	70%
Health	93	2 (10)	166,842	68%

a maximum number of iterations of the algorithm. This is useful because, like Apriori, our algorithm has an exponential time complexity with respect to the number of items in the worst case [30]. However, we empirically observed in our experimental evaluation that a small number of short conditions already allow one to largely characterize the fairness guarantees of tree-based classifiers, hence restricting the number of iterations saves analysis time, while leading to just a small loss in precision in practice. Note that early stopping preserves the soundness of the analysis, while obviously sacrificing its completeness. For space reasons, other interesting implementation details are presented in Appendix A.

IV. Experimental Evaluation

In this section, we experimentally assess the effectiveness of our global fairness verification approach along different axes.

A. Methodology

We evaluate our proposal on decision tree ensembles trained over three public datasets used in the fairness literature [9], each associated with a binary classification task where fairness matters: Adult[2], German[3] and Health[4]. The classification task for Adult requires predicting yearly income (above or below \$50K), while the classification task for German requires assigning credit scores (good or bad), hence these datasets could be used to train classifiers deployed, e.g., to assess loan requests. The classification task for Health, instead, requires predicting ten-year mortality (above or below the median Charlson index), hence it could be used to train classifiers deployed in the health insurance setting. Since in all cases we do not want to discriminate customers based on their gender, we use the attribute *sex* as the binary sensitive feature. For simplicity, we only consider a single sensitive feature leading to a clear understanding of the related fairness issues, however, our proposal supports an arbitrary number of sensitive features.

We pre-process the datasets by following three steps: (i) we normalize all the numerical features in the interval $[0, 1]$; (ii) we perform the one-hot-encoding of all the categorical features; (iii) we remove the features and instances containing missing values. The characteristics of the datasets are in Table I, along with the number of categorical features before and after step (ii). We partition datasets into a training set $\mathcal{D}_{train}$ and a test set $\mathcal{D}_{test}$, using 80-20 stratified random sampling. For each dataset, we use $\mathcal{D}_{train}$ to train standard Random Forest (RF) models as available in sklearn [31]. Experiments are then conducted on both $\mathcal{D}_{test}$ and a synthetic dataset $\mathcal{D}_{rand}$ including 100k random instances, generated by assigning a random value to all their features, while restricting categorical features to a set of plausible values observed in the original dataset. The rationale for having these two sets is that $\mathcal{D}_{test}$ is supposed to follow the same distribution of $\mathcal{D}_{train}$, hence it represents expected inputs at test time, while $\mathcal{D}_{rand}$ provides a much larger view of the entire feature space, which is interesting because our fairness guarantees are global and generalize beyond the test set. Both $\mathcal{D}_{test}$ and $\mathcal{D}_{rand}$ are thus useful to investigate different aspects of our proposal. We write $\mathcal{D}$ to stand for any between $\mathcal{D}_{test}$ and $\mathcal{D}_{rand}$, using the notation $\vec{x} \in \mathcal{D}$ to identify instances in $\mathcal{D}$ while ignoring their class label when $\mathcal{D} = \mathcal{D}_{test}$.

The experiments are designed to answer three key research questions:

1) What is the *precision* of the analysis performed by the synthesis algorithm? In other words, do the synthesized fairness conditions cover well the portions of the feature space where lack of causal discrimination is ensured?
2) What is the *explainability* of the results returned by the synthesis algorithm? Can we effectively characterize the fairness guarantees of classifiers in terms of a small number of conditions of limited complexity?
3) What is the *performance* of the synthesis algorithm? How is the analysis running time influenced by the size and complexity of the classifiers?

B. Precision of the Analysis

In our first experiment, we estimate the precision of our verification approach by comparing the set of hyper-rectangles U returned by the data-independent stability analysis with the formulas F returned by the synthesis algorithm. The completeness theorem ensures that F identifies the subset of the feature space disjoint from all the hyper-rectangles in U, provided that the synthesis algorithm runs up to completion. In practice, however, we expect the synthesis algorithm to be normally subject to early stopping, due to the exponential blow-up in the number of candidates to be analyzed at the different iterations and the increasing complexity of the synthesized formulas. We thus estimate the precision of a partial output obtained when the synthesis algorithm is forcefully stopped after six iterations, because we are interested in formulas with a small number of items, amenable for human understanding. Later experiments assess the impact of the number of iterations on the analysis results and running times.

We first compute the casual discrimination score on the dataset $\mathcal{D}$ based on the set of hyper-rectangles U. In particular, we observe that an instance $\vec{x}$ can only suffer from causal discrimination when it belongs to some $H \in U$, so the casual discrimination score on $\mathcal{D}$ can be conservatively approximated as follows:

$$d(U, \mathcal{D}) = \frac{|\{\vec{x} \in \mathcal{D} \mid \exists H \in U : \vec{x} \in H\}|}{|\mathcal{D}|}$$

Similarly, we can use the formulas in F to reason about fairness by observing that any instance satisfying some formula

[2]https://archive.ics.uci.edu/ml/datasets/adult
[3]https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)
[4]https://www.kaggle.com/c/hhp

TABLE II: Computed measures for different datasets and models.

Dataset	#Trees	Depth	$\mathcal{D}_{test}$			$\mathcal{D}_{rand}$	
			a	d	$\tilde{d}$	d	$\tilde{d}$
Adult	5	5	0.811	0.003	0.003	0.348	0.348
	9	5	0.826	0.003	0.003	0.323	0.323
	13	5	0.824	0.014	0.014	0.330	0.330
	5	6	0.833	0.004	0.004	0.113	0.113
	9	6	0.834	0.004	0.004	0.206	0.206
	13	6	0.840	0.006	0.006	0.235	0.235
German	5	5	0.720	0.230	0.230	0.349	0.349
	9	5	0.725	0.230	0.230	0.349	0.349
	13	5	0.725	0.240	0.240	0.373	0.373
	5	6	0.745	0.230	0.230	0.371	0.371
	9	6	0.745	0.325	0.325	0.437	0.450
	13	6	0.730	0.375	0.415	0.486	0.571
Health	5	5	0.801	0.008	0.008	0.027	0.027
	9	5	0.802	0.008	0.008	0.027	0.027
	13	5	0.803	0.016	0.016	0.028	0.028
	5	6	0.810	0.018	0.018	0.209	0.209
	9	6	0.809	0.018	0.018	0.209	0.209
	13	6	0.810	0.020	0.020	0.259	0.259

in F cannot suffer from causal discrimination. This allows us to compute the following over-approximation of d:

$$\tilde{d}(F, \mathcal{D}) = 1 - \frac{|\{\vec{x} \in \mathcal{D} \mid \exists I \in F : \vec{x} \in [\![I]\!]\}|}{|\mathcal{D}|}$$

Observe that $\tilde{d}$ is always an upper bound of d, because our analysis is sound. Table II reports the values of d and $\tilde{d}$ computed for different datasets and models. We observe that d and $\tilde{d}$ coincide for the very large majority of the cases, meaning that the formulas in F accurately characterize the subset of the feature space which is disjoint from all the hyper-rectangles in U, even when enforcing early stopping after six iterations. Note that $\tilde{d}$ does not coincide with d just in three cases, all associated with the German dataset, but we experimentally assessed that the two scores coincide also there when increasing the number of iterations of the synthesizer to seven. The table also reports the accuracy a on the test set $\mathcal{D}_{test}$ to show that all the analyzed models perform reasonably well in practice.

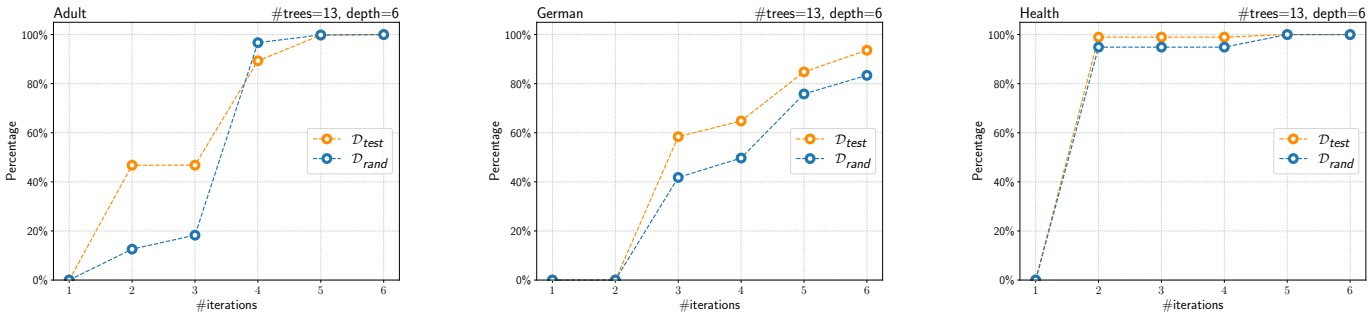
C. Explainability of the Results

To assess to what extent the formulas F are explainable, we carry out selected experiments on the most complex models that we trained on the considered datasets, i.e., ensembles of 13 trees with maximum depth six, because they are expected to be the most challenging to explain. We follow the methodology used in previous work on the use of rules based on logic formulas to extract explainable information from classifiers, like [32], [33], [34]. In these proposals, the complexity of logic formulas is measured by their length (number of atoms) and short logic formulas are preferred because of their clearer understanding. Of course, having a small number of formulas is also important for explainability.

In the first experiment, we assess how the percentage of instances for which F is able to provide a proof of fairness grows when varying the number of iterations of the synthesis algorithm. Of course, we compute this percentage with respect to the number of instances which do not suffer from causal discrimination, i.e., which might actually admit a proof of fairness. Since a run with k iterations can only produce formulas involving at most k items, this experiment provides insights on how complex sufficient conditions for fairness turn out to be in practice. Figure 2 plots the observed trend. The experimental results for the Adult dataset show that just four iterations of the algorithm suffice to establish fairness proofs for more than 90% of the instances of both $\mathcal{D}_{test}$ and $\mathcal{D}_{rand}$, while just two iterations are enough to cover basically all the instances of the two sets for the Health dataset. The German dataset is the most challenging, since five iterations of the algorithm are needed to cover around 80% of the instances in the two sets. In the end, the experiment shows that short logic formulas including at most five items are expressive enough to establish useful fairness proofs in practice, while being small enough to be easily understandable by human experts.

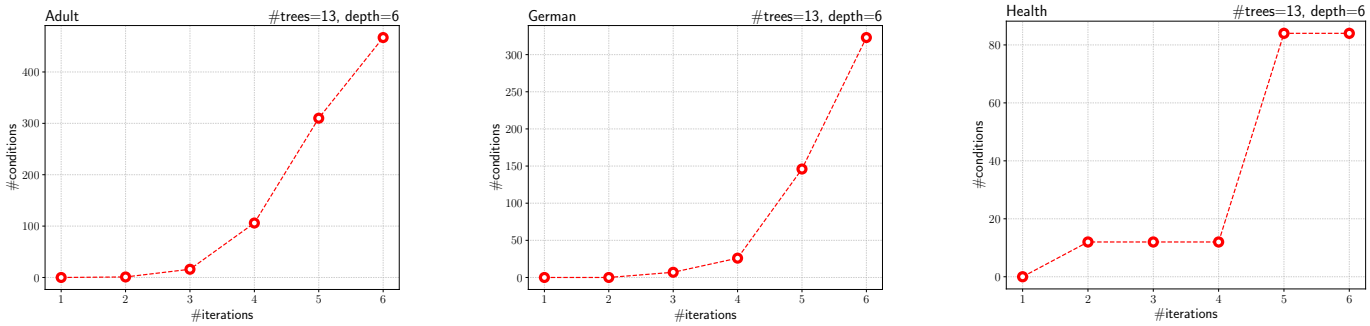
Clearly, however, our first experiment provides just a partial picture of explainability, because it captures information about the complexity of formulas, but it does not tell how many formulas should be taken into consideration by human analysts to draw useful conclusions. Indeed, we observe that the amount of formulas can significantly grow as the number of iterations of the synthesis algorithm increases. Figure 3 plots how the number of synthesized formulas grows when varying the number of iterations of the synthesis algorithm, showing an exponential trend for the Adult and German datasets. We see an interesting trend in the results for the Health dataset: the number of formulas does not necessarily increase from one iteration to another, since in some cases the synthesizer is not able to produce longer formulas that provide proof of fairness. However, the figure shows a significant increase in the number of formulas when five iterations are performed, revealing that later iterations could mine longer formulas even when, during an iteration, new formulas are not discovered.

Nevertheless, we can show that the number of *important* formulas for human analysts is relatively small in practice. We estimate the importance of a formula by counting the number of instances in $\mathcal{D}_{train}$ which are covered by the formula: the intuition is that the more instances are covered by the formula, the more the formula is expressive to prove fairness according to the training data. We identify the set of the top k most important formulas by means of a greedy strategy. We first select the most important formula in terms of number of covered instances, we then remove the covered instances from $\mathcal{D}_{train}$ before selecting the second most important formula and so on, until k formulas have been selected. Thus, by fixing the number of iterations of the synthesis algorithm and varying the number k, we can assess how the percentage of instances proved fair by the top k formulas grows. Figure 4 plots the observed trend for six iterations of the synthesis algorithm. The figure shows that for Adult and Health just the top 10 formulas suffice to cover around 90% of the $\mathcal{D}_{test}$ instances, while for German the top 20 formulas allow one to establish a proof of fairness for 80% of the instances in $\mathcal{D}_{test}$. This shows that a small number of formulas is sufficient to



(a) (b) (c)

Fig. 2: Percentage of instances for which F is able to provide a proof of fairness.



(a) (b) (c)

Fig. 3: Number of formulas in F when varying the number of iterations.

characterize the fairness guarantees on the test data for all the considered datasets. In general, more formulas are needed to cover synthetic instances in $\mathcal{D}_{rand}$. Indeed, the figure shows that, while the top 20 formulas still allow one to cover more than 90% of the instances in $\mathcal{D}_{rand}$ for Health, they cover just around 30% of the instances of Adult and around 60% of the instances of German. The difference between the results on $\mathcal{D}_{test}$ and $\mathcal{D}_{rand}$ can be explained by observing that the synthesized conditions depend on the thresholds learned from $\mathcal{D}_{train}$, which is the same set used to rank the conditions, hence the top conditions generalize better on a test set with the same distribution of $\mathcal{D}_{train}$ than on a set of randomly generated instances. The good news is that a small number of conditions still suffice to cover a non-negligible share of $\mathcal{D}_{rand}$ in all cases.

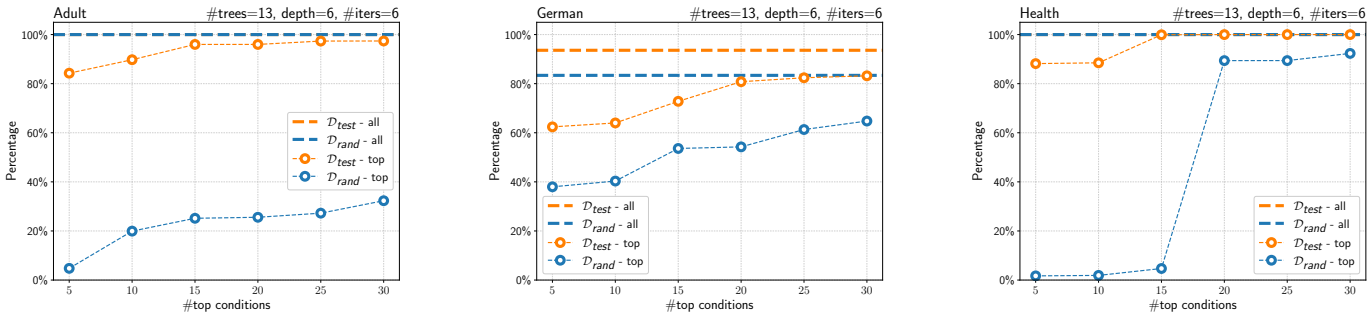
We finally perform a more qualitative evaluation of the explainability of the conditions generated by the synthesis algorithm by considering the German dataset as a case study. For space reasons, this is discussed in Appendix B.

D. Performance Evaluation

We finally analyze the performance of the synthesis algorithm when varying different dimensions of our experimental setting. Recall that the synthesis algorithm invokes the data-independent stability analysis in [26] as a black box, however any other analysis like [28] could be used in its place, so we do not include the cost of this preliminary operation in our performance evaluation. For the sake of completeness, we separately assess the computational cost of the data-independent stability analysis in Appendix C. Times are computed for our sequential implementation of the synthesis algorithm, running on a virtual machine with 20 cores, 98.8GB of RAM and Ubuntu 20.04.4 LTS on a server with an Intel Xeon Gold 6148 2.40GHz.

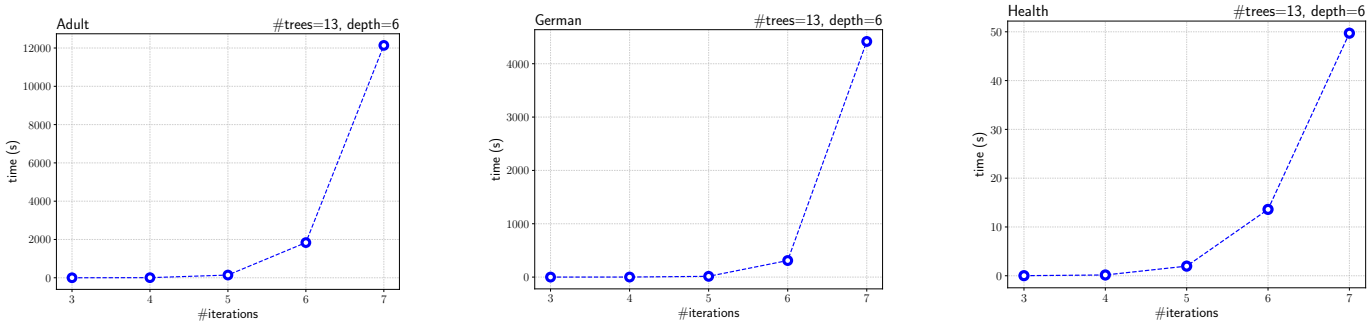
First, we plot how the analysis times change when increasing the number of analysis iterations from one to seven. For the sake of readability, we only focus on ensembles of 13 trees with maximum depth six, which are the most complex models in our experimental evaluation and likely the most challenging models to analyze. The results are shown in Figure 5. The first observation is that all the models can be analyzed in a matter of minutes when performing six iterations of the algorithm, like in our previous experiments: the analysis takes around 30 minutes on the Adult dataset, five minutes on the German dataset and just 13 seconds on the Health dataset. The gap of the analysis time on the third dataset can be explained by the smaller number of categorical features therein. In general, the plots show an exponential growth of the analysis time as the number of iterations increases. Luckily, previous results show that conditions containing at most five items are already enough to cover a large part of the feature space for all the analyzed models, hence further increasing the number of analysis iterations for them is unimportant in practice.

We then show how the running times of the synthesis



(a) (b) (c)

Fig. 4: Percentage of instances for which the top k formulas of F are able to provide a proof of fairness.



(a) (b) (c)

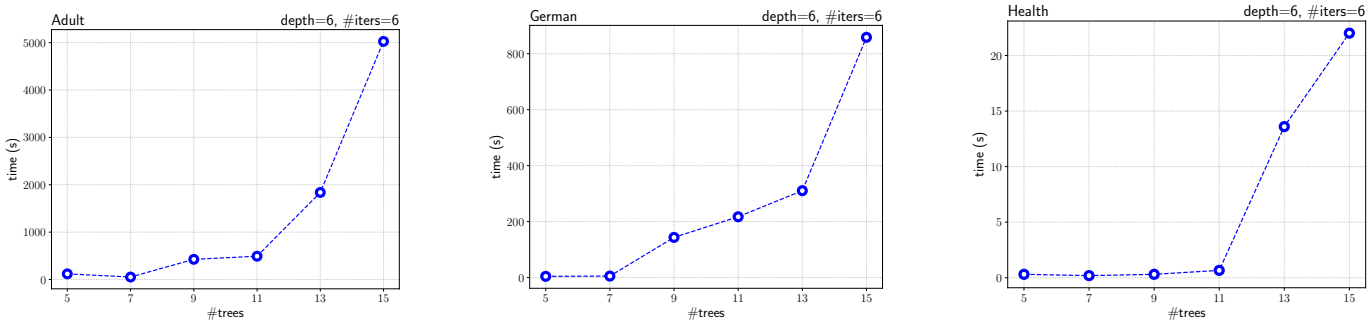
Fig. 5: Running times when varying the number of analysis iterations.

algorithm are affected by the number of trees in the ensemble, as well as by their depth. We first run the algorithm over ensembles including an increasing number of trees of at most depth six (Figure 6) and then over ensembles including 13 trees of increasing maximum depth (Figure 7), stopping the analysis after six iterations. All the settings involving tree ensembles with at most 13 trees or trees with maximum depth six like in our previous experiments terminate in a matter of minutes, thus showing the practicality of our proposal. Nevertheless, both plots show that the analysis times exhibit an exponential growth with respect to the complexity of the model, since the complexity of the analysis algorithm is exponential with respect to the number of items, which in turn depends on the number of features and thresholds occurring in the model. Thus, the analysis time for more complex models, e.g., the tree ensembles with maximum depth seven trained on the Adult dataset, could significantly increase. This further motivates the benefits of an iterative analysis approach as the proposed one, which allows one to leverage an early stopping criterion to collect partial (yet empirically precise) results even for cases where analysis convergence may turn out to be too expensive.

E. Summary

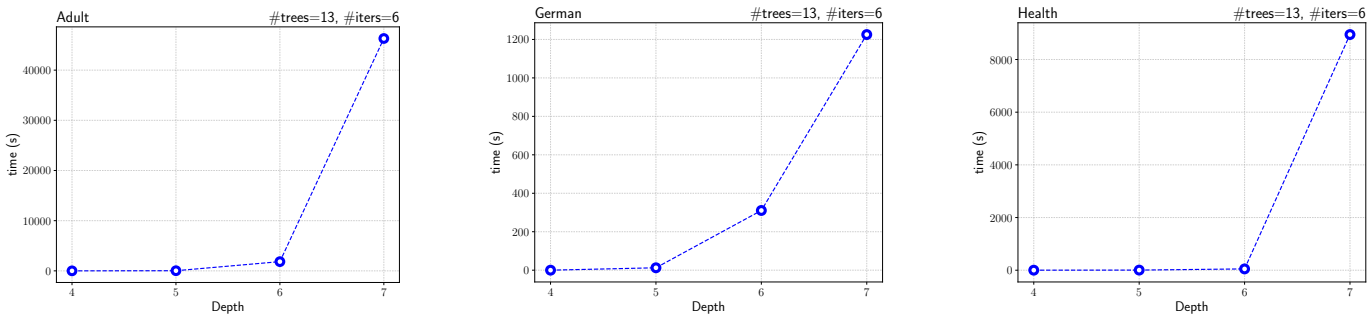
Our previous experiments showed that a small number of iterations of the synthesis algorithm are sufficient to provide a precise and human-understandable characterization of the fairness guarantees provided by the analyzed classifiers. Specifically, for all datasets and models we were consistently able to obtain practically useful analysis results when terminating the analysis after six iterations.

The results were *precise*, because they matched the fairness guarantees provided by the classifier on both the test set and a set of synthetic instances randomly created to provide a more extensive picture of the entire feature space. The results were *explainable*, because a small number of conditions of limited complexity turned out to be sufficient to largely characterize the fairness guarantees provided by the classifier, in particular over the test set. Finally, our algorithm is *efficient* enough for practical adoption on the analyzed models, because all the experiments terminated in a matter of minutes when limiting the number of analysis iterations. Yet, we notice that our algorithm shows an exponential complexity with respect to the number of items, like the Apriori algorithm it is inspired from [30]. In our application setting, the number of items is correlated with the number of features and thresholds occurring in the model, hence we observe that large models in terms of depth or number of trees may eventually pose challenges to scalability. Nevertheless, we showed that limiting the number of analysis iterations is useful to collect meaningful results even for cases that may be too complicated to analyze up to convergence, which might be enough to mitigate scalability problems. We



(a) (b) (c)

Fig. 6: Running times when varying the number of the decision trees in the ensemble.



(a) (b) (c)

Fig. 7: Running times when varying the maximum depth of the decision trees in the ensemble.

leave a more extensive evaluation on larger models and an additional optimization of our implementation to future work.

V. Related Work

We categorize related work in two broad research areas: fairness testing and fairness verification. We also briefly discuss prior work on fairness and explainability, which is a very recent research area that is getting traction.

A. Fairness Testing

Fairness testing estimates the fairness guarantees of a classifier by means of the automated generation of a number of test instances, which are fed to the model to identify potential room for discrimination. The first work on fairness testing we are aware of led to the development of the Themis tool [14], [35]. Themis uses a testing approach to identify instances which may suffer from causal discrimination, which is the same notion of fairness used in this paper. Its approach is based on a random generation of test instances, whose number is determined by a simple statistical test. Later work in the area like Aequitas improved the test instance generation approach by integrating the global random search of Themis with a perturbation-based local search [16]. Even more recent work combined symbolic execution with local explanation techniques to further improve the effectiveness of fairness testing [15], proposed more advanced statistical tests [36] and improved the explainability of the testing results [17].

Contrary to our proposal, fairness testing performs an under-approximated analysis, i.e., it can find counter-examples to fairness, but it cannot prove fairness for specific classes of individuals. This is similar to what happens in software verification, where testing can be used to prove the presence of bugs, but not their absence. Testing thus plays an orthogonal role with respect to verification: when fairness guarantees cannot be proved for specific classes of individuals, one can rely on testing to identify counter-examples. Indeed, it would be interesting to study how our fairness verification approach could boost the effectiveness of fairness testing, in particular by directing it towards portions of the feature space where discrimination might potentially happen. We leave the investigation of this research idea to future work.

B. Fairness Verification

Fairness verification of ML models attracted a lot of attention from the community, as shown by the emergence of several recent surveys on the topic [5], [6], [7]. Fairness is indeed a broad research area and several definitions of fairness have been proposed in the literature, each with its pros and cons: see [37], [22], [23] for a critical comparison of different fairness definitions. Of course, when pursuing formal verification, one has to stick to a specific notion of fairness, which is lack of causal discrimination in our case. This property belongs to the class of individual fairness and we choose it for different reasons: it is intuitive, popular,

and provides global fairness guarantees beyond the test data. Since individual fairness and group fairness deal with different concerns and rely on rather different technical tools, we consider verification of group fairness as complementary to our work [38], [39], [40], [41].

A key difference of our work with respect to the state of the art is its focus on tree-based models, which have been largely ignored by prior work. Indeed, the first work on the verification of individual fairness focused on linear models such as logistic regression and support vector machines [11]. Most later work, instead, focused on neural networks and cannot be applied to decision tree ensembles [18], [12], [10]. A general approach to fairness verification based on SMT solving was presented in [13]. The approach can be applied to tree-based models by encoding them into SMT, however, it treats fairness just as a binary, unconditional property, i.e., a model is only considered to be fair when sensitive features never play any role in classification. This is too restrictive in practice, as shown by their experimental evaluation which only verified two out of 40 analyzed models as fair. Our verification approach is more expressive, because it automatically identifies sufficient conditions for fairness, defining specific classes of individuals where discrimination cannot happen; these classes may be small or large depending on the actual fairness guarantees of the model. Indeed, observe that the trivially true condition subsumes the simple verification problem considered in [13].

The only fairness verification approach designed for tree-based models we are aware of is presented in [9]. Their approach allows the verification of a local fairness property, which just predicates on a specific set of test instances, thus providing weaker fairness guarantees than the global fairness proofs offered by lack of causal discrimination. Also, their proposed approach just computes the causal discrimination score of the classifier over the test set, without providing any explanation of which individuals cannot suffer from discrimination. On the other hand, their approach was integrated into a fair training algorithm called FATT, which is an interesting avenue for future work we would like to pursue.

C. Fairness and Explainability

Reconciling fairness and explainability has been recognized as an important problem for specific application scenarios, such as algorithmic hiring [21]. However, the area is still in its infancy [42] and there has been limited work on the topic so far. Recent work focused on designing post-hoc analyses for fairness that provide feature-level explanations, i.e., estimate the influence of each feature on the bias of the model w.r.t. a fairness definition and a specific set of instances [43], [44]. An alternative type of post-hoc fairness analysis is based on counterfactual explanations computed on specific inputs to explain the unfair behavior of the ML model [45], [46]. Contrary to post-hoc fairness analysis methods, which allow one to derive explanations about the fair behavior of the ML model on a specific set of instances, our verification method analyzes the classifier itself without relying on a specific set of instances and provides explanations about the fair behavior of the classifier over the entire feature space, i.e., all the possible inputs of the classifier, including unseen ones. Moreover, recent research also investigated how to train ML models that are both explainable and fair [47], [48], [49] and explainable fairness for recommender systems [50], [51].

The use of logic formulas for explainability purposes has also been investigated by the community. Prior work proposed approaches to use logic formulas as building blocks of logic-related models, i.e., rule lists [52] and decision sets [53], that exhibit high explainability and accuracy. Moreover, since it is difficult to explain the logic behind decision tree ensembles because of their large number of trees, several contributions in the literature proposed methods to extract a set of decision rules that describe the tree-based model in order to provide explanations at a global level about the outcome of its predictions [32], [33], [34], [54], [55], [56]. However, the idea of using logic formulas to explain the fairness guarantees provided by a ML model is novel to the best of our knowledge. In other words, a key difference of our work with respect to the state of the art is the target of the explanations, because our proposal aims to explain the fairness guarantees of a tree ensemble, not the outcome of its predictions.

VI. Conclusion

In this paper, we presented a new global fairness verification approach for tree-based classifiers. Our approach synthesizes sufficient conditions for fairness that provide global fairness guarantees, since they predicate over the entire feature space rather than on a test set of instances. The generated conditions are expressed as traditional propositional logic formulas that are readily understandable by human experts, i.e., they are explainable. The analysis is proved to be sound and complete. Extensive experimental results on public datasets show that the analysis is precise, easily understandable by human experts and efficient enough for practical adoption.

We foresee a few relevant directions for future work. First, we would like to leverage our verification approach as a powerful foundation to train tree ensembles satisfying global fairness properties. This seems feasible because prior work showed how a local fairness verifier can be used to train locally fair models [9]. Moreover, we plan to integrate fairness verification and fairness testing by using the conditions generated by our analysis and SMT solving to effectively find counterexamples suffering from causal discrimination. Indeed, the conditions returned by our synthesis algorithm identify portions of the feature space that cannot include such counterexamples, so fairness testing can be made more effective by sampling only from different areas. Finally, we would like to explore the generalization of our approach to capture group fairness properties of tree-based models [40].

Acknowledgement

This work was partially supported by iNEST (Interconnected NordEst Innovation Ecosystem, Project ID: ECS 00000043), funded by PNRR (Mission 4.2, Investment 1.5) NextGeneration EU.

References

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2014. [Online]. Available: http://arxiv.org/abs/1312.6199

[2] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. [Online]. Available: https://openreview.net/forum?id=rJzIBfZAb

[3] J. Larson, S. Mattu, L. Kirchner, and J. Angwin, "How we analyzed the COMPAS recidivism algorithm," 2016, available at https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm.

[4] Z. Obermeyer and S. Mullainathan, "Dissecting racial bias in an algorithm that guides health decisions for 70 million people," in *Proceedings of the Conference on Fairness, Accountability, and Transparency, FAT* 2019, Atlanta, GA, USA, January 29-31, 2019*, danah boyd and J. H. Morgenstern, Eds. ACM, 2019, p. 89. [Online]. Available: https://doi.org/10.1145/3287560.3287593

[5] S. Caton and C. Haas, "Fairness in machine learning: A survey," *CoRR*, vol. abs/2010.04053, 2020. [Online]. Available: https://arxiv.org/abs/2010.04053

[6] L. Oneto and S. Chiappa, "Fairness in machine learning," *CoRR*, vol. abs/2012.15816, 2020. [Online]. Available: https://arxiv.org/abs/2012.15816

[7] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning," *ACM Comput. Surv.*, vol. 54, no. 6, pp. 115:1–115:35, 2021. [Online]. Available: https://doi.org/10.1145/3457607

[8] S. Aghaei, M. J. Azizi, and P. Vayanos, "Learning optimal and fair decision trees for non-discriminative decision-making," in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 2019, pp. 1418–1426. [Online]. Available: https://doi.org/10.1609/aaai.v33i01.33011418

[9] F. Ranzato, C. Urban, and M. Zanella, "Fairness-aware training of decision trees by abstract interpretation," in *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, G. Demartini, G. Zuccon, J. S. Culpepper, Z. Huang, and H. Tong, Eds. ACM, 2021, pp. 1508–1517. [Online]. Available: https://doi.org/10.1145/3459637.3482342

[10] A. Ruoss, M. Balunovic, M. Fischer, and M. T. Vechev, "Learning certified individually fair representations," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/55d491cf951b1b920900684d71419282-Abstract.html

[11] P. G. John, D. Vijaykeerthy, and D. Saha, "Verifying individual fairness in machine learning models," in *Proceedings of the Thirty-Sixth Conference on Uncertainty in Artificial Intelligence, UAI 2020, virtual online, August 3-6, 2020*, ser. Proceedings of Machine Learning Research, R. P. Adams and V. Gogate, Eds., vol. 124. AUAI Press, 2020, pp. 749–758. [Online]. Available: http://proceedings.mlr.press/v124/george-john20a.html

[12] C. Urban, M. Christakis, V. Wüstholz, and F. Zhang, "Perfectly parallel fairness certification of neural networks," *Proc. ACM Program. Lang.*, vol. 4, no. OOPSLA, pp. 185:1–185:30, 2020. [Online]. Available: https://doi.org/10.1145/3428253

[13] A. Ignatiev, M. C. Cooper, M. Siala, E. Hebrard, and J. Marques-Silva, "Towards formal fairness in machine learning," in *Principles and Practice of Constraint Programming - 26th International Conference, CP 2020, Louvain-la-Neuve, Belgium, September 7-11, 2020, Proceedings*, ser. Lecture Notes in Computer Science, H. Simonis, Ed., vol. 12333. Springer, 2020, pp. 846–867. [Online]. Available: https://doi.org/10.1007/978-3-030-58475-7_49

[14] S. Galhotra, Y. Brun, and A. Meliou, "Fairness testing: testing software for discrimination," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017, Paderborn, Germany, September 4-8, 2017*, E. Bodden, W. Schäfer, A. van Deursen, and A. Zisman, Eds. ACM, 2017, pp. 498–510. [Online]. Available: https://doi.org/10.1145/3106237.3106277

[15] A. Aggarwal, P. Lohia, S. Nagar, K. Dey, and D. Saha, "Black box fairness testing of machine learning models," in *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2019, Tallinn, Estonia, August 26-30, 2019*, M. Dumas, D. Pfahl, S. Apel, and A. Russo, Eds. ACM, 2019, pp. 625–635. [Online]. Available: https://doi.org/10.1145/3338906.3338937

[16] S. Udeshi, P. Arora, and S. Chattopadhyay, "Automated directed fairness testing," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE 2018, Montpellier, France, September 3-7, 2018*, M. Huchard, C. Kästner, and G. Fraser, Eds. ACM, 2018, pp. 98–108. [Online]. Available: https://doi.org/10.1145/3238147.3238165

[17] E. Black, S. Yeom, and M. Fredrikson, "Fliptest: fairness testing via optimal transport," in *FAT* '20: Conference on Fairness, Accountability, and Transparency, Barcelona, Spain, January 27-30, 2020*, M. Hildebrandt, C. Castillo, L. E. Celis, S. Ruggieri, L. Taylor, and G. Zanfir-Fortuna, Eds. ACM, 2020, pp. 111–121. [Online]. Available: https://doi.org/10.1145/3351095.3372845

[18] H. Khedr and Y. Shoukry, "Certifair: A framework for certified global fairness of neural networks," *CoRR*, vol. abs/2205.09927, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2205.09927

[19] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Wadsworth, 1984.

[20] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[21] C. Schumann, J. S. Foster, N. Mattei, and J. P. Dickerson, "We need fairness and explainability in algorithmic hiring," in *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20, Auckland, New Zealand, May 9-13, 2020*, A. E. F. Seghrouchni, G. Sukthankar, B. An, and N. Yorke-Smith, Eds. International Foundation for Autonomous Agents and Multiagent Systems, 2020, pp. 1716–1720. [Online]. Available: https://dl.acm.org/doi/10.5555/3398761.3398960

[22] S. Verma and J. Rubin, "Fairness definitions explained," in *Proceedings of the International Workshop on Software Fairness, FairWare@ICSE 2018, Gothenburg, Sweden, May 29, 2018*, Y. Brun, B. Johnson, and A. Meliou, Eds. ACM, 2018, pp. 1–7. [Online]. Available: https://doi.org/10.1145/3194770.3194776

[23] K. Makhlouf, S. Zhioua, and C. Palamidessi, "Machine learning fairness notions: Bridging the gap with real-world applications," *Inf. Process. Manag.*, vol. 58, no. 5, p. 102642, 2021. [Online]. Available: https://doi.org/10.1016/j.ipm.2021.102642

[24] Y. Chen, S. Wang, Y. Qin, X. Liao, S. Jana, and D. A. Wagner, "Learning security classifiers with verified global robustness properties," in *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, Y. Kim, J. Kim, G. Vigna, and E. Shi, Eds. ACM, 2021, pp. 477–494. [Online]. Available: https://doi.org/10.1145/3460120.3484776

[25] K. Leino, Z. Wang, and M. Fredrikson, "Globally-robust neural networks," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 2021, pp. 6212–6222. [Online]. Available: http://proceedings.mlr.press/v139/leino21a.html

[26] S. Calzavara, L. Cazzaro, C. Lucchese, F. Marcuzzi, and S. Orlando, "Beyond robustness: Resilience verification of tree-based classifiers," *CoRR*, vol. abs/2112.02705, 2021. [Online]. Available: https://arxiv.org/abs/2112.02705

[27] F. Ranzato and M. Zanella, "Abstract interpretation of decision tree ensemble classifiers," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2020, pp. 5478–5486. [Online]. Available: https://aaai.org/ojs/index.php/AAAI/article/view/5998

[28] J. Törnblom and S. Nadjm-Tehrani, "Formal verification of input-output mappings of tree ensembles," *Sci. Comput. Program.*, vol. 194, p. 102450, 2020. [Online]. Available: https://doi.org/10.1016/j.scico.2020.102450

[29] A. Kantchelian, J. D. Tygar, and A. D. Joseph, "Evasion and hardening of tree ensemble classifiers," in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, ser. JMLR Workshop and Conference Proceedings, M. Balcan and K. Q. Weinberger, Eds., vol. 48. JMLR.org, 2016, pp. 2387–2396. [Online]. Available: http://proceedings.mlr.press/v48/kantchelian16.html

[30] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, J. B. Bocca, M. Jarke, and C. Zaniolo, Eds. Morgan Kaufmann, 1994, pp. 487–499. [Online]. Available: http://www.vldb.org/conf/1994/P487.PDF

[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[32] G. R. Lal, X. Chen, and V. Mithal, "Te2rules: Extracting rule lists from tree ensembles," *CoRR*, vol. abs/2206.14359, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2206.14359

[33] H. Deng, "Interpreting tree ensembles with intrees," *Int. J. Data Sci. Anal.*, vol. 7, no. 4, pp. 277–287, 2019. [Online]. Available: https://doi.org/10.1007/s41060-018-0144-8

[34] M. Mashayekhi and R. Gras, "Rule extraction from random forest: the RF+HC methods," in *Advances in Artificial Intelligence - 28th Canadian Conference on Artificial Intelligence, Canadian AI 2015, Halifax, Nova Scotia, Canada, June 2-5, 2015, Proceedings*, ser. Lecture Notes in Computer Science, D. Barbosa and E. E. Milios, Eds., vol. 9091. Springer, 2015, pp. 223–237. [Online]. Available: https://doi.org/10.1007/978-3-319-18356-5_20

[35] R. Angell, B. Johnson, Y. Brun, and A. Meliou, "Themis: automatically testing software for discrimination," in *Proceedings of the 2018 ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2018, Lake Buena Vista, FL, USA, November 04-09, 2018*, G. T. Leavens, A. Garcia, and C. S. Pasareanu, Eds. ACM, 2018, pp. 871–875. [Online]. Available: https://doi.org/10.1145/3236024.3264590

[36] B. Taskesen, J. H. Blanchet, D. Kuhn, and V. A. Nguyen, "A statistical test for probabilistic fairness," in *FAccT '21: 2021 ACM Conference on Fairness, Accountability, and Transparency, Virtual Event / Toronto, Canada, March 3-10, 2021*, M. C. Elish, W. Isaac, and R. S. Zemel, Eds. ACM, 2021, pp. 648–665. [Online]. Available: https://doi.org/10.1145/3442188.3445927

[37] S. Corbett-Davies and S. Goel, "The measure and mismeasure of fairness: A critical review of fair machine learning," *CoRR*, vol. abs/1808.00023, 2018. [Online]. Available: http://arxiv.org/abs/1808.00023

[38] A. Albarghouthi, L. D'Antoni, S. Drews, and A. V. Nori, "Fairsquare: probabilistic verification of program fairness," *Proc. ACM Program. Lang.*, vol. 1, no. OOPSLA, pp. 80:1–80:30, 2017. [Online]. Available: https://doi.org/10.1145/3133904

[39] O. Bastani, X. Zhang, and A. Solar-Lezama, "Probabilistic verification of fairness properties via concentration," *Proc. ACM Program. Lang.*, vol. 3, no. OOPSLA, pp. 118:1–118:27, 2019. [Online]. Available: https://doi.org/10.1145/3360544

[40] B. Sun, J. Sun, T. Dai, and L. Zhang, "Probabilistic verification of neural networks against group fairness," in *Formal Methods - 24th International Symposium, FM 2021, Virtual Event, November 20-26, 2021, Proceedings*, ser. Lecture Notes in Computer Science, M. Huisman, C. S. Pasareanu, and N. Zhan, Eds., vol. 13047. Springer, 2021, pp. 83–102. [Online]. Available: https://doi.org/10.1007/978-3-030-90870-6_5

[41] B. Ghosh, D. Basu, and K. S. Meel, "Justicia: A stochastic SAT approach to formally verify fairness," in *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 2021, pp. 7554–7563. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/16925

[42] J. Zhou, F. Chen, and A. Holzinger, "Towards explainability for AI fairness," in *xxAI - Beyond Explainable AI - International Workshop, Held in Conjunction with ICML 2020, July 18, 2020, Vienna, Austria, Revised and Extended Papers*, ser. Lecture Notes in Computer Science, A. Holzinger, R. Goebel, R. Fong, T. Moon, K. Müller, and W. Samek, Eds., vol. 13200. Springer, 2020, pp. 375–386. [Online]. Available: https://doi.org/10.1007/978-3-031-04083-2_18

[43] A. Ghosh, A. Shanbhag, and C. Wilson, "Faircanary: Rapid continuous explainable fairness," in *AIES '22: AAAI/ACM Conference on AI, Ethics, and Society, Oxford, United Kingdom, May 19 - 21, 2021*, V. Conitzer, J. Tasioulas, M. Scheutz, R. Calo, M. Mara, and A. Zimmermann, Eds. ACM, 2022, pp. 307–316. [Online]. Available: https://doi.org/10.1145/3514094.3534157

[44] A. Stevens, P. Deruyck, Z. V. Veldhoven, and J. Vanthienen, "Explainability and fairness in machine learning: Improve fair end-to-end lending for kiva," in *2020 IEEE Symposium Series on Computational Intelligence, SSCI 2020, Canberra, Australia, December 1-4, 2020*. IEEE, 2020, pp. 1241–1248. [Online]. Available: https://doi.org/10.1109/SSCI47803.2020.9308371

[45] S. Sharma, J. Henderson, and J. Ghosh, "CERTIFAI: A common framework to provide explanations and analyse the fairness and robustness of black-box models," in *AIES '20: AAAI/ACM Conference on AI, Ethics, and Society, New York, NY, USA, February 7-8, 2020*, A. N. Markham, J. Powles, T. Walsh, and A. L. Washington, Eds. ACM, 2020, pp. 166–172. [Online]. Available: https://doi.org/10.1145/3375627.3375812

[46] S. Wachter, B. D. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the GDPR," *CoRR*, vol. abs/1711.00399, 2017. [Online]. Available: http://arxiv.org/abs/1711.00399

[47] P. A. Grabowicz, N. Perello, and A. Mishra, "Marrying fairness and explainability in supervised learning," in *FAccT '22: 2022 ACM Conference on Fairness, Accountability, and Transparency, Seoul, Republic of Korea, June 21 - 24, 2022*. ACM, 2022, pp. 1905–1916. [Online]. Available: https://doi.org/10.1145/3531146.3533236

[48] Y. Qiang, C. Li, M. Brocanelli, and D. Zhu, "Counterfactual interpolation augmentation (CIA): A unified approach to enhance fairness and explainability of DNN," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, L. D. Raedt, Ed. ijcai.org, 2022, pp. 732–739. [Online]. Available: https://doi.org/10.24963/ijcai.2022/103

[49] E. A. Soares and P. Angelov, "Fair-by-design explainable models for prediction of recidivism," *CoRR*, vol. abs/1910.02043, 2019. [Online]. Available: http://arxiv.org/abs/1910.02043

[50] J. Tan, S. Xu, Y. Ge, Y. Li, X. Chen, and Y. Zhang, "Counterfactual explainable recommendation," in *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, G. Demartini, G. Zuccon, J. S. Culpepper, Z. Huang, and H. Tong, Eds. ACM, 2021, pp. 1784–1793. [Online]. Available: https://doi.org/10.1145/3459637.3482420

[51] Y. Ge, J. Tan, Y. Zhu, Y. Xia, J. Luo, S. Liu, Z. Fu, S. Geng, Z. Li, and Y. Zhang, "Explainable fairness in recommendation," in *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, E. Amigó, P. Castells, J. Gonzalo, B. Carterette, J. S. Culpepper, and G. Kazai, Eds. ACM, 2022, pp. 681–691. [Online]. Available: https://doi.org/10.1145/3477495.3531973

[52] E. Angelino, N. Larus-Stone, D. Alabi, M. I. Seltzer, and C. Rudin, "Learning certifiably optimal rule lists," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*. ACM, 2017, pp. 35–44. [Online]. Available: https://doi.org/10.1145/3097983.3098047

[53] H. Lakkaraju, S. H. Bach, and J. Leskovec, "Interpretable decision sets: A joint framework for description and prediction," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, B. Krishnapuram, M. Shah, A. J. Smola, C. C. Aggarwal, D. Shen, and R. Rastogi, Eds. ACM, 2016, pp. 1675–1684. [Online]. Available: https://doi.org/10.1145/2939672.2939874

[54] M. Mashayekhi and R. Gras, "Rule extraction from decision trees ensembles: New algorithms based on heuristic search and sparse group

lasso methods," *Int. J. Inf. Technol. Decis. Mak.*, vol. 16, no. 6, p. 1707, 2017. [Online]. Available: https://doi.org/10.1142/S0219622017500055

[55] J. Hatwell, M. M. Gaber, and R. M. A. Azad, "CHIRPS: explaining random forest classification," *Artif. Intell. Rev.*, vol. 53, no. 8, pp. 5747–5788, 2020. [Online]. Available: https://doi.org/10.1007/s10462-020-09833-6

[56] C. Bénard, G. Biau, S. D. Veiga, and E. Scornet, "Interpretable random forests via rule extraction," in *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, ser. Proceedings of Machine Learning Research, A. Banerjee and K. Fukumizu, Eds., vol. 130. PMLR, 2021, pp. 937–945. [Online]. Available: http://proceedings.mlr.press/v130/benard21a.html

APPENDIX A
IMPLEMENTATION DETAILS

Another important aspect of the implementation is the treatment of *categorical* features, i.e., features which do not take arbitrary values in $\mathbb{R}$, but may only take values from a set of unordered options, e.g., gender or ethnicity. Categorical features are not part of our model for simplicity, however, they are not difficult to handle in practice. In particular, we handle binary features just like numerical features, since the threshold 0.5 can be used to tell apart the two possible values of the feature. As for other categorical features, we deal with them by using one-hot-encoding, i.e., we generate a new feature f for each possible value of the original feature, with the idea that just one such feature will have a value greater than 0.5. We then enforce that an itemset cannot contain two or more items of the form $x_f > 0.5$ that predicate on features resulting by one-hot-encoding the same categorical feature. Since the categorical features and the result of their one-hot-encoding are known, the implementation of the meet operator checks whether the generated itemsets respect the described integrity condition and the meet is considered undefined in case of violations. The resulting itemsets in the result F are post-processed in order to improve their readability. In particular, we collapse itemsets that differ just for formulas predicating on different values of the same categorical feature by merging these formulas in a single formula using a disjunction.

Finally, we mention a few optimizations of the implementation that we borrowed and adapted from Apriori [30]. The items within an itemset and the itemsets themselves are kept ordered to reduce the number of executions of the meet operator at line 13 and optimize its checks. The items within an itemset are ordered as follows: first, we follow the lexicographic order of the feature involved in the item; then, for any two items involving the same feature, the formulas with predicate $\leq$ precede the ones with predicate $>$; finally, the items with predicate $\leq$ are ordered by decreasing value of the threshold, while we do the opposite for the other items. The order of the items makes it possible to define the prefix of an itemset of size k as the ordered sequence of its first $k-1$ items. Itemsets are ordered to examine itemsets sharing the same prefix consecutively, since only itemsets sharing the same prefix satisfy the preconditions for performing the meet operation. This allows one to reduce the number of times that the meet operation is attempted between two itemsets that do not satisfy the preconditions of the meet definition. In particular, two ordered itemsets of size k are lexicographically sorted using the established order of the items: at first, their prefixes are compared starting from the first item to the $(k-1)$-th item; then, if the two itemsets share the same prefix, they are ordered by comparing their last, different item. Moreover, the order of the items followed by the itemsets sharing the same prefix enables an optimization of the checks of the meet operation. Indeed, for any itemset I' listed after an itemset I sharing the same prefix we have $[\![I]\!] \not\subset [\![I']\!]$. As a result, the implementation of the second condition of the meet $I \sqcap I'$ just needs to check $[\![I']\!] \subset [\![I]\!]$. Another optimization involves the condition at line 15, which would require scanning the entire set U for every generated itemset. We optimize this step by assigning to each $H \in U$ an identifier $id_H \in \mathbb{N}$ and associating to each itemset $I \in C$ a set $ids_I = \{id_H : H \in U \wedge H \cap [\![I]\!] = \emptyset\}$. Then, the set of the identifiers associated to the meet $I = I_1 \sqcap I_2$ of two itemset I_1 and I_2 is $ids_I = ids_{I_1} \cup ids_{I_2}$, since I identifies less instances than both I_1 and I_2 by definition. This information is useful, because the condition at line 15 amounts just to checking whether $|ids_I| = |U|$, rather than scanning U.

TABLE III: Top 5 logic formulas obtained for the German dataset (ensemble with 13 trees of maximum depth six). The formulas are ordered by decreasing importance.

Rank	Formula
1	status = "no checking account" $\wedge$ savings $\neq$ "unknown/no savings account" $\wedge$ installment_plans = "None"
2	250.00 $\leq$ credit_amount $\leq$ 7,464.50 $\wedge$ status = "no checking account" $\wedge$ (credit_history = "no credits/all credits paid back duly" $\vee$ credit_history = "existing credits paid back duly till now" $\vee$ credit_history = "delay in paying off in the past")
3	status = "no checking account" $\wedge$ credit_history $\neq$ "critical account/other credits existing" $\wedge$ savings $\neq$ "unknown/no savings account"
4	250.00 $\leq$ credit_amount $\leq$ 7,464.50 $\wedge$ status = "no checking account" $\wedge$ credit_history = "critical account/other credits existing" $\wedge$ installment_plans = "None"
5	telephone = False $\wedge$ status = "no checking account" $\wedge$ (credit_history = "all credits at this bank paid back duly" $\vee$ credit_history = "existing credits paid back duly till now" $\vee$ credit_history = "delay in paying off in the past")

APPENDIX B
CASE STUDY: GERMAN DATASET

In this section, we present some examples of logic formulas generated by our synthesis algorithm to show that they provide effective explanations about the fairness of the model.

We consider logic formulas synthesized for a tree ensemble with 13 trees of maximum depth six, trained on the German dataset. This dataset consists of 1,000 people who would like to get credit from a bank and the corresponding classification task consists in classifying their requests as high or low risk. We consider *sex* as the sensitive feature, so the logic formulas provide us the description of subsets of people whose credit risk prediction does not change by flipping their sex.

We present the top 5 most important formulas generated after six iterations of the synthesis algorithm in Table III, in decreasing order of importance. We immediately observe that the formulas are explainable, since they predicate on at most four features. We then comment three representative formulas below.

The first-ranked logic formula is interesting to examine, since it covers a subset of people that we expect to be highly represented also in the test set, because the formulas are ranked by the importance computed on the training set (see Section IV). In particular, it indicates that an individual that has no checking account, possesses a saving account whose amount is registered with the bank and has no installment plans will not receive a different credit risk depending on their sex. The formula suggests that the person's savings represent enough information for assigning the credit risk irrespective of the sex of the requester, at least when no information about the individual's checking account or other installment plans is available.

The second formula is interesting too. In particular, the formula indicates that the model does not discriminate by sex individuals who request a small credit amount (between 250 and $\sim$ 7,500 DM), do not have a registered checking account and do not present a critical credit history. The formula highlights a reasonable behavior of the ML classifier: when requesting small credit amounts, the presence of a good credit history already suffices for taking a decision, without relying also on the individual's sex.

Finally, the fourth formula is the only one in the top 5 formulas that predicates on individuals with a bad credit history. In particular, it explains that an individual who requests a small credit amount, has no checking account and other installment plans, but presents a critical credit history, is not discriminated by the classifier based on his sex. This is interesting because bad credit history should be the most important information when assessing a loan request. The reason why this only emerges for small loan requests might be twofold. First, assessing requests for small loans is expected to be easier, hence the corresponding proof of fairness is also easier and is established with a small number of analysis iterations. Moreover, the dataset includes many such requests, hence the importance of the formula increases and leads to its inclusion in the top formulas.

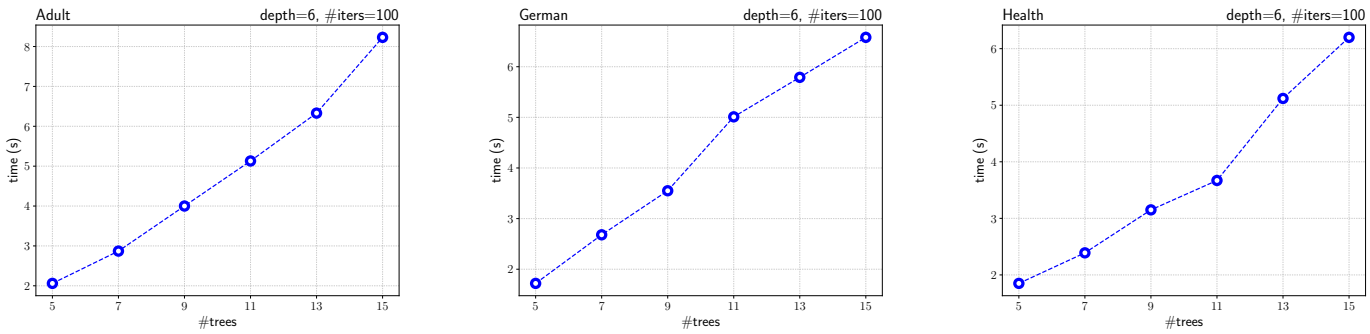
In conclusion, the synthesized logic formulas are useful to the analyst to conclude whether fairness guarantees can be provided for particular subgroups of instances in the domain of interest, not just for specific instances in the test set.

Appendix C
Performance of the Stability Analysis

We present some additional results about the time required by the stability analysis proposed in [26]. The analysis is based on an iterative algorithm that supports iterative refinements in order to deal with the exponential complexity characterizing the problem of verifying decision tree ensembles [29]. In our experiments, we fix the number of analysis iterations to 100, since a limited number of iterations already suffices to obtain a reasonably accurate over-approximation of the subset of the feature space where the ensemble is unstable [26]. We evaluate the performance of the stability analysis when varying the number of trees in the ensemble, as well as their depth, on the three datasets used in Section IV to assess our proposal.

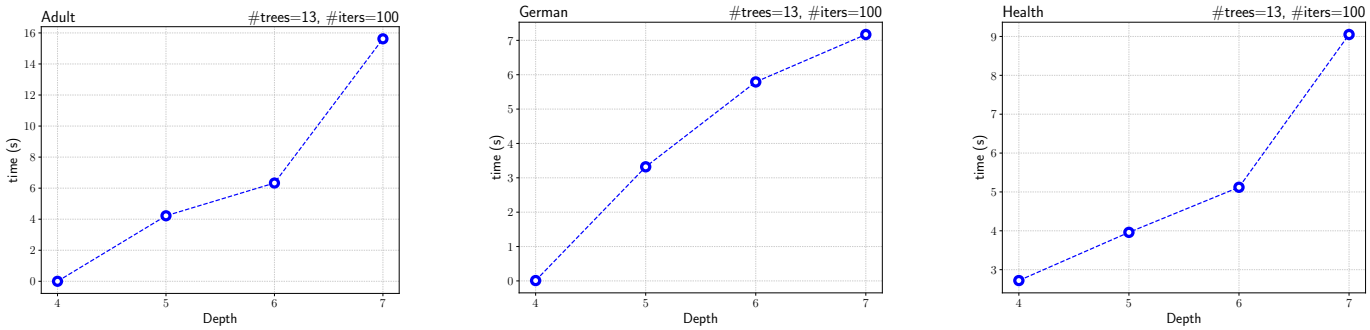
The experimental results are shown in Figure 8 and Figure 9, which show how the running time of the stability analysis changes when increasing the number of trees and their maximum depth respectively. For all the considered datasets and models, the analysis for 100 iterations terminates in a matter of seconds, thus showing the practicality of the stability analysis for the considered cases. Even though the analysis of models trained on the Adult dataset requires more time than analyzing the same models trained on the other two datasets, the stability analysis takes at most 16 seconds on the ensemble of 13 trees with maximum depth seven.

Note that the time required by the stability analysis normally represents a small amount of the time required by the synthesis algorithm developed in our work. For example, consider the case of an ensemble of 13 trees with maximum depth six, trained on the Adult dataset: the stability analysis just takes six seconds, while the synthesis algorithm takes more than 30 minutes (to generate conditions of length six). We do not try to further improve the precision of the stability analysis, since the main goal of our experimental evaluation is to assess the effectiveness of the synthesis algorithm. However, these results show that it would be certainly possible to replace the current implementation of the stability analysis with a more expensive, yet more precise alternative.



(a) (b) (c)

Fig. 8: Running times of the stability analysis when varying the number of the decision trees in the ensemble.



(a) (b) (c)

Fig. 9: Running times of the stability analysis when varying the maximum depth of the decision trees in the ensemble.