Towards Formal Fairness in Machine Learning[⋆]

Alexey Ignatiev[1], Martin C. Cooper[2,4], Mohamed Siala[3,4],
Emmanuel Hebrard[3,4], and Joao Marques-Silva[4]

[1] Monash University, Australia
alexey.ignatiev@monash.edu
[2] IRIT, Université de Toulouse III, France
cooper@irit.fr
[3] LAAS-CNRS, Université de Toulouse, CNRS, INSA, France
{siala,hebrard}@laas.fr
[4] ANITI, Université de Toulouse, France
joao.marques-silva@univ-toulouse.fr

Abstract. One of the challenges of deploying machine learning (ML) systems is fairness. Datasets often include sensitive features, which ML algorithms may unwittingly use to create models that exhibit unfairness. Past work on fairness offers no formal guarantees in their results. This paper proposes to exploit formal reasoning methods to tackle fairness. Starting from an intuitive criterion for fairness of an ML model, the paper formalises it, and shows how fairness can be represented as a decision problem, given some logic representation of an ML model. The same criterion can also be applied to assessing bias in training data. Moreover, we propose a reasonable set of axiomatic properties which no other definition of dataset bias can satisfy. The paper also investigates the relationship between fairness and explainability, and shows that approaches for computing explanations can serve to assess fairness of particular predictions. Finally, the paper proposes SAT-based approaches for learning fair ML models, even when the training data exhibits bias, and reports experimental trials.

1 Introduction

Given the forecast widespread use of ML-enabled systems, in settings that can have a significant impact in the lives and safety of human beings, a range of concerns need to be addressed. Robustness of ML models against adversarial examples is one such concern [22, 37, 43, 44, 50, 55, 60, 61]. Explaining the predictions of ML models represents another concern. A related concern is to learn (or synthesize) interpretable ML models [5,9,30,35,36,41,51,56,65,66]. One additional concern is to ensure that ML-enabled systems are fair [4, 15, 48]. The importance of addressing fairness cannot be overstated, as demonstrated by recent existing troublesome evidence [6,19] that already deployed ML-enabled systems can exhibit very significant bias. Furthermore, recently recommended guidelines at the EU level highlight the importance of fairness [23].

We find many definitions of fairness in the literature [7, 54, 64]. In [64], the authors classify these definitions into three categories: statistical measures, similarity-based measures, and causal reasoning. Statistical measures, such as those presented

[⋆] This work was partially funded by ANITI, funded by the French program "Investing for the Future – PIA3" under Grant agreement n^o ANR-19-PI3A-0004.

in [7, 14, 16, 20, 33, 46], can hardly be studied from a formal angle due to their nature. Similarity-based measures [20, 27, 48], on the other hand, are data independent definitions that are well suited for formal investigations. Last, causal reasoning measures [45, 48, 53] are based on the so-called causal graphs (i.e. graphs capturing relationships between the different features) and are essentially used to synthesise ML models. We focus in this paper on similarity-based measures because they offer an excellent framework for formal analysis compared to the other two.

Addressing fairness can be performed at three different levels: data processing, synthesis, and verification. Current work on the three levels is, to a large extent, heuristic in nature [1,2,4,11,25,26,28,34,48,59,64], offering no formal guarantees in their analyses. This paper proposes a first step towards endowing the analysis (i.e. data processing and verification) and the synthesis of fair ML models with a rigorous footing. We study one concrete criterion of fairness, and propose a rigorous test to assess whether or not an ML model is fair against that criterion. Moreover, the paper shows how the proposed test can be adapted to devise a simple (polynomial-time) algorithm to assess existing bias in datasets, even if datasets are inconsistent. More importantly, in the case of biased datasets, the paper investigates how to adapt exact methods for learning interpretable (logic-based) ML models to synthesize fair ML models.

The paper is organized as follows. Section 2 summarizes the definitions and notation used throughout the paper. Section 3 focuses on a criterion of fairness, and develops tests for assessing whether an ML model is fair, whether a dataset exhibits bias, and whether a particular prediction is fair. Section 4 investigates how fair logic-based models can be synthesized with possibly biased datasets. Section 5 provides a theoretical justification for adopting the fairness criterion used throughout the paper. Finally, Section 6 presents preliminary experimental results and Section 7 concludes the paper.

2 Preliminaries

SAT/SMT-related topics. The paper uses definitions standard in Boolean Satistiability (SAT) and Satisfiability Modulo Theories (SMT) [10]. These include conjunctive and disjunctive normal forms (resp. CNF and DNF), prime implicants and implicates.

Classification problems. This section adapts the definitions used in earlier work [9,41, 49]. We consider a set of features $\mathcal{F} = \{F_1, \ldots, F_K\}$, where each feature F_i takes values from some domain D_i. The space of all assignments to features defined by $\mathbb{F} = \Pi_{i=1}^{K} D_i$ is referred to as *feature space* [32]. Throughout the paper, all domains in the examples and experiments will be binary, i.e. $D_i = \{0, 1\}$, but the results will be derived assuming arbitrary (discrete) domains[5]. Since all features are binary, a literal on a feature F_r will be represented as F_r or as $\neg F_r$.

To learn a classifier, one starts from given training data (also referred to as examples) $\mathcal{T} = \{e_1, \ldots, e_M\}$. Each example has an associated class taken from a set of classes $\mathcal{C}$. The paper focuses mostly on binary classification, i.e. $\mathcal{C} = \{c_0, c_1\}$. (We will associate c_0 with 0 and c_1 with 1, for simplicity.) Thus, $\mathcal{T}$ is partitioned into $\mathcal{T}^+$ and $\mathcal{T}^-$, denoting the examples classified as positive ($c_1 = 1$) and as negative ($c_0 = 0$), respectively. Each example $e_q \in \mathcal{T}$ is represented as a pair $\langle \mathbf{z}_q, c_q \rangle$, where $\mathbf{z}_q \in \mathbb{F}$ denotes the literals associated with the example and $c_q \in \{0, 1\}$ is the class to which the example belongs.

[5] Real-value features can be discretized. Moreover, to focus on binary features, the fairly standard one-hot-encoding [58] is assumed for handling non-binary categorical features.

We have $c_q = 1$ if $e_q \in \mathcal{T}^+$ and $c_q = 0$ if $e_q \in \mathcal{T}^-$. The training data $\mathcal{T}$ is consistent if $\langle \mathbf{z}_q, 0\rangle \in \mathcal{T} \wedge \langle \mathbf{z}_{q'}, 1\rangle \in \mathcal{T} \implies \mathbf{z}_q \neq \mathbf{z}_{q'}$, and inconsistent otherwise. A literal l_r on a feature F_r, $l_r \in \{F_r, \neg F_r\}$, *discriminates* an example e_q if $\mathbf{z}_q[r] = \neg l_r$, i.e. the feature takes the value opposite to the value in the example. We assume that all features are specified for all examples; the work can be generalized for situations where the value of some features for some examples is left unspecified.

An ML model $\mathbb{M}$ is represented as a function $\varphi : \mathbb{F} \to \mathcal{C}$. With a slight abuse of notation, a consistent training data will also be viewed as a partial function $\varphi_{\mathcal{T}} : \mathbb{F} \to \mathcal{C}$. In a general setting, where the training data can be inconsistent, we represent the possible values of training data in each point of feature space by a relation $\mathfrak{T} \subseteq \mathbb{F} \times \mathcal{C}$.

Unless otherwise stated, we focus on *accurate* ML models, indicating that 100% of the training data examples are classified correctly. A non-accurate ML model may misclassify some examples of training data (e.g. this is the case for inconsistent data).

Examples of ML models. The paper focuses almost exclusively on logic-based ML models, namely decision sets (DSs) and trees (DTs).

Definition 1 (Decision set). *A DNF formula ϕ over the literals $\bigcup_{F_r \in \mathcal{F}}\{F_r, \neg F_r\}$ is a decision set for a training set $\mathcal{T}$ if the function that maps $\mathbf{z} \in \mathbb{F}$ to c_1 if $\mathbf{z}$ is a model of ϕ and to c_0 otherwise is equal to $\varphi_{\mathcal{T}}$ on $\mathcal{T}$.*

Definition 2 (Decision tree). *A decision tree for a training set $\mathcal{T}$ is a decision set $\phi = \bigvee_i t_i$ such that there exists a rooted tree with edges labelled with literals of ϕ and such that for every term t_i of ϕ there is a path from the root to a leaf whose set of labels consists of exactly the literals in t_i.*

Fairness. Following standard notation [48], throughout this paper we will assume that $\mathcal{F}$ is partitioned into a set of *protected* features $\mathcal{P} = \{F_{I+1}, \ldots, F_K\}$ and a set of *non-protected* features $\mathcal{N} = \{F_1, \ldots, F_I\}$, with $|\mathcal{N}|= I$ and $|\mathcal{P}|= K - I$. Thus, following the notation used earlier, we define $\mathbb{N} = \Pi_{i=1}^{I} D_i$ and $\mathbb{P} = \Pi_{i=I+1}^{K} D_i$, and so $\mathbb{F} = \mathbb{N}\times\mathbb{P}$. Moreover, $\mathbf{z} \in \mathbb{F}$ is split into $\mathbf{x} \in \mathbb{N}$ and $\mathbf{y} \in \mathbb{P}$. The protected features are those with respect to which we intuitively want ML models *not* to be *sensitive*[6]. Throughout the paper, we will use the notation $\varphi(\mathbf{x}, \mathbf{y})$ as a replacement for $\varphi(\mathbf{x} \cdot \mathbf{y})$ where $\mathbf{z} = \mathbf{x} \cdot \mathbf{y}$ denotes the concatenation of vectors. Moreover, when referring to *specific* points in feature space (or the space of non-protected or protected features), the notation used will be $\mathbf{t} \in \mathbb{F}$, $\mathbf{u} \in \mathbb{N}$, and $\mathbf{v} \in \mathbb{P}$ (resp. instead of $\mathbf{z}$, $\mathbf{x}$ and $\mathbf{y}$). Finally, we use the notation $\mathbb{T}$ to denote the set $\{\mathbf{t}_i \mid \langle \mathbf{t}_i, c_i\rangle \in \mathcal{T}\} \subseteq \mathbb{F}$.

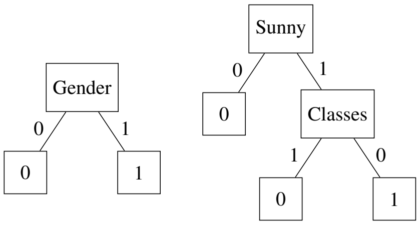
Running example. We use a simple example to illustrate the main points. Minor modifications to the original example will be also considered later in the paper.

Example 1. We consider the example of Figure 1, where the protected feature set is $\mathcal{P} = \{\mathsf{Gender}\}$. The purpose of the example is to decide the circumstances that cause students to enjoy a hike, and we would rather have a classifier that is gender-balanced, if possible. (For the purposes of the example, it is irrelevant whether the values of 0 and 1 of feature Gender correspond to male and female or vice-versa.) By running an off-the-shelf heuristic algorithm for constructing a decision tree, one obtains a single branching node with feature Gender. Figure 1b shows the heuristic decision tree obtained with well-known ML packages (scikit-learn [58] and Orange [18] gave the same result).

[6] For a number of reasons, datasets can contain such protected features, but their removal may be undesirable, for example, because this may induce inconsistencies in datasets.

Ex.	Sunny	Windy	Classes	Exams	Gender	Hike?
e_1	1	0	0	0	1	1
e_2	0	1	0	0	0	0
e_3	1	1	1	0	0	0
e_4	0	1	0	1	0	0
e_5	1	1	0	0	1	1

(a) Dataset



(b) Unfair decision tree (c) Fair decision tree

Fig. 1: Running example

Nevertheless, the evident unfairness of the obtained decision tree results *solely* from the algorithms used for constructing heuristic decision trees. Indeed, as shown in later sections, careful analysis of the training data reveals that there is *no* evidence in the data that justifies that feature Gender should play such a role in deciding the circumstances under which male/female students enjoy hikes. Figure 1c shows an example of a fair decision tree for this example corresponding to the single-term DNF Sunny $\wedge\neg$Classes.

3 Assessing Fairness

We consider in this section an ML classification scenario, with training data $\mathcal{T}$ where the set of features $\mathcal{F}$ is partitioned into a set of protected features $\mathcal{P}$ and a set of non-protected features $\mathcal{N}$, and some (interpretable) ML model $\mathbb{M}$ trained on $\mathcal{T}$

3.1 Fairness Criterion

A number of definitions of fairness have been proposed in recent years [48,64]. This paper considers *fairness through unawareness* (FTU), which was originally proposed as follows:

Criterion 1 (Fairness Through Unawareness (FTU) [31,48]**).** An algorithm is fair if the protected features $\mathcal{P}$ are not explicitly used in the decision-making process.

The operational definition above suggests a syntactic test to decide whether FTU holds. This section proposes instead a semantic characterization of fairness, that respects the syntactic definition of FTU. Besides FTU, a number of additional definitions of fairness are studied in [48,64] and in related work. Moreover, a possible criticism of FTU is that $\mathcal{P}$ may not represent all features capturing discriminatory information [48]. Nevertheless, FTU exhibits a number of advantages over other criteria, and Section 5 provides a theoretical justification for using FTU as a definition of fairness.

Definition 3 (Criterion for FTU). *Given an ML model $\mathbb{M}$ computing some (classification) function $\varphi : \mathbb{F} \to \mathcal{C}$, we say that $\mathbb{M}$ is* fair *if:*

$$\forall(\mathbf{x} \in \mathbb{N})\, \forall(\mathbf{y}_1, \mathbf{y}_2 \in \mathbb{P}).\, [\varphi(\mathbf{x}, \mathbf{y}_1) = \varphi(\mathbf{x}, \mathbf{y}_2)] \tag{1}$$

Next, we investigate how this criterion can be used to analyze both ML models and datasets. Note that this criterion can be seen as a hard version of the causal discrimination measure presented in [27].

3.2 Assessing Model Fairness and Dataset Bias

Checking model fairness. As shown later in the paper, it will be convenient to assess instead the negation of the FTU criterion.

Remark 1. An ML model $\mathbb{M}$ respects Definition 3, i.e. (1) holds, iff the following is false:

$$\exists(\mathbf{x} \in \mathbb{N})\, \exists(\mathbf{y}_1, \mathbf{y}_2 \in \mathbb{P}).\, [\varphi(\mathbf{x}, \mathbf{y}_1) \neq \varphi(\mathbf{x}, \mathbf{y}_2)] \tag{2}$$

Clearly, we can now use (2) to assess whether an ML model $\mathbb{M}$ is fair or not, by searching for satisfying assignments for (2), and this can be achieved with a satisfiability test, given a suitable logic representation of the ML model $\mathbb{M}$. From a practical perspective, we can refine the previous result as follows.

Remark 2. To test condition (2) we only need to test pairs $\mathbf{y}_1, \mathbf{y}_2$ which differ on a single feature. This is because if (2) holds for some $\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2$ then it must hold for some $\mathbf{x}, \mathbf{y}^r, \mathbf{y}^{r+1}$ ($r \in \{0, \ldots, K-1\}$) where $\mathbf{y}^r$ is equal to $\mathbf{y}_1$ on the first $K-r$ features of $\mathbb{P}$ and equal to $\mathbf{y}_2$ on the other features of $\mathbb{P}$. Analyzing each feature separately reduces the search space that needs to be considered.

Checking bias in consistent datasets. By exploiting Remark 1 or Remark 2, we can devise a test to assess whether a dataset exhibits unfairness (in which case we say that the dataset is *biased*). We consider first the case when the dataset is consistent, and use the insights to consider the more general case of inconsistent datasets. For a consistent dataset, the following condition captures FTU in the dataset.

Definition 4 (Consistent dataset bias under FTU). *A consistent dataset $\mathcal{T}$ is biased if the following holds:*

$$\exists(\mathbf{x} \in \mathbb{N})\, \exists(\mathbf{y}_1, \mathbf{y}_2 \in \mathbb{P}).[(\mathbf{x}, \mathbf{y}_1), (\mathbf{x}, \mathbf{y}_2) \in \mathbb{T} \wedge (\varphi_{\mathcal{T}}(\mathbf{x}, \mathbf{y}_1) \neq \varphi_{\mathcal{T}}(\mathbf{x}, \mathbf{y}_2))] \tag{3}$$

Intuitively, a dataset is biased if the protected features serve to distinguish between two different predictions when the non-protected features take the same values. Although (3) is harder to read than (2), it is actually simple to develop a polynomial time procedure for assessing whether a dataset is FTU-biased. However, we develop instead a polynomial (indeed, linear) time algorithm for the more general case of inconsistent data, which is also applicable in the case of consistent data.

Checking bias in inconsistent datasets. Even if the training data is inconsistent, one can devise a test to assess whether a dataset exhibits bias. As motivated in Section 2, in the presence of inconsistent data, we model the expected input-output behavior (given the dataset) as a relation. In the case of an inconsistent dataset, the following condition captures FTU in the dataset.

Definition 5 (Inconsistent dataset bias under FTU). *An inconsistent dataset $\mathcal{T}$ is biased if the following holds:*

$$\begin{aligned} &\exists(\mathbf{x} \in \mathbb{N})\, \exists(\mathbf{y}_1, \mathbf{y}_2 \in \mathbb{P})\, \exists(c_1, c_2 \in \mathcal{C}).[\mathbf{y}_1 \neq \mathbf{y}_2 \wedge c_1 \neq c_2 \wedge \\ &\qquad (\mathbf{x}, \mathbf{y}_1), (\mathbf{x}, \mathbf{y}_2) \in \mathbb{T} \wedge \mathfrak{T}(\mathbf{x}, \mathbf{y}_1, c_1) \wedge \mathfrak{T}(\mathbf{x}, \mathbf{y}_2, c_2)] \end{aligned} \tag{4}$$

There is an alternative definition that considers the dataset to be fair if for each $\mathbf{x} \in \mathbb{N}$, $\mathbf{y}_1, \mathbf{y}_2 \in \mathbb{P}$, the *sets* of classes $c \in \mathcal{C}$ for which $\mathfrak{T}(\mathbf{x}, \mathbf{y}_1, c)$ or $\mathfrak{T}(\mathbf{x}, \mathbf{y}_2, c)$ hold are identical. We ruled out this alternative definition for reasons described in Section 5.

There is a fairly simple linear-(amortized)-time algorithm that can be used both with consistent and with inconsistent data, as shown in Algorithm 1. Correctness of the algorithm follows from the fact that condition (4) is actually logically equivalent to

$$\begin{aligned} &\exists(\mathbf{x} \in \mathbb{N})\exists(\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_1', \mathbf{y}_2' \in \mathbb{P})\, \exists(c_1, c_2, c_1', c_2' \in \mathcal{C}).[\mathbf{y}_1 \neq \mathbf{y}_2 \wedge c_1 \neq c_2 \wedge \\ &\qquad (\mathbf{x}, \mathbf{y}_1), (\mathbf{x}, \mathbf{y}_2), (\mathbf{x}, \mathbf{y}_1'), (\mathbf{x}, \mathbf{y}_2') \in \mathbb{T} \wedge \\ &\qquad \mathfrak{T}(\mathbf{x}, \mathbf{y}_1, c_1') \wedge \mathfrak{T}(\mathbf{x}, \mathbf{y}_2, c_2') \wedge \mathfrak{T}(\mathbf{x}, \mathbf{y}_1', c_1) \wedge \mathfrak{T}(\mathbf{x}, \mathbf{y}_2', c_2)] \end{aligned} \tag{5}$$

Algorithm 1: Checking dataset bias

```
Input: $\mathcal{T}, \mathcal{N}, \mathcal{P}$
Output: Biased / Unbiased
begin
    foreach $\langle(\mathbf{x}_i, \mathbf{y}_i), c_i\rangle \in \mathcal{T}$ do
        $\mathsf{CSet}[\mathbf{x}_i] \leftarrow \emptyset$
        $\mathsf{YSet}[\mathbf{x}_i] \leftarrow \emptyset$
    foreach $\langle(\mathbf{x}_i, \mathbf{y}_i), c_i\rangle \in \mathcal{T}$ do
        $\mathsf{CSet}[\mathbf{x}_i] \leftarrow \mathsf{CSet}[\mathbf{x}_i] \cup \{c_i\}$ ;
        $\mathsf{YSet}[\mathbf{x}_i] \leftarrow \mathsf{YSet}[\mathbf{x}_i] \cup \{\mathbf{y}_i\}$
    foreach $\langle(\mathbf{x}_i, \mathbf{y}_i), c_i\rangle \in \mathcal{T}$ do
        if $|\mathsf{CSet}[\mathbf{x}_i]| > 1 \land |\mathsf{YSet}[\mathbf{x}_i]| > 1$ then
            return Biased
    return Unbiased
end
```

Ex.	Sunny	Windy	Classes	Exams	Gender	Hike?
e_1	1	0	0	0	1	1
e_6	1	0	0	0	0	0
e_7	1	0	0	0	1	0
e_8	1	0	0	0	0	1

Table 1: Extension of the dataset of our running example

The implication (4) $\Rightarrow$ (5) is immediate. To see the implication (5) $\Rightarrow$ (4), suppose that $\mathfrak{T}(\mathbf{x}, \mathbf{y}_1, c_1') \land \mathfrak{T}(\mathbf{x}, \mathbf{y}_2, c_2') \land \mathfrak{T}(\mathbf{x}, \mathbf{y}_1', c_1) \land \mathfrak{T}(\mathbf{x}, \mathbf{y}_2', c_2)$ where $\mathbf{y}_1 \neq \mathbf{y}_2, c_1 \neq c_2$, but that (4) does not hold. We can deduce that $c_1' = c_2'$ and $\mathbf{y}_1' = \mathbf{y}_2'$ and $(\mathbf{y}_1 = \mathbf{y}_1' \lor c_1' = c_1)$, $(\mathbf{y}_1 = \mathbf{y}_2' \lor c_1' = c_2)$, $(\mathbf{y}_2 = \mathbf{y}_1' \lor c_2' = c_1)$, $(\mathbf{y}_2 = \mathbf{y}_2' \lor c_2' = c_2)$ for which it can easily be verified that there is no solution.

In Algorithm 1 each vector $\mathbf{x}$ on the non-protected features is used for indexing both sets CSet, YSet using hashtables. By inspection, the amortized running time is linear (since operations on a hash table have constant amortized complexity [17]).

Example 2. Running the proposed algorithm on the dataset of Example 1 with protected feature set $Y = \{\mathsf{Gender}\}$ confirms that the dataset is unbiased. However, if the same dataset is extended with the rows in Table 1 (where e_1 is added for convenience), then the algorithm reports, as expected, that the dataset is no longer unbiased. Clearly, with $\mathbf{x}_1 = (1, 0, 0, 0) = \mathbf{x}_6 = \mathbf{x}_7 = \mathbf{x}_8$, there are now two different predictions and the dataset is inconsistent. Moreover, there are two reasons for the dataset to be deemed biased: e_1 and e_6 are one reason, and e_7 and e_8 are the other. It should also be noted that e_1 and e_7 do *not* represent a possible reason to declare bias in the dataset.

One way to tackle fairness is to discard the protected features. The following simple result will be used later in the paper.

Proposition 1. Consider a consistent dataset $\mathcal{T}$, and let the dataset $\mathcal{T}'$ be obtained by discarding the protected features $\mathcal{P}$ of $\mathcal{T}$. Hence, $\mathcal{T}'$ may have examples with duplicated sets of feature values. $\mathcal{T}$ is unbiased iff there are no inconsistencies in $\mathcal{T}'$.

3.3 Local Fairness via Explanations

In this section we consider local notions of fairness. An individual is probably more interested in the fairness of a particular decision concerning themselves than in the global fairness of the model. We use the notion of explanation to define local fairness. It turns out that there are two possible definitions of local fairness based on explanations.

To be concrete, consider the problem of an unemployed woman who has been refused a loan and who wants to know if this is because she is a woman. Suppose the bank has learned the following simple model: refuse a loan if the client is unemployed or if they are a woman. This model is clearly unfair with respect to gender, but in this particular case the bank can claim that they would have refused the loan even if the client had been a man. On the other hand, the client can point out there are two explanations for the refusal: the first explanation is that she is unemployed (since all unemployed are refused a loan) and the second explanation is that she is a woman (since all women are refused a loan), and hence the decision should be considered unfair.

There is recent work investigating similar themes [3] which shows that explanations can be used to fake fairness, but the authors study statistical measures of fairness.

Following [39,62], given an ML model $\mathbb{M}$ computing some function φ, an explanation of some prediction $\varphi(\mathbf{z}) = c$ is a prime implicant of the mapping $\varphi : \mathbf{z} \mapsto c$, where c is considered fixed (i.e. a *subset-minimal* subset of the literals of $\mathbf{z}$ which still entails the prediction c). This notion of explanation allows us to define fairness of a particular decision/prediction of a model $\mathbb{M}$.

In the following we view a $\mathbf{z} \in \mathbb{F}$ as a set of literals, one per feature.

Definition 6 (Fair explanation). *An explanation* $\mathbf{e}$ *of a prediction* $\varphi(\mathbf{z}) = c$ *is a subset of* $\mathbf{z}$ *which is minimal under subset inclusion such that* $\forall \mathbf{z}' \in \mathbb{F}$*, if* $\mathbf{e} \subseteq \mathbf{z}'$ *then* $\varphi(\mathbf{z}') = c$*. If* $\mathbf{z} = (\mathbf{x}, \mathbf{y})$ *with* $x \in \mathbb{N}$*,* $y \in \mathbb{P}$*, we say that* $\mathbf{e}$ *is fair if* $\mathbf{e} \cap \mathbf{y} = \emptyset$ *(i.e.* $\mathbf{e}$ *uses no protected literals).*

Definition 7 (Universal/existential Fairness). *A prediction* $\varphi(\mathbf{z}) = c$ *is universally fair if all of its explanations are fair. It is existentially fair if at least one of its explanations is fair.*

It turns out that there is a close connection between FTU and universal fairness.

Proposition 2. Let φ be the function computed by a ML model $\mathbb{M}$. $\mathbb{M}$ is fair according the FTU criterion (1) iff all predictions $\varphi(\mathbf{z}) = c$ ($\mathbf{z} \in \mathbb{F}$) are universally fair.

Proof. It suffices to prove that $\mathbb{M}$ is unfair according to criterion (2) iff there exists an unfair explanation of some prediction.

Suppose that $\mathbb{M}$ is unfair because $\varphi(\mathbf{x}, \mathbf{y}_1) = c \neq \varphi(\mathbf{x}, \mathbf{y}_2)$. All predictions have at least one explanation, so let $\mathbf{e}$ be an explanation of $\varphi(\mathbf{x}, \mathbf{y}_1) = c$. Thus, by Definition 6, $\forall \mathbf{z}' \in \mathbb{F}$, if $\mathbf{e} \subseteq \mathbf{z}'$ then $\varphi(\mathbf{z}') = c$. Since $\varphi(\mathbf{x}, \mathbf{y}_2) \neq c$, this implies that $\mathbf{e} \cap \mathbf{y_1} \neq \emptyset$ and hence $\mathbf{e}$ is an unfair explanation.

Suppose that the prediction $\varphi(\mathbf{z}) = c$ (where $\mathbf{z} = (\mathbf{x}, \mathbf{y})$) has an unfair explanation $\mathbf{e}$. Let $\mathbf{e}' = \mathbf{e} \setminus \mathbf{y}$. Since $\mathbf{e}$ is unfair, $\mathbf{e} \cap \mathbf{y} \neq \emptyset$ and so $\mathbf{e}'$ is a proper subset of $\mathbf{e}$. By subset minimality of the explanation $\mathbf{e}$, $\mathbf{e}'$ cannot be a valid explanation of $\varphi(\mathbf{z}) = c$, and so $\exists \mathbf{z}' \in \mathbb{F}$ such that $\mathbf{e}' \subseteq \mathbf{z}'$ and $\varphi(\mathbf{z}') \neq c$. Let $\mathit{feat}(\mathbf{e})$ denote the features which occur in $\mathbf{e}$ and $\mathbf{z}'[\mathit{feat}(\mathbf{e})]$ the subset of $\mathbf{z}'$ on these features. Now, let $\mathbf{z}'' = \mathbf{e} \cup (\mathbf{z}' \setminus \mathbf{z}'[\mathit{feat}(\mathbf{e})])$. Since $\mathbf{e}$ is an explanation of $\varphi(\mathbf{z}) = c$ and $\mathbf{e} \subseteq \mathbf{z}''$, we must have $\varphi(\mathbf{z}'') = \varphi(\mathbf{z}) = c$. But then $\mathbf{z}''$ and $\mathbf{z}'$ differ only on the features of $\mathbf{e} \cap \mathbf{y}$ (i.e. on protected features) but $\varphi(\mathbf{z}'') = c \neq \varphi(\mathbf{z}')$, so $\mathbb{M}$ is unfair according to criterion (2). □

Using our rigourous definition of explanation and our two notions of fairness of a prediction, we consider two concrete questions, given a prediction $\varphi(\mathbf{z}) = c$:

1. Are all explanations fair (i.e. do not include any feature from the set of protected features)? This problem will be referred to as *universal fairness checking* (UFC).
2. Does there exist a fair explanation (i.e. that does not include any protected feature)? This problem will be referred to as *existential fairness checking* (EFC).

These two problems correspond to our two different notions of fairness of a prediction (existential and universal). They clearly differ semantically, but it would appear that they also differ in terms of the computational complexity to answer them.

Proposition 3. For polytime-computable φ, EFC $\in$ co-NP and UFC $\in \Pi_2^P$.

Proof. To see EFC $\in$ co-NP, observe that a prediction $\varphi(\mathbf{x}, \mathbf{y}) = c$ has a fair explanation iff the non-protected features $\mathbf{x}$ entail the predicted class c. Thus the non-existence of a fair explanation is equivalent to the existence of $\mathbf{y}' \in \mathbb{P}$ such that $\varphi(\mathbf{x}, \mathbf{y}') \neq c$.

To see UFC $\in \Pi_2^P$, observe that all explanations of a prediction $\varphi(\mathbf{z}) = c$ (where $\mathbf{z} = (\mathbf{x}, \mathbf{y})$) are fair iff for all putative explanations $\mathbf{e} \subseteq \mathbf{z}$ of this prediction, either $\mathbf{e}$ does not entail the predicted class c or $\mathbf{e} \setminus \mathbf{y}$ does entail c. This is logically equivalent to

$$\forall(\mathbf{e} \subseteq \mathbf{z})\ \forall(\mathbf{z}'' \supseteq \mathbf{e} \setminus \mathbf{y})\ \exists(\mathbf{z}' \supseteq \mathbf{e}).\ [(\varphi(\mathbf{z}') \neq c) \vee (\varphi(\mathbf{z}'') = c)]$$

which clearly places UFC in Π_2^P. □

Whether UFC is complete for Π_2^P is an open problem. We conjecture that it is, since there is no obvious polynomial-time verifiable certificate. It is worth noting that, by Proposition 2 the problem of testing whether *all* predictions are universally fair is in co-NP, since the counter-example certificate is simply the values of $\mathbf{x}, \mathbf{y_1}, \mathbf{y_2}$ satisfying criterion (2). This is somewhat counter-intuitive in that testing the universal fairness of all predictions may be easier than testing the universal fairness of one prediction.

3.4 Relation of Fairness to Robustness and Adversarial Examples.

Informally speaking, robustness is the property that two almost identical inputs (i.e. points in feature space $\mathbb{F}$) should be labelled equally. We give a general formal definition for robustness, then we show that fairness can be seen as a particular case. This may enable previous work on robustness to be adapted for fairness. We discuss this particular point by presenting the relationship with adversarial examples.

Let $\mathcal{P}(\mathbb{F})$ be the power set of $\mathbb{F}$. Let f be a neighbourhood function: that is, $f : \mathbb{F} \to \mathcal{P}(\mathbb{F})$. We say that an ML model $\mathbb{M}$ with corresponding function φ is robust w.r.t. a neighbourhood function f if $\forall \mathbf{z} \in \mathbb{F}$, $\forall \mathbf{z}' \in f(\mathbf{z})$, $\varphi(\mathbf{z}) = \varphi(\mathbf{z}')$. Consider the example of adversarial robustness [57]. The neighbourhood function related to adversarial robustness can be defined as $f^{ar}(\mathbf{z}) = \{\mathbf{z}' \mid d(\mathbf{z}, \mathbf{z}') \leq \epsilon\}$ where $d : \mathbb{F} \times \mathbb{F} \to \mathbb{R}^+$ is a distance metric and $\epsilon > 0$. Adversarial robustness can then be defined as the property that $\forall \mathbf{z} \in \mathbb{F}$, $\forall \mathbf{z}' \in f^{ar}(\mathbf{z})$, $\varphi(\mathbf{z}) = \varphi(\mathbf{z}')$.

Fairness can be viewed a particular case of robustness. Note that defining fairness using distance (thus neighbourhood) functions is used in the so-called "fairness through awareness" measure [20, 64]. We can consider the neighbourhood function $f^* : \mathbb{F} \to P(\mathbb{F})$ such that $f^*(\mathbf{x}.\mathbf{y}) = \{\mathbf{x}.\mathbf{y}' \mid \mathbf{y}' \in \mathbb{P}\}$. That is, the neighbourhood of an input $\mathbf{z}$ is the set of inputs that have the same unprotected features. The robustness property using the f^* neighbourhood function is identical to the fairness criterion (1): two inputs that have the same unprotected features should be labelled equally.

Using this observation, we can relate work on robustness and fairness. For example, if one can construct an adversarial example (or counterexample) that uses changes to protected features, then the model is deemed unfair.

4 Learning Fair ML Models

Section 3 showed how to assess whether datasets or models could be checked for a specific fairness criterion. The purpose of this section is to investigate ways of synthesizing ML models when a dataset is unbiased, and when it is biased. Whereas the case of unbiased datasets requires simple changes to existing ML model synthesis approaches, the case of biased datasets requires more substantial changes. As argued earlier, the paper focuses on logic-based ML models, namely decision sets, decision trees and decision lists. Moreover, the dataset is assumed to be consistent, for simplicity. The modifications to the case of inconsistent datasets are also briefly discussed.

4.1 Unbiased Datasets

This section shows how to synthesize ML models from unbiased datasets. Two different settings can be envisioned, namely heuristic and optimal approaches.

Heuristic approaches. Starting from an unbiased consistent dataset, we can use an off-the-shelf ML tool to learn a fair ML model. A simple solution is to discard the protected features, since we know from Proposition 1 that inconsistencies will not be introduced. The resulting ML model will not depend on the protected features, and so the model is fair. There are no restrictions on which ML model to consider.

Optimal approaches. Exact approaches for synthesizing DS's, DT's and DL's have been studied since the 90s, with a peak of recent interest due to the concerns of interpretability and explainability [42]. These approaches offer formal guarantees, for example in terms of model size and accuracy. As with heuristic approaches, a simple solution to synthesize fair ML models is to ensure that protected variables are *not* allowed to be used when constructing the ML model. We first briefly recall a formal method for computing minimal size decision sets, proposed in [41], namely the so-called MinDS$_3$ model, then we show how it can be modified to synthesize fair DS's. Notice that similar procedures could be employed with *any* other logic-based ML models.

The choice of this model is motivated by simplicity, and others could be considered as well. Most methods learn a set of disjunctive normal form (DNF) formulas, one for each class. MinDS$_3$ learns one DNF for one class, and uses the examples from training data for the other class as the DNF for that class [41]. Given K features and M examples, we consider the synthesis of N rules associated with class c_1, where each rule is a term (conjunction) of up to K literals. The variables of the model are the following:

- s_{jr}: whether for rule j, the feature F_r is skipped.
- l_{jr}: the literal on feature F_r for rule j, in the case the feature is *not* skipped.
- d^0_{jr}: whether rule j discriminates feature F_r on value 0 (in the sense that F_r occurs as a positive literal in term j).
- d^1_{jr}: whether rule j discriminates feature F_r on value 1 ($\neg F_r$ occurs in term j).
- cr_{jq}: whether rule j covers $e_q \in \mathcal{T}^+$ (i.e. e_q satisfies term j).

The constraints associated with the SAT encoding are the following:

1. Each term (rule) must have at least one literal:

$$\left(\bigvee_{r=1}^{K} \neg s_{jr} \right) \qquad j \in \{1, \ldots, N\} \tag{6}$$

2. One must be able to account for which literals are discriminated by which rules:

$$\begin{aligned} d^0_{jr} &\leftrightarrow \neg s_{jr} \wedge l_{jr} \qquad && j \in \{1, \ldots, N\} \wedge r \in \{1, \ldots, K\} \\ d^1_{jr} &\leftrightarrow \neg s_{jr} \wedge \neg l_{jr} \qquad && j \in \{1, \ldots, N\} \wedge r \in \{1, \ldots, K\} \end{aligned} \tag{7}$$

3. Each negative example $e_q \in \mathcal{T}^-$ must be discriminated by each of the N rules (e_q satisfies no rule). Recall that $\mathbf{z}_q[r]$ denote the value of feature F_r for e_q. Then, we have:

$$\left(\bigvee_{r=1}^{K} d^{\mathbf{z}_q[r]}_{jr} \right) \qquad j \in \{1, \ldots, N\} \wedge e_q \in \mathcal{T}^- \tag{8}$$

4. Each positive example $e_q \in \mathcal{T}^+$ must be covered by (i.e. satisfy) some rule.
   - First, define whether a rule covers some specific positive example:

$$cr_{jq} \leftrightarrow \left(\bigwedge_{r=1}^{K} \neg d^{\mathbf{z}_q[r]}_{jr} \right) \qquad j \in \{1, \ldots, N\} \wedge e_q \in \mathcal{T}^+ \tag{9}$$

   - Second, each $e_q \in \mathcal{T}^+$ must be covered by some rule.

$$\left(\bigvee_{j=1}^{N} cr_{jq} \right) \qquad e_q \in \mathcal{T}^+ \tag{10}$$

To ensure that the DS respect the FTU rule, we add the constraints (s_{jr}), $j = 1, \ldots, N$, for each $F_r \in \mathcal{P}$, denoting that a literal of feature F_r should not be used in rule j. This way, the synthesized DS will not include literals on the protected features.

4.2 Biased Datasets

If a dataset is biased, then a completely accurate ML model *must* exhibit unfairness.

Proposition 4. For a consistent dataset $\mathcal{T}$, if (4) holds, i.e. the dataset is biased, then *any* ML model that is accurate must exhibit FTU unfairness.

Proof. If (4) holds, then there exist in $\mathcal{T}$ a point $(\mathbf{x}, \mathbf{y}_1)$ with some prediction c_1 and a point $(\mathbf{x}, \mathbf{y}_2)$ with some prediction c_2. Thus, if some ML model is accurate, criterion (2) must be false. □

Proposition 4 indicates that if a dataset is biased then accuracy implies loss of fairness and fairness implies loss of accuracy. This section investigates how logic-based models can be synthesized such that fairness is ensured. Due to Proposition 4, the price to pay is that the model is no longer 100% accurate[7]. Furthermore, this section also illustrates how accuracy can be traded off with the size of the ML model representation.

Maximum accuracy. The first problem we study is: find a DS that is fair and has maximum accuracy. As we show next, there is a simple algorithm for obtaining a DS that maximizes accuracy on training data. For each $\mathbf{u} \in \mathbb{N}$ having $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{P}$ such that $\varphi(\mathbf{u}, \mathbf{v}_1) \neq \varphi(\mathbf{u}, \mathbf{v}_2)$, consider the set of examples, denoted by $Conflict_u$, where the values of the non-protected features are those in tuple $\mathbf{u}$. Consider the *least frequent* class among examples in $Conflict_u$. (For ties, pick one class randomly.) Redefine the training data by removing the examples in $Conflict_u$ that are associated to the least frequent class. Now, use the model proposed in Section 4 to learn a DS, that discards the protected features.

[7] It should be noted that, in ML settings, logic-based models that are not 100% accurate are expected to be less sensitive to overfitting. Thus, the fact that some accuracy is lost is not necessarily a drawback [8].

Proposition 5. The algorithm above yields a fair decision set with maximum accuracy.

Proof. The learned DS is fair by construction. Moreover, the lack of accuracy is solely due to protected features being relevant for constructing an accurate decision set. For each set of common non-protected features, the most frequently-occurring prediction is chosen. Hence, accuracy cannot be improved if the model is to be fair. □

4.3 Non-Accurate Interpretable Models

In practice, 100% accurate models are often unwieldy. So an interesting problem is how to synthesize a fair DS, that respects some accuracy target, while placing some bound on the size of the ML model. The solution we propose is when values $\mathbf{x}$ of the non-protected features have multiple predictions (due to the protected features $\mathbf{y}$) to allow freedom of the prediction to be picked (instead of imposing a majority vote) provided it is a function of only $\mathbf{x}$. This ensures fairness of the learnt model according to (1), but this flexibility can be used for reducing the number of rules (or the number of literals in rules) while ensuring that some target accuracy metric is met.

We now study how to synthesize interpretable models that are not 100% accurate. It should be noted that earlier work often makes a number of assumptions regarding trading off accuracy with representation size [5, 30, 36, 51]; ours makes none.

We consider the encoding proposed in Section 4 for computing a smallest decision set. Recall that to ensure fairness, we add the constraints (s_{jr}), $j = 1, \ldots, N$, for each $F_r \in \mathcal{P}$, so that protected features F_r are not be used in any rule j. A model corresponding to a solution of this SAT instance covers all positive examples and discriminates all negative examples. To adapt this model, i.e. (6) to (10), so that it is not necessarily accurate, one can allow each negative example in the training data not to be discriminated. For $e_q \in \mathcal{T}^-$, let nd_{jq} denote that e_q is not discriminated by rule j, and let nd_q denote that e_q is not discriminated (by any rule). Then, we update the model as follows:

$$\begin{array}{ll} nd_{jq} \leftrightarrow \left(\bigwedge_{r=1}^{K} \neg d_{jr}^{\mathbf{z}_q[r]}\right) & j \in \{1, \ldots, N\} \wedge e_q \in \mathcal{T}^- \\ nd_q \rightarrow \left(\bigvee_{j=1}^{N} nd_{jq}\right) & e_q \in \mathcal{T}^- \end{array} \tag{11}$$

(Equivalence for the second constraint is not needed.) If the target accuracy is $0 \le \tau \le 1$, then the constraint on accuracy becomes:

$$\sum_{e_q \in \mathcal{T}^-} nd_q \le \lfloor \tau \times |\mathcal{T}^-| \rfloor \tag{12}$$

If the model includes DNF's for other classes, then (12) must be extended accordingly, both the sum and the right-hand side (e.g. $\tau \times (|\mathcal{T}^-| + |\mathcal{T}^+|)$).

These proposed changes to the model introduced in Section 4 (and taken from [41]) enables learning a decision set that is no longer 100% accurate, and so one can trade off accuracy for representation size while ensuring fairness.

5 Theoretical study of criteria for dataset bias

This section provides a theoretical justification for considering FTU in Section 3. We consider the axioms which a criterion for deciding dataset bias should satisfy. A criterion for bias is a boolean function f on datasets $\mathcal{T} \in \mathcal{P}(\mathbb{F} \times \mathcal{C})$ such that $f(\mathcal{T}) = 1$ if $\mathcal{T}$ is biased. First, any such criterion should be independent of the coding of features and classes: replacing a boolean feature F_i by $\neg F_i$ or interchanging the positive and negative classes should not alter dataset bias. Similarly for merging or splitting features, provided this concerns just protected features or just unprotected features: for example, whether we use two boolean features or a single feature with values in $\{0, 1, 2, 3\}$ should not have any effect on deciding dataset bias. Formally, *coding-independence* is invariant under bijective renaming of feature-values or class-names and merging of any pair of features which are either both protected or both non-protected.

A criterion f for bias should always return 0 if all data are identical on the protected features. We call this the *lack of arbitrariness* condition (if $\exists \mathbf{y_0}$ s.t. $\forall \langle(\mathbf{x}, \mathbf{y}), c\rangle \in \mathcal{T}$, $\mathbf{y} = \mathbf{y_0}$, then $f(\mathcal{T}) = 0$). For example, we cannot decide that there is racial bias if all data only concern people of the same race.

Another desirable property of a bias criterion is *monotonicity*: a dataset should not become less biased by eliminating unprotected features (if $\mathcal{T}'$ is obtained from $\mathcal{T}$ by discarding some unprotected features, then $f(\mathcal{T}') \geq f(\mathcal{T})$). For example, if salary and age are unprotected features, and race is a protected feature, then keeping the same classification but ignoring age should not make the dataset less biased, since we are ignoring information which could legitimately be used to classify the data.

We choose a purely logical approach to learning as opposed to a statistical approach. In many applications, and for various possible reasons, the sample distribution may not reflect the true distribution on which the learned model is to be used. For example, the sample data could have been specifically selected by a teacher to cover extreme cases rather than being a random sample. In this logical context, we also impose the following *simplicity* condition: bias can be proved by exhibiting just two different decisions (if $f(\mathcal{T}) = 1$, then $\exists \mathcal{T}' \subseteq \mathcal{T}$ containing just 2 examples with $f(\mathcal{T}') = 1$). Simplicity is a restrictive condition which is only justified in the context of explainable AI: other notions of fairness are possible if we relax this simplicity condition [64].

We say that a dataset is class-uniform if it classifies all data into the same class and that it is protected-uniform if all data are identical on the protected features. A criterion f for bias is *discerning* if it categorizes as unbiased at least one dataset which is neither class-uniform nor protected-uniform and categorizes as biased at least one dataset.

It turns out that the FTU (as given by (4)) is the only possible criterion for bias that satisfies all the above conditions.

Proposition 6. The only discerning criterion for data set bias which satisfies the coding-independence, lack of arbitrariness, monotonicity and simplicity conditions is the FTU.

Proof. By coding independence, we can merge all non-protected features in $\mathcal{N}$ and all protected features in $\mathcal{P}$, so that we effectively have only two features (with possibly large domains). Coding-independence means that applying any permutation to values does not change bias. This implies that the only operation we can use on features or classes is equality (or inequality). The simplicity condition implies that we can detect bias from two examples $(\mathbf{x}_1, \mathbf{y}_1)$, $(\mathbf{x}_2, \mathbf{y}_2)$ which belong to different classes. Since the criterion for bias is not arbitrary, we know that it cannot impose $\mathbf{y}_1 = \mathbf{y}_2$. A criterion which was a function only of $\mathbf{x}_1, \mathbf{x}_2$ would violate the monotonicity condition, since eliminating all unprotected features would then leave us with a trivial bias criterion. We therefore have to impose the condition $\mathbf{y}_1 \neq \mathbf{y}_2$ in the test for bias.

Suppose now that the criterion decides bias by testing just $\mathbf{y}_1 \neq \mathbf{y}_2$. Since the criterion is discerning, it must categorize as unbiased some dataset $\mathcal{T}$ which is neither class-uniform nor protected-uniform. Since $\mathcal{T}$ is not protected-uniform, there are data $(\mathbf{u}_1, \mathbf{v}_1)$, $(\mathbf{u}_2, \mathbf{v}_2)$ in $\mathcal{T}$ such that $\mathbf{v}_1 \neq \mathbf{v}_2$. Since $\mathcal{T}$ is categorized as unbiased, both $(\mathbf{u}_1, \mathbf{v}_1)$, $(\mathbf{u}_2, \mathbf{v}_2)$ must belong to the same class in $\mathcal{T}$. Now, since $\mathcal{T}$ is not class-uniform, there is $(\mathbf{u}_3, \mathbf{v}_3)$ which belongs to another class in $\mathcal{T}$. But, since $\mathbf{v}_3$ cannot be equal both to $\mathbf{v}_1$ and $\mathbf{v}_2$, the criterion for bias decides that $\mathcal{T}$ is biased, which is a contradiction.

We therefore have to impose $\mathbf{y}_1 \neq \mathbf{y}_2$ together with a condition on $\mathbf{x}_1, \mathbf{x}_2$ in the criterion for bias. If we also impose $\mathbf{x}_1 \neq \mathbf{x}_2$, then this would not satisfy monotonicity (since eliminating all unprotected features could render the dataset unbiased). The only remaining case is to impose the condition $\mathbf{x}_1 = \mathbf{x}_2$ together with $\mathbf{y}_1 \neq \mathbf{y}_2$. This criterion for bias corresponds exactly to the FTU (as given by (4)). □

Another desirable property of a dataset bias criterion is that bias is invariant under the addition of irrelevant (i.e. not used by the model) unprotected features, such as shoe-size when deciding to grant a loan. This appears to be a reasonable condition. However, the following proposition shows that it is impossible to satisfy this *irrelevant-features* condition together with all conditions stated above.

Proposition 7. There is no discerning criterion for dataset bias which satisfies the coding-independence, lack of arbitrariness, monotonicity, simplicity and irrelevant-features conditions.

Proof. By Propositin 6 FTU is the only possible candidate criterion for bias satisfying all these conditions. Consider a dataset which is categorized as biased by the FTU. However, by adding sufficient irrelevant features we can ensure that all data are distinct on the unprotected features and hence the FTU would consider the extended dataset to be unbiased, thus contradicting invariance under addition of irrelevant features. □

For example, when a bank decides whether to grant a loan to one of its clients it has information stored such as their bank account number. If this number is considered as an unprotected feature, any model will satisfy the FTU criterion (we are assuming there are no two clients with the same account number). The automatic detection of irrelevant features is an interesting problem for future research.

6 Preliminary Experimental Results

This section assesses empirically the proposed ideas on a selection of well-known datasets. The aim of the experiments is to (1) illustrate that datasets can be practically checked for bias using Algorithm 1, (2) show that ML models can be tested for fairness using condition (2), and (3) make an attempt to synthesize fair decision sets [41,49] as discussed in Section 4.

Experimental Setup. For the experiments, several Python scripts were implemented, instrumenting SAT and SMT oracle calls. Whenever needed, MiniSat 2.2 [21] was used as a SAT oracle while Z3 [52] was employed as an SMT solver. The solvers were accessed through the well-known Python APIs, namely PySAT [38] and PySMT [29].

The experiments were performed on a Macbook Pro with an Intel Core i7 2.6GHz CPU and 16GB of memory and focused on a few publicly available datasets studied in the context of algorithmic fairness. These include *Compas*, *Adult*, *German*, and *Ricci*. The datasets are binarized using the standard one-hot encoding method [58]. Their sizes is detailed in Figure 2a. *Compas* is a popular dataset known [6] for exhibiting racial bias of the COMPAS algorithm used for scoring a criminal defendant's likelihood of reoffending; the dataset includes a few protected features, namely, race-related parameters *African American*, *Asian*, *Hispanic*, *Native American*, and *Other* but also *Female*.

Adult [47] is originally taken from the Census bureau and targets predicting whether or not a given adult person earns more than $50K a year depending on various features, among which the protected ones are *Race* and *Sex*. *German* credit data (e.g. see [24]), given a list of people's features, classifies them as good or bad credit risks; the protected features are *Sex* and *Age*. The *Ricci* dataset [26] comes from the case of Ricci vs. DeStefano [63], "a case before the U.S. Supreme Court in which the question at issue was an exam given to determine if firefighters would receive a promotion"; the protected feature is *Race*.

6.1 Assessing Dataset Bias

The first part of the experimental assessment aims at checking whether the aforementioned datasets exhibit bias with respect to the corresponding protected features. Note that although the *Compas* and *German* datasets are inconsistent, they can still be tested for bias (see Section 3). Running Algorithm 1 reports that (1) *Compas* and *Adult* are biased with respect to all the protected features while (2) *German* and *Ricci* do not exhibit bias with respect to any protected feature. Point (1) indicates that there is *no way* to train an ML model of maximum accuracy whilst being fair with respect to the protected features. In particular, this confirms the famous bias-related issues of the *Compas* algorithm. Moreover, the results for the *Ricci* dataset should be highlighted. Since this dataset is unbiased, one is guaranteed that a fair ML model can be synthesized. (This should be contrasted with the unfair heuristic models studied in earlier work [26].)

6.2 Assessing Model Fairness

Here we focus on testing fairness of boosted tree models trained with the XGBoost algorithm [13] for the considered datasets. To perform an exhaustive experiment assessing accuracy of the target models and to be able to draw conclusions, we followed the standard paradigm of 10-fold cross-validation. As such, each dataset is randomly divided into 10 equally sized chunks of examples and the experiment is done 10 times (one per chunk), each time dealing with 90% target dataset, i.e. with 1 of the 10 chunks discarded. This way every dataset is tested for fairness of the respective model wrt. each protected feature 10 times. Overall, this results in 60 fairness tests for *Compas*, 20 for *Adult*, 20 for *German*, and 10 for *Ricci* – the total number of tests to perform is 110. (Recall that each test is made as an SMT oracle call dealing with formula (2).)

Each XGBoost model trained for this experiment contains 50 trees per class with each tree having depth 3 (this suffices to get a reasonable classification accuracy). Boosted trees are encoded into SMT by applying a simple encoding proposed in [40].

The minimum, maximum, and average running time per test for each of the datasets is shown in Figure 2b. Observe that testing fairness is not computationally expensive and can be done for medium-sized boosted trees. Also note that fairness tests are on average more time consuming for the *German* dataset.

Regarding the fairness of the trained ML models, the tests reported that only 2 (out of 10) models trained for the *Compas* dataset are fair wrt. protected feature *Other*. All the other models trained for all datasets are *unfair* with respect to *every* protected feature. This should not come as a surprise given that these models were trained with no knowledge about the protected features, which is usually the case in practice. Since the fairness check (2) is *model agnostic*, this result confirms the power and applicability of the proposed ideas in practical situations when fairness of ML models is a concern.

	Adult	**Compas**	**German**	**Ricci**
orig. features	12	11	21	5
binary features	65	46	1073	235
examples	14113	6172	1000	118

(a) Size of the considered datasets.

	Adult	**Compas**	**German**	**Ricci**
min. (s)	0.31	0.24	2.50	0.33
avg. (s)	12.22	0.56	87.08	0.36
max. (s)	63.77	3.37	1062.32	0.43

(b) Running Time for Checking Model Fairness.

Fig. 2: Size of the Datasets (left) & Time for Assessing (XGBoost) Model Fairness (right)

6.3 Synthesizing Fair Decision Sets

This section aims at synthesizing fair decision sets for the case of biased dataset *Compas* and unbiased dataset *Ricci*, based on the approach of [41] (results for *Adult* and *German* are not reported here because synthesis of DS models for these datasets seems too challenging). While for the former dataset (*Compas*) one can drop the protected features from the dataset and then trade off accuracy with DS size, for the unbiased *Ricci* dataset it suffices to modify the DS model, as described in Section 4.

Furthermore, since *Ricci* is unbiased wrt. the protected feature *Race*, we can achieve 100% accuracy with the resulting DS model (similarly with decision trees/lists). In fact, a perfectly accurate decision set for *Ricci* computed by the modified MinDS_3 model [41] has only two rules, i.e. one rule per class. The downside here is that each rule is quite long, s.t. the decision set has 68 literals in total. This is in clear contrast to the interpretability purpose of decision sets. However, by trading off accuracy with the DS size, one can get the following non-overlapping MinDS_1 model [41] with 97.5% accuracy and having only 4 rules and 7 literals in total, which is easy to interpret:

IF $\mathsf{Position} \neq \mathsf{Leutenant} \land \mathsf{Oral} \leq 63.75$ THEN $\mathsf{Class} = 0$
IF $\mathsf{Combine} \leq 69.372$ THEN $\mathsf{Class} = 0$

IF $\mathsf{Position} = \mathsf{Leutenant} \land \mathsf{Combine} > 69.372$ THEN $\mathsf{Class} = 1$
IF $\mathsf{Oral} > 63.75 \land \mathsf{Combine} > 69.372$ THEN $\mathsf{Class} = 1$

Since *Compas* is biased wrt. the protected features, training a fair ML model for this dataset can be done by sacrificing the model's accuracy. Concretely, the *maximum feasible* accuracy for this dataset is 69.73%. Although DS models achieving this accuracy for *Compas* can be trained, they are too large to interpret (each has at least a few hundred literals). Thus, one may want to sacrifice accuracy further and get a more interpretable (i.e. smaller) DS model instead. For instance, the following non-overlapping MinDS_1 model has 66.32% accuracy and it is fair with respect to all the protected features:

IF $\mathsf{Number\_of\_Priors} > 17.5 \land \neg \mathsf{score\_factor}$ THEN $\mathsf{Two\_yr\_Recidivism}$
IF $\mathsf{Number\_of\_Priors} > 17.5 \land \mathsf{Age\_Above\_FourtyFive} \land \mathsf{Misdemeanor}$ THEN $\mathsf{Two\_yr\_Recidivism}$

IF $\mathsf{Number\_of\_Priors} \leq 17.5$ THEN $\neg \mathsf{Two\_yr\_Recidivism}$
IF $\mathsf{score\_factor} \land \neg \mathsf{Age\_Above\_FourtyFive}$ THEN $\neg \mathsf{Two\_yr\_Recidivism}$
IF $\mathsf{score\_factor} \land \neg \mathsf{Misdemeanor}$ THEN $\neg \mathsf{Two\_yr\_Recidivism}$

7 Conclusions and Research Directions

We studied the fairness of ML models, by considering the criterion FTU [31,48], but proposing instead a semantic definition. We also proposed theoretical justifications for the use of FTU. Moreover, we developed criteria for assessing fairness in ML models and bias in datasets, and related fairness with explanations and robustness. Finally, we investigated approaches for synthesizing fair ML models. Future work will address limitations of the current work, namely assessing non-protected features exhibiting discriminatory information and/or taking statistical measures into account [12,67].

References

1. Adebayo, J.A.: FairML: ToolBox for diagnosing bias in predictive modeling. Master's thesis, Massachusetts Institute of Technology (2016)
2. Adebayo, J.A.: FairML: auditing black-box predictive models (2017)
3. Aïvodji, U., Arai, H., Fortineau, O., Gambs, S., Hara, S., Tapp, A.: Fairwashing: the risk of rationalization. In: ICML. pp. 161–170 (2019)
4. Aïvodji, U., Ferry, J., Gambs, S., Huguet, M., Siala, M.: Learning fair rule lists. CoRR **abs/1909.03977** (2019), http://arxiv.org/abs/1909.03977
5. Angelino, E., Larus-Stone, N., Alabi, D., Seltzer, M., Rudin, C.: Learning certifiably optimal rule lists for categorical data. J. Mach. Learn. Res. **18**, 234:1–234:78 (2017)
6. Angwin, J., Larson, J., Mattu, S., Kirchner, L.: Machine bias. propublica.org, http://tiny.cc/a3b3iz (May 2016)
7. Berk, R., Heidari, H., Jabbari, S., Kearns, M., Roth, A.: Fairness in criminal justice risk assessments: The state of the art. Sociological Methods & Research (03 2017). https://doi.org/10.1177/0049124118782533
8. Berkman, N.C., Sandholm, T.W.: What should be minimized in a decision tree: A re-examination. Departement of Computer Science (1995)
9. Bessiere, C., Hebrard, E., O'Sullivan, B.: Minimising decision tree size as combinatorial optimisation. In: CP. pp. 173–187 (2009)
10. Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185. IOS Press (2009)
11. Bird, S., Hutchinson, B., Kenthapadi, K., Kiciman, E., Mitchell, M.: Fairness-aware machine learning: Practical challenges and lessons learned. In: KDD. pp. 3205–3206 (2019)
12. Cardelli, L., Kwiatkowska, M., Laurenti, L., Paoletti, N., Patane, A., Wicker, M.: Statistical guarantees for the robustness of bayesian neural networks. In: Kraus, S. (ed.) Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019. pp. 5693–5700. ijcai.org (2019), https://doi.org/10.24963/ijcai.2019/789
13. Chen, T., Guestrin, C.: XGBoost: A scalable tree boosting system. In: KDD. pp. 785–794 (2016)
14. Chouldechova, A.: Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. Big Data **5**(2), 153–163 (2017)
15. Chouldechova, A., Roth, A.: A snapshot of the frontiers of fairness in machine learning. Commun. ACM **63**(5), 82–89 (2020)
16. Corbett-Davies, S., Pierson, E., Feller, A., Goel, S., Huq, A.: Algorithmic decision making and the cost of fairness. In: KDD. p. 797–806. KDD '17 (2017)
17. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd Edition. MIT Press (2009), http://mitpress.mit.edu/books/introduction-algorithms
18. Demsar, J., Curk, T., Erjavec, A., Gorup, C., Hocevar, T., Milutinovic, M., Mozina, M., Polajnar, M., Toplak, M., Staric, A., Stajdohar, M., Umek, L., Zagar, L., Zbontar, J., Zitnik, M., Zupan, B.: Orange: data mining toolbox in python. J. Mach. Learn. Res. **14**(1), 2349–2353 (2013)
19. Dressel, J., Farid, H.: The accuracy, fairness, and limits of predicting recidivism. Science advances **4**(1), eaao5580 (2018)
20. Dwork, C., Hardt, M., Pitassi, T., Reingold, O., Zemel, R.: Fairness through awareness. In: ITCS. p. 214–226 (2012)
21. Eén, N., Sörensson, N.: An extensible SAT-solver. In: SAT. pp. 502–518 (2003)
22. Ehlers, R.: Formal verification of piece-wise linear feed-forward neural networks. In: ATVA. pp. 269–286 (2017)

23. European Union High-Level Expert Group on Artificial Intelligence: Ethics guidelines for trustworthy AI. https://ec.europa.eu/digital-single-market/en/news/ethics-guidelines-trustworthy-ai (April 2019)
24. Feldman, M., Friedler, S.A., Moeller, J., Scheidegger, C., Venkatasubramanian, S.: Certifying and removing disparate impact. In: KDD. pp. 259–268. ACM (2015)
25. Friedler, S.A., Scheidegger, C., Venkatasubramanian, S.: On the (im)possibility of fairness. CoRR **abs/1609.07236** (2016), http://arxiv.org/abs/1609.07236
26. Friedler, S.A., Scheidegger, C., Venkatasubramanian, S., Choudhary, S., Hamilton, E.P., Roth, D.: A comparative study of fairness-enhancing interventions in machine learning. In: FAT. pp. 329–338 (2019)
27. Galhotra, S., Brun, Y., Meliou, A.: Fairness testing: testing software for discrimination. In: FSE. pp. 498–510 (2017)
28. Garg, S., Perot, V., Limtiaco, N., Taly, A., Chi, E.H., Beutel, A.: Counterfactual fairness in text classification through robustness. In: AIES. pp. 219–226 (2019)
29. Gario, M., Micheli, A.: PySMT: a solver-agnostic library for fast prototyping of SMT-based algorithms. In: SMT Workshop (2015)
30. Ghosh, B., Meel, K.S.: IMLI: an incremental framework for MaxSAT-based learning of interpretable classification rules. In: AIES. pp. 203–210 (2019)
31. Grgic-Hlaca, N., Zafar, M.B., Gummadi, K.P., Weller, A.: The case for process fairness in learning: Feature selection for fair decision making. In: NIPS Symposium on Machine Learning and the Law (2016)
32. Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques, 3rd edition. Morgan Kaufmann (2012)
33. Hardt, M., Price, E., Srebro, N.: Equality of opportunity in supervised learning. In: Lee, D.D., Sugiyama, M., von Luxburg, U., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29. pp. 3315–3323 (2016), http://papers.nips.cc/paper/6374-equality-of-opportunity-in-supervised-learning
34. Holstein, K., Vaughan, J.W., III, H.D., Dudík, M., Wallach, H.M.: Improving fairness in machine learning systems: What do industry practitioners need? In: CHI. p. 600 (2019)
35. Hu, H., Siala, M., Hebrard, E., Huguet, M.J.: Learning optimal decision trees with maxsat and its integration in adaboost. In: IJCAI. pp. 1170–1176 (2020)
36. Hu, X., Rudin, C., Seltzer, M.: Optimal sparse decision trees. In: NeurIPS. pp. 7265–7273 (2019)
37. Huang, X., Kwiatkowska, M., Wang, S., Wu, M.: Safety verification of deep neural networks. In: CAV. pp. 3–29 (2017)
38. Ignatiev, A., Morgado, A., Marques-Silva, J.: PySAT: A Python toolkit for prototyping with SAT oracles. In: SAT. pp. 428–437 (2018)
39. Ignatiev, A., Narodytska, N., Marques-Silva, J.: Abduction-based explanations for machine learning models. In: AAAI. pp. 1511–1519 (2019)
40. Ignatiev, A., Narodytska, N., Marques-Silva, J.: On validating, repairing and refining heuristic ML explanations. CoRR **abs/1907.02509** (2019), http://arxiv.org/abs/1907.02509
41. Ignatiev, A., Pereira, F., Narodytska, N., Marques-Silva, J.: A SAT-based approach to learn explainable decision sets. In: IJCAR. pp. 627–645 (2018)
42. Kamath, A.P., Karmarkar, N., Ramakrishnan, K.G., Resende, M.G.C.: A continuous approach to inductive inference. Math. Program. **57**, 215–238 (1992)
43. Katz, G., Barrett, C.W., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: An efficient SMT solver for verifying deep neural networks. In: CAV. pp. 97–117 (2017)
44. Katz, G., Huang, D.A., Ibeling, D., Julian, K., Lazarus, C., Lim, R., Shah, P., Thakoor, S., Wu, H., Zeljic, A., Dill, D.L., Kochenderfer, M.J., Barrett, C.W.: The marabou framework for verification and analysis of deep neural networks. In: CAV. pp. 443–452 (2019)

45. Kilbertus, N., Rojas-Carulla, M., Parascandolo, G., Hardt, M., Janzing, D., Schölkopf, B.: Avoiding discrimination through causal reasoning. In: NeurIPS. p. 656–666 (2017)
46. Kleinberg, J.M., Mullainathan, S., Raghavan, M.: Inherent trade-offs in the fair determination of risk scores. In: 8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA. pp. 43:1–43:23 (2017)
47. Kohavi, R.: Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In: KDD. pp. 202–207 (1996)
48. Kusner, M.J., Loftus, J.R., Russell, C., Silva, R.: Counterfactual fairness. In: NeurIPS. pp. 4066–4076 (2017)
49. Lakkaraju, H., Bach, S.H., Leskovec, J.: Interpretable decision sets: A joint framework for description and prediction. In: KDD. pp. 1675–1684 (2016)
50. Leofante, F., Narodytska, N., Pulina, L., Tacchella, A.: Automated verification of neural networks: Advances, challenges and perspectives. CoRR **abs/1805.09938** (2018), http://arxiv.org/abs/1805.09938
51. Malioutov, D., Meel, K.S.: MLIC: A MaxSAT-based framework for learning interpretable classification rules. In: CP. pp. 312–327 (2018)
52. de Moura, L.M., Bjørner, N.: Z3: an efficient SMT solver. In: TACAS. pp. 337–340 (2008)
53. Nabi, R., Shpitser, I.: Fair inference on outcomes. In: AAAI. pp. 1931–1940 (2018)
54. Narayanan, A.: Translation tutorial: 21 fairness definitions and their politics. In: FAT (2018)
55. Narodytska, N.: Formal analysis of deep binarized neural networks. In: IJCAI. pp. 5692–5696 (2018)
56. Narodytska, N., Ignatiev, A., Pereira, F., Marques-Silva, J.: Learning optimal decision trees with SAT. In: IJCAI. pp. 1362–1368 (2018)
57. Narodytska, N., Kasiviswanathan, S.P., Ryzhyk, L., Sagiv, M., Walsh, T.: Verifying properties of binarized deep neural networks. In: AAAI. pp. 6615–6624 (2018)
58. Pedregosa, F., et al.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
59. du Pin Calmon, F., Wei, D., Vinzamuri, B., Ramamurthy, K.N., Varshney, K.R.: Optimized pre-processing for discrimination prevention. In: NeurIPS. pp. 3992–4001 (2017)
60. Pulina, L., Tacchella, A.: An abstraction-refinement approach to verification of artificial neural networks. In: CAV. pp. 243–257 (2010)
61. Ruan, W., Huang, X., Kwiatkowska, M.: Reachability analysis of deep neural networks with provable guarantees. In: IJCAI. pp. 2651–2659 (2018)
62. Shih, A., Choi, A., Darwiche, A.: A symbolic approach to explaining bayesian network classifiers. In: IJCAI. pp. 5103–5111 (2018)
63. Supreme Court of the United States: Ricci v. DeStefano. U.S. 557, 174 (2009)
64. Verma, S., Rubin, J.: Fairness definitions explained. In: FairWare@ICSE. pp. 1–7 (2018)
65. Verwer, S., Zhang, Y.: Learning decision trees with flexible constraints and objectives using integer optimization. In: CPAIOR. pp. 94–103 (2017)
66. Verwer, S., Zhang, Y.: Learning optimal classification trees using a binary linear program formulation. In: AAAI. pp. 1625–1632 (2019)
67. Wu, M., Wicker, M., Ruan, W., Huang, X., Kwiatkowska, M.: A game-based approximate verification of deep neural networks with provable guarantees. Theor. Comput. Sci. **807**, 298–329 (2020), https://doi.org/10.1016/j.tcs.2019.05.046