Fair Decision Making via Automated Repair of Decision Trees

Anonymous Author(s)

ABSTRACT

Data-driven decision-making allows more resource allocation tasks, such as approval of loans, to be done by programs. Unfortunately, real-life training datasets may capture human biases, and the learned models can be unfair. To resolve this, one could either train a new, fair model from scratch or to repair an existing unfair model. The former approach is liable for unbounded semantic difference, hence is unsuitable for social or legislative decisions. Meanwhile, the scalability of state-of-the-art model repair techniques is unsatisfactory.

In this paper, we aim to automatically repair unfair decision models by converting any decision tree or random forest into a fair one with respect to a specific dataset and sensitive attributes. We built the *FairRepair* tool, inspired by automated program repair techniques for traditional programs. It uses an MaxSMT solver to decide which paths in the decision tree could be flipped or refined, with both fairness and semantic difference as hard constrains. Our approach is sound and complete and the output repair always satisfies the desired fairness and semantic difference requirements.

FairRepair is able to repair an unfair decision tree on the well-known COMPAS dataset [2] in 1 minute on average, achieving 90.3% fairness and only 2.3% semantic difference. We compared *FairRepair* with 4 state-of-the-art fairness learning algorithms [21, 26, 44, 46]. While achieving a similar fairness by training new models, they incur 8.9% to 13.5% semantic difference. The results show that *FairRepair* is capable of repairing an unfair model while maintaining the accuracy and incurring small semantic difference.

1 INTRODUCTION

The introduction of data-driven decision-making has automated resource allocation tasks that have been previously performed by humans. Programs in this domain are built from training datasets. For example, for a loan approval program, the training dataset contains information about past loan applications and approval decisions. Past decisions, however, could be influenced by various sources of bias (including human biases), and the decision making programs obtained from such data could therefore be unfair. Testing, analysis, and verification of decision-making programs in relation to fairness properties have been studied in recent years. However, the repair of these programs was less studied. Most previous works [21, 26, 44–46] on fairness learning algorithms aimed to produce fair classifiers from *datasets*, rather than *repairing* an already existing unfair model. Training new models from scratch may lead to undesired semantic difference (i.e., proportion of inputs that receive different predictions in the two models). This is crucial in social or legislative domains. For example, the U.S. courts has used the COMPAS (Correctional Offender Management Profiling for Alternative Sanctions) algorithm to predict the defendants' underlying recidivism risks. The predictions were biased against African-American defendants [24]. Such predictions were used in accessing pre-trial release and sentencing, and altering these decisions at a large scale would cause undesired social consequences.

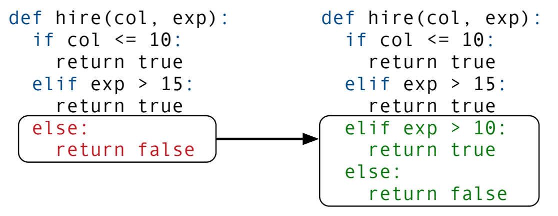
ICSE 2022, May 21–29, 2022, Pittsburgh, PA, USA
2018. ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

In addition, certain legal guidelines require fairness to be hard bounded. The *80% rule* (or more generally, the p-rule) states that if the selection rate for a certain group is less than 80% (or p) of the group with the highest selection rate, it is deemed a discrimination against that group [9, 44]. State-of-the-art fair learning approaches usually do not hard constrain fairness and semantic difference [21, 44, 46], making them unsuitable for repairing models like COMPAS. In addition, state-of-the-art fair decision tree learning approaches can only encode simple fairness properties on binary sensitive attributes [25], hence these techniques cannot account for the p-rule. Moreover, as decision tree learning is essentially an optimisation problem, scalability is a major challenge for large trees.

To the best of our knowledge, the only existing fairness-guided repair approach is DIGITS [6], which relies on probability distributions to generate inputs and to achieve probabilistic guarantees of completeness. In practice, input distributions are often unavailable and hard to approximate. Instead, our approach uses a possibly unfair dataset of past human decisions. While DIGITS provides convergence guarantee, its scalability is unsatisfactory and is less suitable for large scale decision tree repair tasks. We further compare the two approaches in Section 8.

In this paper, we study automatic repair of decision-making programs. We present our *FairRepair* approach to derive fair programs while maintaining high accuracy and a small semantic difference, without using sensitive attributes in the repair procedure. The output from *FairRepair* can be used to help the users understand implicit or unconscious biases in manual decisions, and to explain how to improve organisational decision-making workflows. Our work is in line with existing work on automated program repair (APR), except that we target fairness specifically. We focus on repairing decision trees and random forests, since they are commonly used to capture decision making in software systems in various fields, including industrial operations research. We focus on *group fairness*, modified from [19] and [6]. Our new notion requires the ratios of the selection rates of the minority groups and the majority groups to be bounded from both below and above, instead of from below only. This modification is to accommodate the cases where the minority groups are unknown.

FairRepair uses the dataset to repair the unfair model and produces a fair model with bounded semantic difference as output. It works by transforming the decision trees and random forests into their logical equivalents, encoding the fairness and semantic difference criteria as hard constraints, and using an MaxSMT solver to produce repairs. Our approach then refines the decision tree paths and changes their leaf labels as needed. In Section 6 we prove that our approach is sound and complete. That is, for any fairness and semantic difference constraints, whenever the constraints are satisfiable, *FairRepair* outputs a repaired model as a solution. In contrast, fairness learning algorithms usually have no guarantee on semantic difference. DIGITS does prove soundness and completeness over



```
def hire(col, exp):
  if col <= 10:
    return true
  elif exp > 15:
    return true
  else:
    return false
```

```
def hire(col, exp):
  if col <= 10:
    return true
  elif exp > 15:
    return true
  elif exp > 10:
    return true
  else:
    return false
```

(a) Hiring example classifier **(b) Repaired hiring classifier**

Gender	Col	Exp	Gender	Col	Exp
F	7	8	M	6	13
F	11	6	M	14	28
F	25	2	M	28	21
F	33	20	M	38	9
F	49	13	M	43	2

(c) Sample dataset used for repair

Figure 1: Hiring example classifier before and after repair.

its search space, but the search space itself is limited. A repair may exist, but if it is outside the search space then DIGITS would be unable to find it. This renders DIGITS essentially incomplete.

We evaluate *FairRepair* on three publicly available datasets: Adult dataset from UC Irvine, UFRGS entrance exam and GPA dataset from Harvard, and COMPAS dataset from ProPublica. Our implementation achieves the claimed fairness guarantees for different fairness thresholds, semantic difference bounds and sensitive attributes. We also study the scalability of *FairRepair*, and find that it is able to repair decision trees with 10k leaves on a dataset with 48k data points and 14 features in under 1 minute (average time), while achieving 90.2% fairness and only 5.7% semantic difference. In contrast, DIGITS [6] was able to repair small decision trees (with fewer than 100 leaves) with three features in 10 minutes and achieve 85% fairness and 9.8% semantic difference. We also evaluated four state-of-the-art fair learning algorithms [21, 26, 44, 46]. Although these algorithms were able to construct models on the Adult dataset with similar fairness, we found that the resulting models had a significant semantic difference (up to 11.7%). We make our *FairRepair* tool available online as open-source [47].

In summary, our contributions are:

1. We propose a novel MaxSMT-based solution to fairness-guided automated repair of decision trees and random forests.

2. We prove that our approach is sound and complete in finding fair repairs with a bounded semantic difference.

3. We implement our approach in the tool *FairRepair*. We show that *FairRepair* outperforms state-of-the-art fairness learning algorithms on both fairness and semantic difference.

2 FAIRREPAIR OVERVIEW

Consider a simple decision tree T_{job} in Figure 1(a). This classifier predicts if a candidate should be offered a job based on two features: college ranking (`col`) and working experience in years (`exp`). A simple dataset D in Figure 1(c) is used for illustration. It is important that T_{job} does not discriminate against minorities. In this example, we show how our techniques can be applied with respect to *demographic parity*, also known as *group fairness*. Group fairness is based on the legal guideline for avoiding hiring discrimination, the “80% rule”. This rule says that the rate at which minority candidates are offered jobs should be at least 80% of the rate at which majority candidates are offered jobs. We extend this notion to bound the ratios of selection rates from both below and above. Let *gender* be the sensitive attribute, group fairness is represented by

$$Y_{\text{job}} \equiv \frac{r_{\text{m}}}{r_{\text{f}}} \geq 0.8 \wedge \frac{r_{\text{f}}}{r_{\text{m}}} \geq 0.8 \equiv 0.8 \leq \frac{r_{\text{f}}}{r_{\text{m}}} \leq 1.25,$$

where $r_{\text{f}} = \mathbb{P}(\text{get jobs}|\text{female})$ and $r_{\text{m}} = \mathbb{P}(\text{get jobs}|\text{male})$. With D, we can compute (by counting) that $r_{\text{m}} = 0.6$ and $r_{\text{f}} = 0.4$, i.e., $Y_{\text{job}} = $ `FALSE` and the model T_{job} is not group fair.

To repair unfair decision trees and random forests, FairRepair performs two actions: *flip* and *refine*. The flip action changes the label of a decision tree path, and the refine action splits a path into sub-paths and assign new labels to them. Each decision tree path corresponds to a unique hyper-rectangular region in the input space. We use the term *path hyper-rectangles (PH)* to denote decision tree paths. T_{job} can thus be interpreted as three PHs with different labels. Below, the numbers in parenthesis represent the probabilities of male and female applicants in each PH. By definition, all these probabilities add up to 1.

- PH1: $(col \leq 10)$, `TRUE`, $(m : 0.1, f : 0.1)$
- PH2: $(col > 10) \times (exp > 15)$, `TRUE`, $(m : 0.2, f : 0.1)$
- PH3: $(col > 10) \times (exp \leq 15)$, `FALSE`, $(m : 0.2, f : 0.3)$

For example, the probability of an input being in PH1 is 0.2, with both male and female contribute 0.1 probability. When a population model is unavailable, the probabilities are approximated by enumerating a dataset. The acceptance rate is 60% for male candidate, but 40% for female candidates, hence this model does not satisfy the 80% rule. We now proceed to explain the two actions.

The Flip Action. One way to repair the model to meet Y_{job} is to flip PH2, i.e., change its label to `FALSE`. This will change r_{f} and r_{m} to both 0.2, thus satisfying Y_{job}. However, such repairs introduce new inequalities and applicants who could have been offered a job earlier are now rejected. For large scale government policies, such changes could cause serious social consequences. Hence, it is important to bound the semantic difference, i.e., the proportion of applicants that receive different outcomes before and after the repair.

Our approach is SMT-based, which encodes the semantic difference as a logical constraint. Theoretically, *FairRepair* is able to find the repair with minimal semantic difference, by encoding the constraints as minimising the semantic difference. But optimisation tasks induce higher complexity. As a practical compromise, we bound the semantic difference. In this case, let the bound be 0.1, i.e., no more than 10% of the population should be given different outcomes after the repair. Now assign each PH$_i$ a pseudo-Boolean SMT variable X_i, representing their label. If we add up the probabilities of paths receiving `TRUE` labels, Y_{job} can be represented as (after simplification):

$$0.8 \leq \frac{r_{\text{m}}}{r_{\text{f}}} = \frac{0.1X_1 + 0.2X_2 + 0.2X_3}{0.1X_1 + 0.1X_2 + 0.3X_3} \leq 1.25.$$

For example, if the classification outcome of PH2 X_2 = FALSE, as a pseudo-Boolean variable, its numeric value will be 0 and its corresponding terms will vanish, i.e., $0.3 = 0.2 + 0.1$ proportion of the input population will get different classification outcomes. The total semantic difference requirement can thus be represented by:

$$(0.1+0.1)\text{XOR}(X_1,1)+(0.2+0.1)\,\text{XOR}(X_2,1) \\ +(0.2+0.3)\,\text{XOR}(X_3,0) \le 0.1$$

The two SMT formulas above (fairness requirement and semantic different requirement) are UNSAT, hence no solutions can be found by only flipping the PHs. To continue, we need to refine the PHs.

The Refine Action. The next step of *FairRepair* is to split the PHs into smaller sub-PHs and repeat SMT solving (full description in Section 4.4). In brief, each time we select an attribute that is most correlated with the sensitive attribute and split the PH into sub-PHs on this attribute. The disjoint union of the sub-PHs is the original PH. For now, assume that we obtained a refinement of PH3 into

∘ PH4: $(col > 10) \times (exp \le 10)$, FALSE, $(0.2, 0.2)$.
∘ PH5: $(\text{col} > 10) \times (10 < \text{exp} \le 15)$, FALSE, $(0, 0.1)$.

Assigning new pseudo-Boolean variables X_4 and X_5 to PH4 and PH5, the fairness requirement is now (simplified):

$$0.8 \le \frac{r_{\text{m}}}{r_{\text{f}}} = \frac{0.1X_1 + 0.2X_2 + 0.2X_4 + 0X_5}{0.1X_1 + 0.1X_2 + 0.2X_4 + 0.1X_5} \le 1.25.$$

The semantic difference requirement becomes:

$$(0.1+0.1)\text{XOR}(X_1,1) + (0.2+0.1)\,\text{XOR}(X_2,1) + \\ (0.2+0.2)\,\text{XOR}(X_4,0) + (0+0.1)\,\text{XOR}(X_5,0) \le 0.1$$

The new SMT formulas are SAT, and one SAT assignment is X_1 = TRUE, X_2 = TRUE, X_4 = FALSE and X_5 = TRUE. By splitting PH3 into PH4 and PH5, and flipping PH5, we produced a repaired decision tree that satisfies both the fairness and the semantic difference criteria (Figure 1(b)). For real life dataset and models, the SMT formulas are more complex and require calibrated encoding. We also include additional soft constraints on accuracy and syntactic change. *FairRepair* keeps refining the PHs until the SMT result is SAT. Sections 4 and 6 detail the *FairRepair* algorithm and its correctness.

3 PROBLEM FORMULATION

3.1 Preliminaries

A decision making program P is defined on an input space S, such that $\forall x \in S.\, P(x) \in \{\text{TRUE}, \text{FALSE}\}$. Let S be n-dimensional. Let D be a dataset. Each data point in D contains an n-tuple $(a_1, a_2, \ldots, a_n) \in S$ and a Boolean outcome b. An input distribution p is a probability density function defined on the input space. If S is not associated with an input distribution, we assume that D represents the input distribution perfectly. The frequencies of n-tuples in D are interpreted to be the density function p.

Some attributes are *sensitive*, e.g., gender and race. Sensitive groups S_i's are subsets of the input space, partitioned by combinations of different values of sensitive attributes, e.g., (female, Asian). Let M denote the total number of sensitive groups. For program P, passing rate r_i of a sensitive group S_i is the probability of an input in S_i receiving a pre-defined outcome (TRUE, if not specified). That is, $r_i := \mathbb{P}(P(x) = \text{B} | x \in S_i)$, where B = TRUE.

Definition 3.1 (p-rule Score). *The p-rule score measures the ratios of the passing rates as the degree of unfairness.*

$$p\text{-rule score} := \min_{\forall 1 \le i,j \le M} \frac{r_i}{r_j} \tag{1}$$

Definition 3.2 (Group Fairness). *For an input space S and a decision making program P, if the following holds,*

$$\forall 1 \le i, j \le M, cr_i \le r_j \le \frac{r_i}{c} \tag{2}$$

or equivalently p-rule score $\ge c$, *then we say that P attains group fairness. The factor c is called the* fairness threshold.

Definition 3.3 (Semantic Difference). *(Equiv. semantic distance.) For an input space S and two decision making programs P_1, P_2 defined on S, their semantic difference $SD(P_1, P_2, S)$ (or $SD(P_1, P_2)$ when S is clear) is the proportion of the population that receives different outcomes in the two programs. Formally, $SD(P_1, P_2, S) := \mathbb{P}(P_1(x) \neq P_2(x) | x \in S)$.*

3.2 Fairness Repair Problem

Given an input space S and a decision making program P defined on S and not satisfying group fairness, a *repair* is a decision making program R defined on S that attains group fairness. An *optimal repair* R_{op} is a repair that minimises the semantic difference between itself and P. By definition, R_{op} might not be unique.

Definition 3.4 (α-optimal Repair). *Let R_{op} be an optimal repair. An* α-optimal repair *is a repair R with a bounded semantic difference compared to* P, *i.e., if there exists an optimal repair R_{op}, R should be bounded by a multiplicative factor $\alpha > 1$ compared to that of the optimal repair. Formally, $SD(R, \text{P}) \le \alpha \cdot SD(R_{op}, \text{P})$.*

To solve a *fairness repair problem* is to find an α-optimal repair R. A repair can be any program that satisfies the pre-defined fairness requirement. To minimise the semantic difference is to ensure a minimal proportion of inputs are affected (i.e., receiving different labels before and after the repair). Since many decision making problems on fairness are relevant to social policies or resource distribution, it is important to ensure that the change is small, i.e., fewest people are affected due to the new policy. To avoid disparate treatment, we refrain from using sensitive attributes in the repaired model, but allow using them in the algorithm. In addition, we want to ensure that the original program is not substantially impacted. Minimising the semantic difference between the repaired program and the original one also helps to minimise the changes in the accuracy and precision due to the repair process.

In principle, the algorithm can be forced to find an optimal solution R_{op} as we continue decreasing α, but it is too computationally expensive for realistic models, because finding R_{op} is a complex optimisation problem. Therefore, we chose a more practical approach with a fixed α, and as shown in Section 7, it does provide satisfactory semantic distance without making the problem intractable.

4 FAIRREPAIR FOR DECISION TRESS

In this section, we explain how *FairRepair* repairs a decision tree. The output decision tree will satisfy the fairness requirement, and the semantic difference between the output and the original model, as compared to that of the optimal repair, will be bounded by a

Algorithm 1 Top-level algorithm in FairRepair for Decision Trees

```
Input: D = dataset, T = decision tree, c = fairness threshold.
Output: Solution for Repaired Tree.
  [H] ← CollectPH(T)
  [P] ← PathProbCal(D, H) {Path probabilities.}
  [R] ← LowerBoundCal([P]) {Desired passing rates.}
  [HC], [SC] ← Hard Constraints, Soft Constraints
  while isRefinable([H]) == TRUE do
    if MaxSMT([H], [P], [R], [HC], [SC]) == UNSAT then
      Refine([H]) {Refine path hyper-rectangles.}
    else
      return MaxSMT([H], [P], [R], [HC], [SC])
  while TRUE do
    [HC] ← RelaxSemDiffConstraint([HC])
    if MaxSMT([H], [P], [R], [HC], [SC]) == SAT then
      return MaxSMT([H], [P], [R], [HC], [SC])
```

multiplicative coefficient α. Note that our approach is able to output a solution for any $\alpha > 1$ (see proof in Section 6). We assume that the dataset used for repair correctly represents the input distribution.

Algorithm 1 outlines the repair procedure. We first collect all decision tree path hyper-rectangles (PHs). Each PH represents a region in the input space that is specified by its path constraints. Next we calculate the path probabilities, which are the proportion of inputs that reside in each PH. We use TRUE as the desired outcome when computing the passing rates, which are part of the fairness inequalities as described in Section 3. To meet the fairness criteria, the passing rates need to be altered. We compute the theoretical lower bound of the changes in passing rates, i.e., the minimal proportion of inputs that will be affected (receive different outcome after repair) in each sensitive group. The fairness and semantic difference criteria are encoded as hard MaxSMT constraints, while the accuracy and syntactic change criteria are encoded as soft MaxSMT constraints. They are all sent to a MaxSMT solver. For some PHs, we flip their labels. For some others, we refine (by inserting additional conditions) and assign them different labels to meet the desired passing rates. The MaxSMT procedure terminates when a solution is found. After all PHs have been fully refined, i.e., no PH can be split further, we relax the semantic difference constraint. Note that this relaxation does not pose a problem (we explain why in Section 6). Finally, we modify and output the decision tree based on the solution from the MaxSMT solver. Next, we detail each of the above steps.

4.1 Computing Path Probabilities

Given a dataset D with an underlying input space S and a decision tree T defined on S, the algorithm starts by collecting the tree paths. Note that T may not be trained from D. The paths are labelled π_i, $1 \leq i \leq n$, where n is the total number of paths in T. The path hyper-rectangles are labelled H_i's, and there is a one-to-one correspondence between H_i's and π_i's. Note that each $H_i \subseteq S$ is a subspace of S. We abuse the notation to let H_i also denote $D \cap H_i$, the set of data points in D that reside in H_i, when there is no ambiguity. Let Y_i denote the label of H_i. Let $\{A_1, \ldots, A_m\}$ be the set of sensitive attributes. There are in total $M := |A_1| \times |A_2| \times \cdots \times |A_m|$ sensitive groups, where each $|A_i|$ is the number of partitions of the valuations of A_i. Let $S(p)$ denote the sensitive group of a data point p. For any p, $S(p) \in [1, M]$. For each path hyper-rectangle H_i, count the number of data points inside (denoted $|H_i|$) and use them to compute the *path probabilities*, $p_i := |H_i|/|D|$. In addition, we record $p_{i,j} := |p \in H_i : S(p) = j|/|D|$. Note that $p_i = \sum_{1 \leq j \leq M} p_{i,j}$, for all $1 \leq i \leq n$. Define $P_j := \sum_{1 \leq i \leq n} p_{i,j}$ as the probability of sensitive group j. By definition, $\sum_{1 \leq j \leq M} P_j = 1$. In our previous example, there is only one sensitive attribute and $M = 2$. There are three PHs, i.e., $n = 3$. $P_{\text{male}} = P_{\text{female}} = 0.5$ are the probabilities of the two sensitive groups.

4.2 Calculating Lower Bounds for Changes in the Passing Rates

With the path probabilities, we can now compute the passing rates for each sensitive groups. In particular, for sensitive group S_j, its passing rate $r_j = \sum_{1 \leq i \leq n, Y_i = \text{TRUE}} p_{i,j}/P_j$. In our previous example, $r_f = 0.4$ and $r_m = 0.52$ for female and male groups. Recall the fairness requirement: $\forall 1 \leq i, j \leq M, \frac{r_i}{r_j} > c$, where r_i and r_j are passing rates of subtrees. To meet this requirement, we need to modify the decision tree such that for each $1 \leq i \leq M$, the passing rate of T_i changes from r_i to x_i, where each x_i is a real variable to be computed in the following linear optimisation problem. We aim to find a solution with the minimal semantic difference from the original, unfair decision tree. To find the x_i's, we solve:

$$\begin{cases} \text{minimise } \sum_{i=1}^{M} p_i \cdot |x_i - r_i|, \\ \forall\, 1 \leq i, j \leq M, \dfrac{x_i}{x_j} \geq c, \\ \forall\, 1 \leq i \leq M, 0 \leq x_i \leq 1. \end{cases}$$

The second line is the fairness requirement. Since passing rates are probabilities, the third line requires them to be bounded by 0 and 1. The first line accounts for the semantic difference. Each $p_i \cdot |x_i - r_i|$ is a lower bound for the proportion of data points being affected in set S_i, and not necessarily conforms to R_{op}, the optimal repair. It is possible that the passing rates x_i and r_i corresponds to different groups of data points, and the actual number of data points being affected is not $|x_i - r_i|$, but $x_i + r_i$. In our example, a hypothetical PH6 ($col > 40$) has equal path probability as PH1 does, but it has no overlap with PH1. Thus, $p_i \cdot |x_i - r_i|$ is only a lower bound on the proportion of data points affected in S_i.

The optimisation problem always has a solution, giving us the "desired passing rates" $x_1, x_2, \ldots, x_M$ which are used to modify the path hyper-rectangles in the decision tree, as discussed in the following.

4.3 Calculating patches with MaxSMT Solver

We now convert the PHs into their logical equivalents and encode the constraints. Let I_i be a pseudo-Boolean value that represents the *current* label of H_i, and X_i be a pseudo-Boolean variable that represents the *desired* label of H_i. If $X_i == I_i$, the PH is unchanged; otherwise it is *flipped*. Below are the constraints in MaxSMT solving.

Hard Constraints. The fairness requirements are the first set of hard constraints.

$$\forall 1 \le j, k \le M, c \le \frac{\sum_{1\le i\le n} p_{i,j} \cdot X_i / p_j}{\sum_{1\le i\le n} p_{i,k} \cdot X_i / p_k} \ge 1/c,$$

where $0 \le c \le 1$ is the fairness threshold. The numerator represents the passing rate r_j for sensitive group S_j, while the denominator represents r_k. We also account for semantic difference in the hard constraints.

$$\sum_{1\le i\le n} p_i \cdot \mathrm{Xor}(X_i, I_i) \le \alpha \sum_{i=1}^{M} p_i \cdot |x_i - r_i|.$$

The left hand side is the semantic difference between our modified tree and the original tree. The right hand side is the theoretical lower bound of semantic difference, multiplied by α.

Soft Constraints. The soft constraints requires X_i to be same as I_i. The solver should satisfy the maximal number of these soft constraints, which means that we should flip the return value of as few path hyper-rectangles as possible.

$$\forall 1 \le i \le n, X_i = I_i.$$

Note that the above set of formulas are not guaranteed to have a solution, because the path probabilities might be too large. If the solver returns UNSAT for the maxSMT problem, then we cannot make the decision tree fair by only flipping PHs.

4.4 Refining Path Hyper-rectangles

If the fairness criteria cannot be satisfied by flipping the outcome of the PHs (i.e., the hard constraints are unsatisfiable), we proceed to refine the PHs. In particular, each time we split one PH into two based on a single attribute. There is no restriction whether this attribute should be discrete or continuous, but we forbid the use of sensitive attributes. When the solver outputs UNSAT, we refine one PH and re-run the solver. Since the total number of PHs is bounded by the size of the dataset, this procedure always terminates after a finite number of steps.

Refinement requires choosing a PH and a constraint on an attribute. As a PH may contain inputs from various sensitive groups, flipping one PH can impact the passing rates of multiple sensitive groups. In general, we choose PHs that are most imbalanced in terms of the proportions of members from each sensitive group, and choose attributes that are most correlated with the sensitive ones. This allows us to separate the sensitive groups to the greatest extent and provides more room for manipulating the labels. When the sensitive attributes are unavailable, the inputs from each group can only be distinguished using attributes that correlate to the sensitive ones. We choose the most correlated ones by default as an optimisation. Choosing other correlated attributes would make our approach slower because of the greater number of refinements. In practice, this would not matter significantly as refinement steps are often not needed in practice (2/3 of the repair tasks in our experiments are completed without refinement). We refer the reader to Section 7 for more details.

For each path hyper-rectangle, we rank all attributes based on their correlation with the sensitive attribute. This is done by Pearson's product-moment coefficient:

$$\rho_{X,Y} = \frac{\mathbb{E}[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}.$$

We compute $\rho_{S,A}$ for the sensitive attribute S and each attribute A. The computation is based on the data points in H_i's. It is possible that different PHs have different rankings of correlated attributes. Let A_i denote the most correlated attribute for H_i, and the attribute values of A_i are denoted as $a_{i,k}$'s, where $1 \le k \le |A_i|$.We assume all numerical attributes are divided into intervals so that they can be treated as categorical attributes, i.e., all $|A_i|$'s are finite. Note that it is possible that $A_i = A_j$ for $i \ne j$. Each H_i is now divided into $|A_i|$ disjoint sub-PHs, denoted as $H_{i,k}$'s. During the refinements, we recompute $p_{i,j,k}$ by counting the data points in each sensitive group in $H_{i,k}$. The notions $I_{i,k}$ and $X_{i,k}$ follow the same change to denote the current label and the desired label of $H_{i,k}$.

The MaxSMT formulas are as follows:

Hard constraints. The group fairness constraints are now (with c being the fairness threshold):

$$\forall 1 \le j, l \le M, c \le \frac{\sum_{1\le i\le n} \sum_{1\le k\le |A_i|} p_{i,j,k} \cdot X_{i,k} / p_j}{\sum_{1\le i\le n} \sum_{1\le k\le |A_i|} p_{i,l,k} \cdot X_{i,k} / p_l} \ge 1/c.$$

Still, the numerator and the denominator represent the passing rates for S_j and S_l respectively. The semantic difference constraints are now:

$$\sum_{1\le i\le n} \sum_{1\le k\le |A_i|} p_{i,k} \cdot \mathrm{Xor}(X_{i,k}, I_{i,k}) \le \alpha \sum_{i=1}^{M} p_i \cdot |x_i - r_i|,$$

with the right hand side unchanged.

Soft constraints. The soft constraints still require each $X_{i,k}$ to be the same as $I_{i,k}$. The solver will find a solution that maximally satisfy the following constraints:

$$\forall 1 \le i \le n, 1 \le k \le |A_i|, X_{i,k} = I_{i,k}.$$

4.5 Relaxing the Semantic Distance Constraint

When all the PHs are fully refined, i.e., no PH can be split into subsets any further without using sensitive attributes, the refinement step stops. At this point we adopt an alternative method. We relax the semantic distance constraint gradually, until a repair that meets the fairness requirement is found. The MaxSMT variables and constraints remain the same as in Section 4.3, except for the semantic difference constraints below.

$$\sum_{1\le i\le n} \sum_{1\le k\le |A_i|} p_{i,k} \cdot \mathrm{Xor}(X_{i,k}, I_{i,k}) \le \alpha \cdot \mathrm{SemDiff}.$$

Initially, SemDiff, the semantic difference, is set to $\sum_{i=1}^{M} p_i \cdot |x_i - r_i|$, the theoretical lower bound computed in Section 4.2. If the solver returns UNSAT, we relax the semantic difference constraint by multiplying SemDiff by α and re-run the solver. We iteratively update SemDiff until the solver finds a solution of $X_{i,k}$'s. To see that our algorithm always terminates, note that there exists a trivial solution for the fairness constraints, i.e., all $X_{i,k}$'s are TRUE. The left hand side of the above inequality represents the actual semantic change in fraction, so it is bounded by 1. Meanwhile, since $\alpha > 1$, as we repeatedly relax SemDiff, the right hand side of the inequality is unbounded and will exceed 1 after finitely many iterations. Thus the solver is guaranteed to find a solution after a finite number of steps. This concludes our algorithm.

5 FAIRREPAIR FOR RANDOM FORESTS

The algorithm in Section 4 can also be extended to repair random forests. Here we only provide a sketch of the extension instead of an algorithm in full detail. A random forest is an ensemble of decision trees. For a given input, its outcome is the majority vote (for classification) or average (for regression) of the outcomes from the decision trees. In our case, the outcome is binary, so we use the majority vote definition. Let $F := \{T_1, T_2, \dots, T_n\}$ be a random forest with n decision trees. For input x,

$$F(x) = \underset{1\le i\le n}{\text{Maj}}\ T_i(x),\ \text{where}\ \underset{1\le i\le n}{\text{Maj}}\ X_i = \begin{cases} 1, \text{if } \sum X_i > \dfrac{n}{2}, \\ 0, \text{otherwise.} \end{cases}$$

Let D and S be as defined in Section 4, and F be a random forest defined on S. We extract the decision trees $T_1, \dots, T_n$ from F, and collect the PHs as in Section 4.1. Let $\{A_1, \dots, A_m\}$ and M be defined as earlier. Each PH is given by a tuple (i, j, hid), indicating the tree T_i it belongs to, the sensitive group S_j it resides in, and a unique index "hid" to identify the PH in T_i. We next calculate the passing rates r_i's for each sensitive group. Since the optimal passing rates are purely determined by the classification results, we have the same linear optimisation problem for a random forest as for a decision tree. Solving the linear optimisation problems, we obtain the theoretical optimal passing rates x_i's.

The outcome for an input $d \in D$ is the majority vote of the outcomes of PHs where point d resides, across all decision trees. The point d resides in the intersection of these PHs. If we compute the intersections of all PHs over all decision trees (for a random forest with 30 decision trees each with 6,000 paths in our case) the total number of intersections quickly explodes. We overcome this by only considering the intersections containing at least one data point from the dataset D. This follows from our assumption that the dataset D represents the input distribution. For each data point $d \in D$, we identify the PH where d resides in each decision tree. For decision tree T_i, let the index pair of the PH that contains d be $(i, \text{hid}_{i,d})$. The intersection I_d is

$$I_d := \bigcap_{1\le i\le n} \mathrm{P}_{i,hid_{i,d}},$$

where $\mathrm{P}_{i,hid_{i,d}}$ is the PH in T_i with $hid_{i,d}$ being the unique index of the PH. For all $d \in D$, we compute I_d and record $(i, \text{hid}_{i,d})$ for each $1 \le i \le n$. It is possible that $I_{d_1} = I_{d_2}$ for $d_1 \ne d_2$. We shall group data points into equivalence classes based on the intersection they reside in. We say $d_1 \sim d_2$ if $I_{d_1} = I_{d_2}$, and denote d_1 as the equivalence class without loss of generality. Let $p_d = |x \in I_d|/|D|$ and $p_{d,j} = |x \in I_d : S(x) = j|/|D|$. We use e to denote the equivalence classes in D (e_d as the representative of e) and use $\mathcal{D}$ to denote the set of equivalence classes in D.

Next, we construct the SMT formulas. For each PH $\mathrm{P}_{i,hid}$ in the random forest F, assign an SMT pseudo-Boolean variable $X_{i,hid}$ to denote its label. The fairness hard constraints are as follows:

$$\forall\, 1 \le k, l \le M,\ \frac{\sum\limits_{e\in\mathcal{D}} \underset{1\le i\le n}{\text{Maj}}\ (X_{i,hid_{i,e}}) \cdot p_{d,k}/p_k}{\sum\limits_{e\in\mathcal{D}} \underset{1\le i\le n}{\text{Maj}}\ (X_{i,hid_{i,e}}) \cdot p_{d,l}/p_l} \ge c.$$

The semantic hard constraint is:

$$\sum_{e\in\mathcal{D}} p_{e_d} \cdot \text{Xor}(\underset{1\le i\le n}{\text{Maj}}\ (X_{i,hid_{i,e_d}}), F(e_d)) \le \alpha \sum_{i=1}^{M} p_i \cdot |x_i - r_i|.$$

The soft constraints are very much similar to those in Section 4 hence omitted.

If the result of the above SMT constraints is UNSAT, we treat each intersection PH separately. Assign for each I_d a pseudo-Boolean variable X_d. That is, we treat each I_d as an independent PH, and solve for their labels. This reduces to case of repairing a decision tree, hence we omit the constraints. Once we obtain the solution for X_d's, we update the changes in the random forest. If $I_d = X_d$, we do not change anything. If $I_d \ne X_d$, the majority vote in the intersection is changed and we compute the minimum number of PHs whose outcomes need to be flipped to alter the majority vote. Recall the forest F has n trees. Let $I_{i,hid_{i,d}}$ denote the outcomes of the PHs $\mathrm{P}_{i,hid_{i,d}}$ for $1 \le i \le n, d \in D$. Then $\underset{1\le i\le M}{\text{Maj}}\ (I_{i,hid_{i,d}}) = X_d$. To flip X_d, randomly select minimal number of decision trees T_i with $I_{i,hid_{i,d}} \ne X_d$ and add the PH I_d with prediction X_d to T_i, such that $\underset{1\le i\le M}{\text{Maj}}\ (I_{i,hid_{i,d}})$ is flipped. If $I_d = \texttt{TRUE}$, minimally $\sum\limits_{1\le i\le M} (I_{i,hid_{i,d}}) - \left\lfloor \dfrac{n}{2} \right\rfloor$ decision trees with $I_{i,hid_{i,d}} = \texttt{TRUE}$ will have added a new PH I_d (the intersection PH) with a `False` prediction. If $I_d = \texttt{FALSE}$, minimally $\left\lceil \dfrac{n}{2} \right\rceil - \sum\limits_{1\le i\le M} (I_{i,hid_{i,d}})$ decision trees with $I_{i,hid_{i,d}} = \texttt{FALSE}$ will have added a new PH I_d with a `TRUE` prediction.

6 GUARANTEES

We now present a proof of soundness and completeness of our repair technique. This completeness property distinguishes our work from existing decision tree repair tool DIGITS [6]. The only assumption we make is the soundness and completeness of the SMT solver. The proof for decision trees and random forests are similar. Here we omit the proof for random forests.

Given a sound SMT solver, if *FairRepair* outputs a solution, it must satisfy the hard constraints of the fairness requirement and the (possibly relaxed) semantic change constraint. We now prove that whenever our algorithm produces a repaired model, its fairness and semantic difference is always guaranteed.

Theorem 6.1. *For a given decision tree T and a dataset D, for any $\alpha > 1$, if the group fairness requirement is satisfiable and there exists an optimal repair R_{op}, our algorithm (Section 4) is able to find a repair R_{subop} satisfying the fairness requirement, such that $SD(R_{subop}, T) \le \alpha SD(R_{op}, T)$. Here R_{subop} refers to a solution (assignment) of the SMT variables, and SD is the semantic distance function defined in Section 3. For any assignment R of $X_{i,j}$'s,*

$$SD(R, T) = \sum_{i=1}^{M} \sum_{j=1}^{|S_j|} p_{i,j} \cdot \text{Xor}(X_{i,j}, I_{i,j}),$$

where the symbols are defined in Section 4.

Proof. In Section 4.5, we have shown that our algorithm always terminates with an output repair whose fairness is hard-constrained by the SMT solver. It suffices to show that repair satisfies the semantic requirement. A solution could be produced before or after the relaxation of α. If the algorithm terminates before relaxation, the hard constraint ensures that:

$$\sum_{1\le i\le n}\sum_{1\le k\le|A_i|} p_{i,k}\cdot \mathrm{Xor}(X_{i,k}, I_{i,k}) \le \alpha\sum_{i=1}^{M} p_i\cdot|x_i - r_i|.$$

Since the right hand side is the lower bound of semantic difference, $SD(R_{op}, T) \ge \sum_{i=1}^{M} p_i\cdot|x_i - r_i|$. Therefore $SD(R_{subop}, T) \le \alpha SD(R_{op}, T)$. Now it only lefts to show that the same holds after relaxation. To distinguish R_{subop} from R_{op}, let $R_{op} = \{x_{i,j} : 1 \le i \le n, 1 \le j \le M\}$, and $R_{subop} = \{y_{i,j} : 1 \le i \le n, 1 \le j \le M\}$ such that both are assignments of the variables $X_{i,j}$'s. After the PHs have been fully refined, if the SMT solver still returns UNSAT, it means that the theoretical lower bound of semantic difference cannot be achieved. Here we use the completeness of the solver. Thus, we know that

$$\sum_{1\le i\le n}\sum_{1\le k\le|A_i|} p_{i,k}\cdot \mathrm{Xor}(X_{i,k}, I_{i,k}) > \alpha\cdot \mathrm{SemDiff},$$

for all assignments for $X_{i,j}$'s that satisfy the fairness requirement. In particular,

$$SD(R_{op}, T) = \sum_{1\le i\le n}\sum_{1\le k\le|A_i|} p_{i,k}\cdot \mathrm{Xor}(x_{i,k}, I_{i,k}) > \alpha\cdot \mathrm{SemDiff}.$$

As explained in Section 4.5, the relaxing procedure always terminates. Let $\mathrm{SemDiff}_0$ be the SemDiff when the solver return UNSAT at the last time. Then we have the following:

$$SD(R_{op}, T) = \sum_{1\le i\le n}\sum_{1\le k\le|A_i|} p_{i,k}\cdot \mathrm{Xor}(x_{i,k}, I_{i,k}) > \alpha\cdot \mathrm{SemDiff}_0$$

$$SD(R_{subop}, T) = \sum_{1\le i\le n}\sum_{1\le k\le|A_i|} p_{i,k}\cdot \mathrm{Xor}(y_{i,k}, I_{i,k}) \le \alpha\cdot\alpha\cdot \mathrm{SemDiff}_0.$$

The above inequalities imply

$$SD(R_{subop}, T) \le \alpha\cdot SD(R_{op}, T).$$

□

7 EVALUATION

We evaluated the performance of our algorithm empirically with respect to fairness, semantic difference and accuracy. We conduct the experiments on three real-world datasets, and we compare the results with state-of-the-art algorithms.

7.1 Methodology

For our experiments, we used three popular datasets often used in fairness evaluation. All the datasets involve the attribute *gender* or *sex*, which may be interpreted with different meanings. To avoid ambiguity, we use the term *gender* for all datasets.

- The Adult dataset [1] from the UC Irvine contains 14 demographic attributes over 48,842 individuals. The class labels state whether their income is higher than \$50,000. We choose *gender* (p-rule score = 0.36) and *race* (p-rule score = 0.43) as the sensitive attributes.
- The UFRGS [3] dataset contains 10 attributes over of 43,302 individuals. The attributes are the students' gender and their entrance exam scores (9 subjects). We choose *gender* (p-rule score = 0.61) as the sensitive attribute. The predictions are their GPAs in the college, categorised into two classes, < 3.0 and ≥ 3.0.
- The COMPAS [2] dataset from ProPublica contains 13 attributes over previously convicted criminals. The class labels the defendants' recidivism states within two years. We follow [21] and [41], choose *race* as the senstive attribute, and consider only Caucasian and non-Caucasian defendants (p-rule score = 0.78), in total 6,172 individuals.

We developed *FairRepair* in 3,000 lines of Python code. *FairRepair* repairs models trained using scikit-learn [36] *DecisionTreeClassifier* and *RandomForestClassifier*. In general, our algorithm is not restricted to any specific decision tree or random forest training tool. *FairRepair* changes the structures of the models, but it does not change the output range of classifications or assume new input features. All experiments were run with Python 3.8.10, on a Linux Ubuntu 20.04 server with 28-core 2.0 GHz Intel(R) Xeon(R) CPU and 62GB RAM. For all three datasets, we split them into 40/40/20 divisions as the training subset, repairing subset and the testing subset in the decision tree repair process, so to test the generalisation accuracy of *FairRepair*. The first 80% is used for conducting the repair procedure, while the last 20% for assessing the accuracy and fairness of the repaired model. When generating a *DecisionTreeClassifier* or a *RandomForestClassifier*, we used an 80/20 training-test split of the repairing dataset. For the Adult dataset, 2 choices of sensitive attribute(s) were tested, while for the other two dataset, only one sensitive attribute was tested. For experiments on random forests, we selected the default forest size to be 30 trees[1]. Each experiment was run with 5 different random seeds used by scikit-learn's training procedure, and we recorded the average results.

7.2 Fairness and Accuracy

We evaluate the fairness and accuracy of the repaired models produced by *FairRepair* in the following aspects: 1) how they compare with the original models; and 2) how they compare with state-of-the-art fairness learning algorithms.

Changes in accuracy. While achieving the fairness requirement, the changes in classification accuracy between the original and the repaired models are limited. We set as baseline fairness threshold $c = 0.8$ and semantic bound $\alpha = 1.2$. We further tighten the requirements by increasing c to 0.95 and decreasing α to 1.05. We note that for all three datasets, the improvements in semantic difference led by $\alpha < 1.05$ are all less than 0.1%, hence we do not require semantic bounds to be less than 1.05. We record the accuracy, precision, recall and F1-scores in Table 1 at $\alpha = 1.05$ for decision trees. The random forest results (changes in accuracy before and after repair) are similar hence omitted.

The changes in classification accuracy due to repairs are highly dependent on the input distribution and initial degree of unfairness presence in the model. We shall use models obtained from the adult dataset for explanation. For sensitive attribute *gender*, the accuracy

[1]We measured the accuracy for random forests of different sizes. The differences in initial classification error rate was less than 1% for forests with more than 30 trees.

Dataset	**Adult**	**COMPAS**	**UFRGS**
Sensitive attribute	Gender / Race	Race	Gender
Accuracy (before)	81.0 / 81.0	61.8	69.4
Accuracy (c = 0.8)	76.8 / 80.7	61.4	66.8
Accuracy (c = 0.95)	76.1 / 80.1	59.6	71.2
Precision (before)	61.8 / 61.8	57.3	31.9
Precision (c = 0.8)	52.8 / 60.6	57.0	29.4
Precision (c = 0.95)	51.3 / 59.9	55.5	30.8
Recall (before)	62.9 / 62.9	57.9	33.6
Recall (c = 0.8)	63.6 / 63.4	58.2	35.6
Recall (c = 0.95)	64.2 / 63.7	59.6	24.3
F1-score (before)	62.4 / 62.4	57.7	32.7
F1-score (c = 0.8)	57.6 / 62.0	57.6	32.2
F1-score (c = 0.95)	57.0 / 61.7	57.4	27.2

Table 1: Accuracy, precision, recall and F1-score (in %) of decision trees for the Adult, COMPAS and UFRGS datasets.

Adult	**Accuracy**	**Fairness**	**Sem. Diff.**
FairRepair (tree)	76.2%	90.2%	5.7%
FairRepair (forest)	80.9%	90.1%	5.6%
FAGTB [21]	85.0%	90.8%	10.9%
Kamishima [26]	82.6%	90.1%	11.7%
Zafar [44, 45]	82.3%	88.5%	9.5%
Zhang [46]	83.1%	89.7%	10.0%
COMPAS	**Accuracy**	**Fairness**	**Sem. Diff.**
FairRepair (tree)	59.9%	90.3%	2.3%
FairRepair (forest)	64.2%	90.6%	3.7%
FAGTB [21]	64.6%	90.1%	8.9%
Kamishima [26]	64.0%	90.0%	12.1%
Zafar [44, 45]	63.9%	89.5%	13.5%
Zhang [46]	64.1%	90.0%	9.7%

Table 2: Comparisons between *FairRepair* and other fairness learning algorithms on accuracy, fairness and semantic difference.

at $c = 0.95$ is 76.1%. The initial passing rates computed from the predicted results are (on average) 31.0% for males and 11.3% for females, which is only one-third of males. The initial model has a high accuracy, hence the unfairness in the dataset is highly preserved in the model and the "correct" predictions can themselves be unfair. To balance the passing rates, the algorithm has to explicitly make wrong predictions, which decreases accuracy. In particular, to increase the passing rate for females (and to decrease the passing rate for males), some female applicants (data points) who were initially classified with a `FALSE` prediction have to be given `TRUE` prediction now (and some male applicants who were initially classified with `TRUE` prediction are now given `FALSE` prediction). The female group constitutes 31% of the population, and the male group constitutes 69%. To achieve 0.95 fairness threshold, the semantic difference is at least 6.0%, hence causes the loss in accuracy. For the same reason, the false positive rate and false negative rate changes, causing fluctuations in the precision and recall values.

FairRepair is able to repair the decision trees and random forests, bounding the ratio of the passing rates within a fairness threshold of $c = 0.95$. In real life, such a high fairness threshold is usually unnecessary. For example, the 80% rule requires only $c = 0.8$. We note that at $c = 0.8$, the changes in accuracy and F1-scores for all three datasets are less than 5%. In particular, the accuracy for the repaired models trained on the UFRGS dataset increased at $c = 0.95$. We believe this is a reasonable trade-off for a more fair model. We believe such results do show that *FairRepair* is able to repair unfair models, while their accuracy.

Comparison with state-of-the-art. A good repair should preserve the accuracy and be semantically close to the original model. To concretely evaluate our repair methodology, we selected four state-of-the-art fairness learning algorithms (not restricted to decision tree learners) [21, 26, 44, 46]. These works train new classifiers from scratch, instead of repairing existing unfair models. In particular, they are unable to provide a guaranteed bound on the semantic difference between the original and the new program. We follow the methodology in [21], and compared *FairRepair* with these algorithms on the accuracy, the fairness and the semantic difference (as compared with the original model) achieved by the new/repaired model.

Table 2 lists our experimental results for different algorithms on the well-known Adult dataset and COMPAS dataset, both with gender as the sensitive attribute. We did not include the UFRGS dataset, as it is not in the benchmarks of the open-source implementations of the tools[2][3]. For all these tools, we compute the average of 10 trials with random division of the dataset into 80/20 training/testing subsets. We compared the outcome models of the four tools with an initial model used by *FairRepair*, and computed the semantic difference. We recorded the results of *FairRepair* for both decision tree and random forest repair. For decision trees, the initial models have accuracy of 81.5% and 62.3% for the Adult and the COMPAS datasets respectively. Accordingly, the numbers are 85.1% and 65.6% for random forests. We set fairness threshold $c = 0.9$ and semantic bound $\alpha = 1.2$ for *FairRepair*. For the other tools, we also try to achieve accuracy near 90% by leveraging the parameters, so that we can compare the accuracy and semantic difference more directly. For the COMPAS dataset, all five tools have comparable accuracy. For the Adult dataset, we do note that the accuracy of the repaired model was limited by that of the original one. The models used by the other tools have better prediction capability than simple decision trees and outperformed *FairRepair* even before repair. *FairRepair* improved the p-rule score from 0.78 to 0.90 in exchange of 4.2% accuracy decrease, while maintaining the smallest semantic difference among all 5 tools. The results show that our tool is able to produce a fair repair with respect to a dataset while minimally affect the accuracy and semantic difference.

7.3 Scalability

FairRepair uses an input dataset to compute the fairness and semantic differences as it executes. For large datasets, these operations may be expensive. To investigate scalability we only use the Adult

[2]https://github.com/algofairness/fairness-comparison
[3]https://github.com/Trusted-AI/AIF360

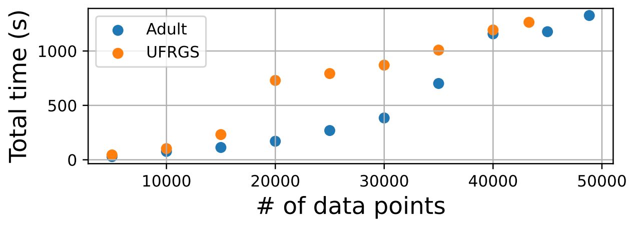


Figure 2: The total running time of *FairRepair* for random forests on the Adult and the UFRGS datasets with increasing number of data points.

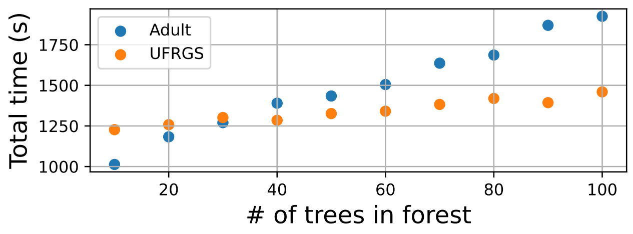


Figure 3: The total running time of *FairRepair* for random forests on the Adult and the UFRGS dataset with increasing forest size.

dataset and the UFRGS dataset, since the COMPAS dataset is relatively small. Figure 2 plots the total running time of *FairRepair* repairing random forests using subsets of the two datasets with varying sizes. We created partitions of the dataset by sampling uniform random subsets of increasing size (Adult: 5K to 45K points, UFRGS: 5K to 40K points). Here we do not show the plots for decision trees, as for both datasets (and their different subsets), the average repair time is less than 1 minute. We found that the choice of fairness threshold c and semantic bound α made little difference to the running time. There is no clear relationship between c and α, and the time needed to repair a model. These experiments used *gender* as the sensitive attribute.

The figure illustrates that larger datasets do take longer to repair. Since fairness and α does not significantly affect the running time, we fixed fairness threshold to be 0.8 and α to be 1.2. As the size of the dataset increases, it takes up to 20 minutes (on average) to repair a random forest on the complete UFRGS dataset, and 25 minutes (on average) for the complete Adult dataset. We also measured the scalability against various forest size, from 10 trees to 100 trees, with fixed fairness threshold 0.8 and $\alpha = 1.2$. We observe a mostly proportional increase in the amount of repair time as the forest size increases. For random forest on the Adult (resp. UFRGS) dataset with 100 trees, *FairRepair* takes up to 35 minutes (resp. 25 minutes) to find a repair. We also note that our code is in Python and we believe it could be further optimized to achieve much better scalability.

8 RELATED WORK

Fairness and its Testing / Analysis. There has been an increased research effort on fairness in machine learning in recent years. Various fairness definitions have been proposed [10, 22, 27, 29]. In our paper, we used group fairness, which is used in the fairness testing work of [19]. Other recent works on fairness testing includes black-box fairness testing [4] and individual fairness testing [40]. There have also been attempts to verify fairness properties, including calculating integration with measure theoretical results [7], and using concentration inequalities to bound the error [8]. Many approaches to train fairness classifiers have also been studied [13, 14, 17], including modifying dataset fragments to remove sensitive attribute correlations [15], and encoding fairness requirements into classifier training process [22]. Fairness is a special form of probabilistic property. The study of probabilistic properties involves software engineering techniques, as studied in past works, including probabilistic symbolic execution [20][33][16], probabilistic model checking [39][30][11], and probabilistic program analysis / verification [7][38][8].

Automated Program Repair. There is a large literature in the field of automated program repair (e.g. [31] is a recent review of the research area). Automated repair techniques aim to produce patches to meet certain specifications such as passing a given test-suite. Heuristic repair, or search-based repair, iterates over a predefined search space of fixes, and checks the validity of the fixes (e.g. see [42]). To introduce higher flexibility than simple mutations, heuristics like plastic surgery hypothesis [23] or competent programmer hypothesis are used to assume the patch can be produced from (i) other parts of the code [37], or (ii) guided by templates obtained from human patches [28], or (iii) via the use of learning to prioritize patch candidates [32]. In contrast to search-based or template-based works on program repair, our work is more related to constraint-based or semantic repair techniques [34, 35, 43]. In these techniques a repair constraint is inferred as a specification via symbolic execution, and this specification is used to drive a program synthesis step which generates the patch. Previous works on semantic repair are mostly test-driven, where the repair attempts to pass a given test-suite. In contrast, fairness is a hyperproperty [12] which is an accumulative property over collection of traces, and thus fairness repair is a fundamentally different problem from repairing a program based on given tests/assertions.

Fairness in Decision Tree Learning. Other than repairing a trained decision tree or random forest, attempts have been made to integrate fairness into the model training procedure [5][18]. Among them, Kimaran et al. [25] also employ the technique of relabeling leaf nodes in decision trees to achieve fairness requirements. They adopted a different fairness definition that requires the difference (instead of the ratio) of the passing rates to be bounded. They compute the contribution to minimising the difference of flipping each leaf and use a greedy algorithm to accumulate flipped leaves. This definition restricts their algorithm to be applied to binary sensitive attributes only. Aghaei et al. [5] construct an optimization framework for learning optimal and fair decision trees with the aim of mitigating unfairness caused by both biased datasets and machine mis-classification.

DIGITS [6]. The work of [6] performs fairness repair guided by input population distributions. In contrast, our work is focused on rectifying an unfair decision tree from an unfair dataset given as the input. A dataset appear to be similar to an input probability distribution, but they represents two different methodologies. Our approach is able to cure unconscious biases in a decision making entity. With a dataset as input, we first construct a decision tree/random forest with a good fit, then produce a minimal repair to the model. The original models should closely capture the biases in the dataset. This allows us to detect and to cure the implicit and unconscious biases in the previous decision making procedure. The repaired model could be used as a guideline for making future decisions, and it also works as an explanation for the previous unfairness. In contrast, the DIGITS method requires continuing sampling data points from the input distribution to achieve better convergence to the optimal repair. This requires more information of the population, compared to just the past decision results. We are unable to obtain a precise input distribution of the three datasets used in Section 7, hence did not include a comparison with DIGITS.

The completeness guarantee of DIGITS is relative to assumptions such as a manually provided sketch (describing repair model), which is not needed in FairRepair. While its algorithm is designed for binary sensitive attributes, ours is able to handle multiple sensitive groups. Our results are not directly comparable with DIGITS, although they adopted similar fairness criteria and bench-marked on the Adult dataset. Their decision trees were relatively small (less than 100 lines code) and employed at most three features. In contrast, our decision trees employ at least 10 features and scale to real-life applications, with 10K leaves (path hyperrectangles) before splits on sensitive groups. With sensitive attribute being *sex* and fairness threshold set to 0.85, their algorithm was able to produce a repair in 10 minutes with semantic difference of 9.8%. In contrast, FairRepair is able to achieve a semantic difference of 5.7% with a higher fairness threshold 0.9 within the same amount of time. According to DIGITS synthesis algorithm, longer training time allows better convergence to the optimal repair, so it might find a repair that satisfies the fairness requirements at an early stage but far from the optimal. But the comparison does show the efficiency of our tool on both achieving fairness criteria and bounding semantic difference.

REFERENCES

[1] [n.d.]. Adult dataset. https://archive.ics.uci.edu/ml/datasets/Adult. Accessed: 2021/08/08.

[2] [n.d.]. COMPAS dataset. https://www.propublica.org/datastore/dataset/compas-recidivism-risk-score-data-and-analysis. Accessed: 2021/08/08.

[3] [n.d.]. UFRGS dataset. https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/O35FW8. Accessed: 2021/08/08.

[4] A Aggarwal, P Lohia, S Nagar, K Dey, and D Saha. 2019. Blackbox fairness testing of machine learning models. In *International Symposium on Foundations of Software Engineering (FSE).*

[5] Sina Aghaei, Mohammad Javad Azizi, and Phebe Vayanos. 2019. Learning Optimal and Fair Decision Trees for Non-Discriminative Decision-Making. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI.* AAAI Press, 1418–1426.

[6] A Albarghouthi, L D'Antoni, and S Drews. 2017. Repairing Decision-Making Programs under Uncertainty. In *International Conference on Computer Aided Verification (CAV).*

[7] A Albarghouthi, L D'Antoni, S Drews, and A Nori. 2017. FairSquare: Probabilistic Verification for Program Fairness. In *International Symposium on Object-oriented Programming Systems, Languages and Applications (OOPSLA).*

[8] O Bastani, X Zhang, and A Solar-Lezama. 2019. Probabilistic Verification of Fairness Properties via Concentration. In *International Symposium on Object-oriented Programming Systems, Languages and Applications (OOPSLA).*

[9] Dan Biddle. 2017. *Adverse Impact and Test Validation.* Taylor and Francis.

[10] T. Calders, F. Kamiran, and M. Pechenizkiy. 2009. Building Classifiers with Independency Constraints. In *IEEE International Conference on Data Mining Workshops.* 13–18.

[11] Edmund Clarke and Paolo Zuliani. 2011. Statistical Model Checking for Cyber-Physical Systems. In *Automated Technology for Verification and Analysis (ATVA).* Springer, 1–12. https://doi.org/10.1007/978-3-642-24372-1_1

[12] Michael R. Clarkson and Fred B. Schneider. 2010. Hyperproperties. *J. Comput. Secur.* 18, 6 (2010), 1157–1210.

[13] Sam Corbett-Davies, Emma Pierson, Avi Feller, Sharad Goel, and Aziz Huq. 2017. Algorithmic Decision Making and the Cost of Fairness. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)* (Halifax, NS, Canada). Association for Computing Machinery, 797–806. https://doi.org/10.1145/3097983.3098095

[14] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. 2012. Fairness through Awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS)* (Cambridge, Massachusetts). Association for Computing Machinery, 214–226. https://doi.org/10.1145/2090236.2090255

[15] Michael Feldman, Sorelle A. Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. 2015. Certifying and Removing Disparate Impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)* (Sydney, NSW, Australia). Association for Computing Machinery, 259–268. https://doi.org/10.1145/2783258.2783311

[16] Antonio Filieri, Corina S. Păsăreanu, and Willem Visser. 2013. Reliability Analysis in Symbolic Pathfinder. In *Proceedings of the 2013 International Conference on Software Engineering (ICSE)* (San Francisco, CA, USA). IEEE Press, 622–631.

[17] Benjamin Fish, Jeremy Kun, and Ádám Dániel Lelkes. 2016. A Confidence-Based Approach for Balancing Fairness and Accuracy. In *Proceedings of the 2016 SIAM International Conference on Data Mining.* SIAM, 144–152.

[18] Sorelle A. Friedler, Carlos Scheidegger, Suresh Venkatasubramanian, Sonam Choudhary, Evan P. Hamilton, and Derek Roth. 2019. A Comparative Study of Fairness-Enhancing Interventions in Machine Learning. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT).* Association for Computing Machinery, 329–338. https://doi.org/10.1145/3287560.3287589

[19] S Galhotra, Y Brun, and A Meliou. 2017. Fairness Testing: Testing Software for Discrimination. In *Proceedings of the 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE).* 498–510.

[20] Jaco Geldenhuys, Matthew Dwyer, and Willem Visser. 2012. Probabilistic symbolic execution. *International Symposium on Software Testing and Analysis (ISSTA)* (2012), 166–176. https://doi.org/10.1145/2338965.2336773

[21] Vincent Grari, Boris Ruf, Sylvain Lamprier, and Marcin Detyniecki. 2020. Achieving Fairness with Decision Trees: An Adversarial Approach. *Data Science and Engineering* 5 (06 2020). https://doi.org/10.1007/s41019-020-00124-2

[22] Moritz Hardt, Eric Price, and Nathan Srebro. 2016. Equality of Opportunity in Supervised Learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS)* (Barcelona, Spain). Curran Associates Inc., 3323–3331.

[23] Mark Harman. 2010. Automated patching techniques: the fix is in: technical perspective. *Commun. ACM* 53 (01 2010), 108.

[24] Surya Mattu Julia Angwin, Jeff Larson and Lauren Kirchner. [n.d.]. Machine Bias. *ProPublica* ([n. d.]).

[25] Faisal Kamiran, Toon Calders, and Mykola Pechenizkiy. 2010. Discrimination Aware Decision Tree Learning. In *Proceedings of the IEEE International Conference on Data Mining (ICDM).* IEEE Computer Society, 869–874. https://doi.org/10.1109/ICDM.2010.50

[26] Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma. 2012. Fairness-Aware Classifier with Prejudice Remover Regularizer. In *Machine Learning and Knowledge Discovery in Databases*, Peter A. Flach, Tijl De Bie, and Nello Cristianini (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 35–50.

[27] Niki Kilbertus, Mateo Rojas-Carulla, Giambattista Parascandolo, Moritz Hardt, Dominik Janzing, and Bernhard Schölkopf. 2017. Avoiding Discrimination through Causal Reasoning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)* (Long Beach, California, USA). Curran Associates Inc., 656–666.

[28] Dongsun Kim, Jaechang Nam, and Jaewoo Song. 2013. Automatic patch generation learned from human-written patches. *Proceedings - International Conference on Software Engineering*, 802–811. https://doi.org/10.1109/ICSE.2013.6606626

[29] Matt Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. 2017. Counterfactual Fairness. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS).* Curran Associates Inc., 4069–4079.

[30] M. Kwiatkowska, G. Norman, and D. Parker. 2010. Advances and challenges of probabilistic model checking. In *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton).* 1691–1698.

[31] Claire Le Goues, Michael Pradel, and Abhik Roychoudhury. 2019. Automated Program Repair. *Commun. ACM* 62, 12 (2019).

[32] Fan Long and Martin Rinard. 2016. Automatic Patch Generation by Learning Correct Code. In *ACM SIGPLAN Symposium on Principles of Programming Languages (POPL)*.

[33] Kasper Luckow, Corina Păsăreanu, Matthew Dwyer, Antonio Filieri, and Willem Visser. 2014. Exact and approximate probabilistic symbolic execution for nondeterministic programs. In *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering (ASE)*. 575–586. https://doi.org/10.1145/2642937.2643011

[34] Sergey Mechtaev, Jooyong Yi, and Abhik Roychoudhury. 2016. Angelix: Scalable Multiline Program Patch Synthesis via Symbolic Analysis. In *Proceedings of the 38th International Conference on Software Engineering (ICSE)* (Austin, Texas). Association for Computing Machinery, 691–701. https://doi.org/10.1145/2884781.2884807

[35] Hoang Nguyen, Dawei Qi, Abhik Roychoudhury, and Satish Chandra. 2013. SemFix: Program repair via semantic analysis. In *Proceedings of the International Conference on Software Engineering (ICSE)*. 772–781. https://doi.org/10.1109/ICSE.2013.6606623

[36] 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. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[37] Baishakhi Ray, Vincent Hellendoorn, Saheel Godhane, Zhaopeng Tu, Alberto Bacchelli, and Premkumar Devanbu. 2016. On the "naturalness" of buggy code. 428–439. https://doi.org/10.1145/2884781.2884848

[38] Adrian Sampson, Pavel Panchekha, Todd Mytkowicz, Kathryn McKinley, Dan Grossman, and Luis Ceze. 2014. Expressing and Verifying Probabilistic Assertions. *ACM SIGPLAN Notices* 49 (2014). https://doi.org/10.1145/2594291.2594294

[39] Vitaly Shmatikov. 2004. Probabilistic Model Checking of an Anonymity System. *Journal of Computer Security* 12 (2004).

[40] S Udeshi, P Arora, and S Chattopadhyay. 2018. Automated directed fairness testing. In *International Conference on Automated Software Engineering (ASE)*.

[41] Christina Wadsworth, Francesca Vera, and Chris Piech. 2018. Achieving Fairness through Adversarial Learning: an Application to Recidivism Prediction. (06 2018).

[42] Westley Weimer, ThanhVu Nguyen, Claire Le Goues, and Stephanie Forrest. 2009. Automatically Finding Patches Using Genetic Programming. In *Proceedings of the 31st International Conference on Software Engineering (ICSE)*. IEEE Computer Society, 364–374. https://doi.org/10.1109/ICSE.2009.5070536

[43] Jifeng Xuan, Matias Martinez, Favio Demarco, Maxime Clement, Sebastian Lamelas Marcote, Thomas Durieux, Daniel Le Berre, and Martin Monperrus. 2016. Nopol: Automatic repair of conditional statement bugs in java programs. 43, 1 (2016).

[44] Muhammad Zafar, Isabel Valera, Manuel Rodriguez, and Krishna P. Gummadi. 2015. Fairness Constraints: A Mechanism for Fair Classification. (07 2015).

[45] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P. Gummadi. 2017. Fairness Beyond Disparate Treatment & Disparate Impact: Learning Classification without Disparate Mistreatment. In *Proceedings of the 26th International Conference on World Wide Web* (Perth, Australia) *(WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1171–1180. https://doi.org/10.1145/3038912.3052660

[46] Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. 2018. Mitigating Unwanted Biases with Adversarial Learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society* (New Orleans, LA, USA) *(AIES '18)*. Association for Computing Machinery, New York, NY, USA, 335–340. https://doi.org/10.1145/3278721.3278779

[47] Jiang Zhang. [n.d.]. https://github.com/fairrepair/fair-repair. Accessed: 2021/09/01.