Fair Decision Making via Automated Repair of Decision Trees

Jiang Zhang
zhangj@comp.nus.edu.sg
National University of Singapore
Singapore

Ivan Beschastnikh
bestchai@cs.ubc.ca
University of British Columbia
Canada

Sergey Mechtaev
s.mechtaev@ucl.ac.uk
University College London
United Kingdom

Abhik Roychoudhury
abhik@comp.nus.edu.sg
National University of Singapore
Singapore

ABSTRACT

Data-driven decision-making allows more resource allocation tasks 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 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 a MaxSMT solver to decide which paths in the decision tree could be flipped or refined, with both fairness and semantic difference as hard constraints. 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 [10, 13, 16, 18]. While achieving similar fairness by training new models, they incur 8.9% to 13.5% semantic difference. These results show that *FairRepair* is capable of repairing an unfair model while maintaining the accuracy and incurring small semantic difference.

CCS CONCEPTS

• **Computing methodologies** → Philosophical/theoretical foundations of artificial intelligence; • **Social and professional topics** → **Race and ethnicity**.

KEYWORDS

algorithmic fairness, automated program repair, decision trees

FairWare '22 , May 9, 2022, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-9292-1/22/05.
https://doi.org/10.1145/3524491.3527306

ACM Reference Format:
Jiang Zhang, Ivan Beschastnikh, Sergey Mechtaev, and Abhik Roychoudhury. 2022. Fair Decision Making via Automated Repair of Decision Trees. In *International Workshop on Equitable Data and Technology (FairWare '22), May 9, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3524491.3527306

1 INTRODUCTION

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. Past decisions, however, could be influenced by various sources of biases (including human biases), causing the decision making programs 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 [10, 13, 16–18] 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 [11]. Such predictions were used in accessing pre-trial release and sentencing, and altering these decisions at a large scale may cause serious social consequences.

State-of-the-art fair learning approaches usually do not hard constrain fairness and semantic difference [10, 16, 18], hence are unsuitable for repairing models like COMPAS. To the best of our knowledge, the only existing fairness-guided repair approach is DIGITS [5], which relies on probability distributions to generate inputs and to achieve probabilistic guarantees of completeness. We compare the two approaches in Section 6.

In this paper, we study automatic repair of decision-making programs. We present the *FairRepair* approach to derive fair programs while maintaining high accuracy and small semantic difference, without using sensitive attributes in the repair procedure. 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 [5] and [9]. 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, by transforming the decision trees and random forests into their logical equivalents, encoding the fairness and semantic difference criteria as hard constraints, and using a MaxSMT solver to produce repairs. The algorithm then refines the decision tree paths and changes their leaf labels as needed. Our approach is sound and complete [6] with respect to the given dataset, i.e., for any fairness and semantic difference constraints, whenever the constraints are satisfiable, *FairRepair* outputs a repaired model as a solution.

We evaluated *FairRepair* on three publicly available datasets: the 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. We also evaluated four state-of-the-art fair learning algorithms [10, 13, 16, 18]. 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 [7].

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 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 our techniques with respect to *group fairness*. Group fairness is based on the legal guideline for avoiding hiring discrimination, the "80% rule", i.e., 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., $Y_{\text{job}} = \texttt{FALSE}$ 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. The acceptance rate is 60% for male candidate, but 40% for female candidates, hence this model does not satisfy the 80% rule.

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, we 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

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

(a) Hiring example classifier

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

(b) Repaired hiring classifier

Gender	Col	Exp
F	7	8
F	11	6
F	25	2
F	33	20
F	49	13

Gender	Col	Exp
M	6	13
M	14	28
M	28	21
M	38	9
M	43	2

(c) Sample dataset used for repair

Figure 1: Hiring example classifier before and after repair.

(after simplification):

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

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:

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

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: $(col > 10) \times (exp \leq 10)$, FALSE, $(0.2, 0.2)$.
- PH5: $(\mathrm{col} > 10) \times (10 < \mathrm{exp} \leq 15)$, FALSE, $(0, 0.1)$.

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

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

The semantic difference requirement becomes:

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

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 details the *FairRepair* algorithm.

3 PROBLEM FORMULATION

A decision making program P is defined on an input space S, such that $\forall x \in S.\, P(x) \in \{\mathsf{TRUE}, \mathsf{FALSE}\}$. 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, $r_i := \mathbb{P}(P(x) = \mathsf{B} | x \in S_i)$, 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.*

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

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

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

or equivalently p-rule score $\geq c$*, 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, $SD(P_1, P_2, S) := \mathbb{P}(P_1(x) \neq P_2(x) | x \in S)$.*

The Fairness Repair Problem. Given an input space S and a decision program P defined on S and not satisfying group fairness, a *repair* is a 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, $SD(R, \mathrm{P}) \leq \alpha \cdot SD(R_{op}, \mathrm{P})$.*

To solve a *fairness repair problem* is to find an α-optimal repair R. 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. 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 5, 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. Interested readers may refer to a detailed explanation [6] for the random forests repair algorithm and the proof for soundness and completeness. The output decision tree satisfies the fairness requirement, and the semantic difference between the output and the original model, as compared to that of the optimal repair, is bounded by a multiplicative coefficient α. Note that our approach is able to output a solution for any $\alpha > 1$ [6]. We assume that the dataset used for repair correctly represents the input distribution.

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])
```

Algorithm 1 outlines the repair procedure. We first collect all decision tree PHs. 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. 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 an input space S, a dataset $D \subseteq S$ 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, $1 \le i \le n$, 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. 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 $p_i = \sum_{1 \le j \le M} p_{i,j}$, for all $1 \le i \le n$. Define $P_j := \sum_{1 \le i \le n} p_{i,j}$ as the probability of sensitive group j. By definition, $\sum_{1 \le j \le M} p_j = 1$. 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 of Changes

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 $r_j = \sum_{1 \le i \le n, Y_i = \text{TRUE}} p_{i,j}/P_j$. In our previous example, $r_{\text{f}} = 0.4$ and $r_{\text{m}} = 0.52$ for female and male groups. Recall the fairness requirement: $\forall 1 \le i, j \le M, \frac{r_i}{r_j} > c$, 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 $1 \le i \le M$, 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:

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

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. 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. We first have the fairness requirements.

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

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. Next, we account for the semantic difference.

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

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.

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

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 5 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 $1 \leq k \leq |A_i|$.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 $i \neq j$. 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):

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

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

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

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:

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

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. 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.

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

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 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.

5.1 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.

1. The Adult dataset [1] 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.

2. 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.

3. The COMPAS [2] dataset contains 13 attributes over previously convicted criminals. The class labels the defendants' recidivism states within two years. We follow [10] and [15], 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 [14]. 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. When evaluating generalisation accuracy, we split the datasets 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 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.

5.2 Fairness and Accuracy

We evaluated 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 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 use the adult dataset for explanation. For *gender*, the accuracy at $c = 0.95$ is 76.1%. The initial passing rates computed from the predicted results are 31.0% for males and 11.3% for females. To balance the passing rates, the algorithm has to explicitly make wrong predictions, which decreases accuracy. 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.

In real life, a high fairness threshold $c = 0.95$ 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%. We believe this is a reasonable trade-off for a more fair model, and such results do show that *FairRepair* is able to repair unfair models, while their accuracy is mostly maintained.

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) [10, 13, 16, 18]. These works train new classifiers from scratch, instead of repairing existing unfair models. We 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	**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.

Model	Adult		COMPAS	
	Acc.	**Sem.D.**	**Acc.**	**Sem.D.**
FairRepair (tree)	76.2%	5.7%	59.9%	2.3%
FairRepair (forest)	80.9%	5.6%	64.2%	3.7%
FAGTB [10]	85.0%	10.9%	64.6%	8.9%
Kamishima [13]	82.6%	11.7%	64.0%	12.1%
Zafar [16, 17]	82.3%	9.5%	63.9%	13.5%
Zhang [18]	83.1%	10.0	64.1%	9.7%

Table 2: Comparisons between *FairRepair* (at $c = 0.9$) and other fairness learning algorithms (at around 90% fairness) on accuracy and semantic difference.

[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.

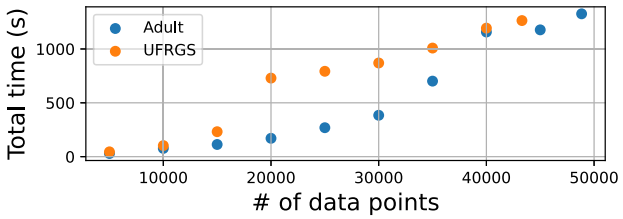

Figure 2: The total running time of *FairRepair* for random forests with increasing number of data points.

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.

5.3 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 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.

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

6 RELATED WORK

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 [4][8]. Among them, Kimaran et al. [12] 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. [4] 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 [5]. The work of [5] 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. 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. 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, as their decision trees were relatively small (less than 100 lines code) and employed at most three features. 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. The comparison does show the efficiency of our tool on both achieving fairness criteria and bounding semantic difference.

ACKNOWLEDGEMENT

This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme.

REFERENCES

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

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

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

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

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

[6] Anon. [n.d.]. FairRepair. https://github.com/fairrepair/fair-repair/blob/master/FairRepair.pdf.

[7] Anon. [n.d.]. FairRepair. https://github.com/fairrepair/fair-repair.

[8] 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)*. 329–338.

[9] 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.

[10] 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

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

[12] 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.

[13] 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.

[14] 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.

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

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

[17] 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

[18] 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.