Fairness Invariants: A Relational Approach to Explaining and Mitigating Fairness Bugs

RANIT DEBNATH AKASH, University of Illinois at Chicago, USA
ASHISH KUMAR, Pennsylvania State University, USA
GANG TAN, Pennsylvania State University, USA
SAEID TIZPAZ-NIARI, University of Illinois at Chicago, USA

Data-driven software systems are increasingly deployed in high-stakes socio-economic domains, from criminal justice to financial lending. However, these systems often exhibit individual discrimination—unjustified disparities in which a program yields different outcomes for similar individuals who differ only in their protected attributes (e.g., race, gender, age). While existing research has focused on detecting and quantifying these bugs, there remains a critical lack of principled mechanisms to explain and localize individual fairness bugs. Current explanation techniques are largely designed for single-input decisions rather than the relational nature of discrimination, which inherently involves a comparison between an original and a counterfactual pair.

We present Remi, a framework for the automated localization, explanation, and mitigation of individual discrimination. Inspired by loop-invariant synthesis in formal methods, we treat counterfactual fairness as a relational invariant discovery problem. We introduce a bidirectional relational explanation framework that learns over paired examples (x, x') to identify regions of the input space where fairness is violated. Unlike traditional one-way implication pairs used in invariant inference, our approach enforces bidirectional constraints: requiring identical outcomes for both original and counterfactual samples. Remi utilizes three data-alignment techniques to infer interpretable rule-based models that act as "fairness invariants." These rules serve as guardrails to selectively block or relabel unfair predictions without requiring model retraining. Our evaluation on symbolic and neural network programs demonstrates that Remi localizes ground-truth fairness bugs in over 83% of cases, significantly outperforming state-of-the-art baselines and reducing discriminatory decisions in black-box models by up to 70%.

CCS Concepts: • **Software and its engineering** → **Software testing and debugging**; • **Computing methodologies** → **Machine learning**; *Rule learning*.

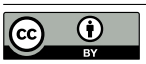
Additional Key Words and Phrases: Machine Learning, Bias Mitigation, Fairness, AI Ethics, Interpretability

ACM Reference Format:
Ranit Debnath Akash, Ashish Kumar, Gang Tan, and Saeid Tizpaz-Niari. 2026. Fairness Invariants: A Relational Approach to Explaining and Mitigating Fairness Bugs. *Proc. ACM Softw. Eng.* 3, ISSTA, Article ISSTA019 (October 2026), 24 pages. https://doi.org/10.1145/3832110

1 Introduction

Automated decision-support software systems have become foundational to modern socio-economic infrastructure. They have been used to make critical decisions in domains such as criminal justice, healthcare, financial lending, and hiring. However, because these models often learn from historical datasets, they risk encoding and amplifying biases related to protected attributes like race, gender, or disability status. Hence, ensuring their fairness has emerged as a critical requirement.

Authors' Contact Information: Ranit Debnath Akash, University of Illinois at Chicago, Chicago, USA, rakas@uic.edu; Ashish Kumar, Pennsylvania State University, University Park, USA, azk640@psu.edu; Gang Tan, Pennsylvania State University, University Park, USA, gtan@psu.edu; Saeid Tizpaz-Niari, University of Illinois at Chicago, Chicago, USA, saeid@uic.edu.



ACM 2994-970X/2026/10-ARTISSTA019
https://doi.org/10.1145/3832110

In high-stakes scenarios, this manifests as "fairness bugs"—unjustified disparities where the software yields different outcomes for individuals who are identical in all relevant qualifications, but differ in a protected characteristic. For instance, studies on FICO scoring found that black non-defaulters were often assigned higher risk scores than their white counterparts [32].

In response to these risks, the software engineering and machine learning communities have developed a diverse array of techniques to detect, explain, and mitigate bias through pre-processing data [15, 16], in-processing algorithmic adjustments [31, 49, 59], and post-processing calibration [32]. While these efforts have significantly improved our ability to address fairness issues, a principled mechanism for explaining individual fairness bugs remains largely lacking. Current explanation techniques primarily focus on local interpretability, explaining why a model made a specific decision for a single input [52]. However, individual fairness is inherently relational: it cannot be violated by a single input, but requires an analysis between a specific individual x and their "similar" counterpart x' who may differ only in their protected attributes. We identify two core limitations.

(1) **The Localization Insufficiency**: Existing tools provide explanations for model decisions (e.g., "why was this loan denied?"), but they lack the framework to explain discriminatory outcomes (e.g., "why was this loan denied to x, but granted to x'?"). We currently lack a way to consider multiple similar inputs simultaneously to pinpoint the exact logic driving the disparity.
(2) **The Mitigation Problem.** Developers require compact and precise characterizations of how the counterfactual unfairness occurs. Without a way to localize these "fairness bugs" to specific relational constraints, mitigation strategies are limited and ineffective.

Intuition. Inspired by loop-invariant synthesis from the programming language literature [56], we develop a relational explanation and mitigation technique for software fairness. In loop-invariant synthesis, the goal is to identify a region of the program state space that captures all reachable program states. The key is to enforce 'implication pairs': if a state satisfies the head of an implication pair, then it must also satisfy the tail (a positive instance). Decision tree inference with implication pairs [29] formalizes this by classifying states into positive and negative examples. The learned decision tree must therefore respect these unidirectional relational constraints in addition to separating positive and negative instances. The relational fairness problem can be viewed as an analogous task.

Key Observation. Instead of reasoning about reachable program states, we infer fairness invariants of automated decision-making software and identify regions of the input space where the original x (head) and counterfactual x' (tail) pairs disagree. Each relational pair (x, x') acts as a constraint, requiring that the program assign the same outcome to both original and counterfactual samples that may only differ in their protected attributes. Unlike the one-directional implication pairs used in invariant inference, these counterfactual constraints are *bidirectional*: if one element of the pair receives a favorable outcome, the other must as well for fairness to hold, and likewise for unfavorable outcomes. Hence, the approach for using decision trees to respect unidirectional relational constraints can be extended to bidirectional constraints, enabling us to use decision trees to solve the counterfactual fairness problem.

Approach. We present Remi (Relational Explanation and MItigation), a framework to localize, explain, and mitigate individual fairness bugs. Remi first generates counterfactual instances to identify discriminatory pairs. It then constructs a relational dataset where pairs are labeled '+' if the model treats them identically (fair) and '-' if the outcomes differ (unfair). We propose three alignment techniques—original, horizontal, and vertical extension—to structure this relational data. From this, Remi infers interpretable rules that distinguish fair regions from unfair ones Finally, Remi extracts these discriminatory rules as guardrails to selectively block or relabel unfair predictions, mitigating bias without requiring model retraining.

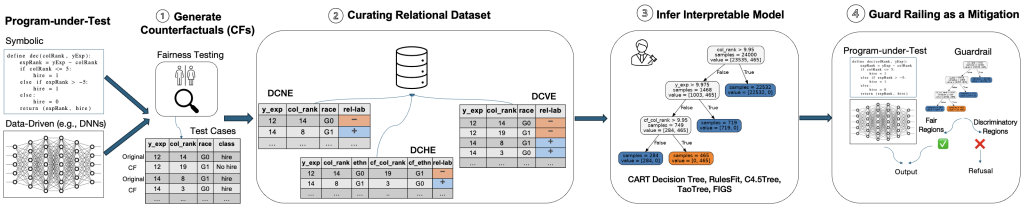


Fig. 1. Remi Framework.

Experiments. We perform experiments both on symbolic (rule-based) and neural network (black-box) programs. Since the ground truth for the symbolic programs is known and verifiable, we first use Remi to answer four research questions about i) the performance of three relational dataset curation techniques; ii) the performance of nine different rule inference algorithms; iii) the characteristics and precision of extracted rules vs. the ground truth; and iv) the performance of guardrails technique for a mitigation, based on the extracted discriminatory rules. We find that horizontal extension outperforms the baseline and other alignment methods, and C4.5Tree, FIGS, and CART tree inferences outperform other rule-based inference methods. The alignment and inference techniques localize the ground truth in more than 83% of the cases, compared to the state-of-the-art AFT [71] approach. Additionally, the rule-based guardrails reduce individual instances of discrimination in all cases. Finally, we study whether our experiences with Remi over symbolic programs generalize to black-box neural networks. We found that applying guardrail rules to neural networks reduces individual discriminatory decisions by at least 40%, up to 70%, significantly outperforming the retraining-based bias mitigation techniques [24, 60, 70, 71]

Contributions. The key contributions of this paper are:

- Inspired by the loop invariant synthesis, we design a data-driven approach to infer fairness invariants for automated decision-making programs,
- We develop a novel guardrail strategy to mitigate individual discrimination,
- We put forward Remi, a framework that automatically detects, localizes, explains, and mitigates individual fairness bugs.
- Our experiments on both rule-based symbolic and data-driven neural network programs show that Remi significantly outperformed the baseline techniques in terms of localizing and mitigating individual discrimination.

2 Overview

Following our intuition that individual fairness is a relational property that cannot be violated over a single point x. But it requires a relational analysis between the original and counterfactual inputs (x, x') that only differ in their protected attributes. In light of this individual fairness [23], a decision by decision-making programs-under-test (DPuTs) is considered unfair (discriminatory) if only flipping the protected attributes (and any logically dependent non-protected attributes) changes the outcome of DPuTs. We demonstrate that explaining the DPuT's decision for a single data point, e.g., local explanation methods [52] is insufficient.

Remi Workflow Summary. Here we will give an overview of how Remi works. Remi is a tool that identifies and explains individual fairness violations within decision-making programs. It identifies the specific logic that causes a program to produce discriminatory outcomes based on protected attributes, such as race. Our workflow consists of four main stages, as illustrated in Figure 1. It has two elements: (1) the decision-making program under test (DPuT) and (2) a dataset containing protected and non-protected attributes.

A. Counterfactual generation and finding discriminatory instances. The first stage is to build a relational data point. In doing so, Remi generates a counterfactual instance for any given input by changing the protected attributes. A relational data point becomes discriminatory (unfair) when the DPuT gives different outcomes for each input in the pair.

B. Relational dataset construction and curation. Evaluating the DPuT on both x and x' yields a relational label: + if both outcomes agree, and − if they disagree (a fairness bug). Remi introduces three data curation schemes, which all keep the relational label $\mathcal{L}$ "fair (+) vs unfair (−)", but transform the features differently:

- *Data curation with no extension (DCNE).* This approach keeps only the original point (x), but the label of the point is the corresponding relational label $\mathcal{L}$ as derived by comparing the outcome of DPuT on x against the counterfactual point x'.
- *Data Curation with vertical extension (DCVE).* It augments the original data point x with its counterfactual (x') as a separate instance. The labels of both instances are common and equal to the relational label $\mathcal{L}$. This doubles the number of training instances and exposes the tree learner to more examples of fair and discriminatory profiles.
- *Data Curation with horizontal extension (DCHE).* This method concatenates original (x) and counterfactual (x') features into a single extended feature vector, keeping only the features that are not identical in both ($x_i \neq x'_i$). This explicitly exposes interactions between original vs counterfactual values that correlate with unfairness.

Across all the preprocessing techniques, the target is the relational label $\mathcal{L}(x, x')$, i.e., (+ or −). The non-protected attributes (and, for DCHE, their counterfactual counterparts) are the only inputs to the interpretable learner, so that the learned rules describe where the DPuT is unfair.

C. Interpretable learning and rule extraction. Given the relational dataset, Remi instantiates the explainer $\mathbb{E}$ using a set of tree-based algorithms (e.g., CART, C4.5, TaoTree, RulesFit, FIGS). Our target is to retrieve a performant, interpretable learner that can explain what features distinguish fair relational samples from the unfair ones. The tree partitions the curated feature space into regions whose leaves are labeled with relational labels. From leaves with a high proportion of unfair relational labels, Remi extracts if–then rules that characterize discriminatory subspaces (e.g., "age $\leq 20.5 \wedge$ priors_count ≤ 3.5" or "college rank = $50 \wedge$ years-of-experience ≤ 1 year").

D. Guardrail-based mitigation. Finally, Remi uses the extracted rules as guardrails at the final debiased model during deployment time. When a new input falls into an unsafe region, the framework can deny automatic prediction and defer to human review. Because guardrails are enforced selectively on localized regions, Remi avoids heavy-handed global post-processing that might degrade accuracy or introduce new fairness violations elsewhere.

Listing 1. FairSquare [8] populaiton generation

```
def popModel()
  ethnicity = gauss(0,10)
  colRank = gauss(25,10)
  yExp = gauss(10,5)

  if ethnicity > 10:
    colRank = colRank + 5

  return (colRank, yExp)
```

Listing 2. FairSquare [8] hiring decision

```
def dec(colRank, yExp):
    expRank = yExp - colRank
    if colRank <= 5:
        hire = 1
    else if expRank > -5:
        hire = 1
    else:
        hire = 0
    return (expRank, hire)
```

Overview Example: We use an example of a hiring program, taken from Albarghouthi et al. [8], to illustrate our approach (see Listings 1,2), where the ethnicity is a protected attribute, and the college rank of an individual is causally and directly affected by their ethnicity (e.g., being Hispanic and

attending a Hispanic-Serving Institution). Specifically, a low college rank and more years of prior job experience are key to hiring. The program's decision function appears fair, as it uses only college rank and years of experience to decide hiring, but an upstream generative model causally influences college rank.

Our approach starts by generating the population (following Listing 1) and passing those samples to obtain the score and hiring decision from `dec` program 2. Remi first generates counterfactual applicants by changing ethnicity and adjusting affected attributes, then labels pairs where the hiring decision flips as unfair. For this example, Remi generates 30,000 applicant samples from the program and their corresponding counterfactual with a different ethnicity. Our sampling finds 2,250 individual discriminatory instances, applicant pairs that differ only in ethnicity, but receive different hiring decisions (labeled '-'). Note that the rest (27,750 samples) is fair w.r.t. ethnicity and labeled '+'. We are interested in explaining and mitigating the individual discriminatory bugs in this program.

Which curation scheme best enables discrimination localization? Among the three relational dataset curation, we found that DCNE infers models with very high predictive metrics (accuracy around 0.99, Precision close to 0.99), leading to the localization of bugs in up to 2,234 out of the 2,250 individual discriminatory instances (IDIs). This convinces us that the relational labels in the training data are a key driver of root-cause localization quality.

Which interpretable learner is the best at identifying discriminatory regions? We compare multiple interpretable learners (e.g., CART, C4.5, TaoTree, rule-based baselines), while fixing the data curation method to DCNE. We found that a C4.5Tree [51] achieves superior performance and localizes 2,237 IDIs among the 7 tree-based and interpretable algorithms.

How does our approach perform to pinpoint the root cause of the fairness bug? Our relational decision trees yield compact rules that localize unfair regions of interest. In total, it retrieves 97 rules and covers 98% of the considered discriminatory instances. For example, one rule captures 301 of the 2250 IDIs (13.4% discrimination coverage) is the following: $\{\ 14.6 < \texttt{col-rank} \le 17.25 \wedge 11.99 < \texttt{y-exp} \le 14.84 \wedge \texttt{cf-ethnicity} > 10.5 \wedge 17.85 < \texttt{cf-col-rank} \le 22.95\ \}$ where it finds a narrow region of college rank and year of experience when the DPuT of PG1 becomes significantly discriminatory based on the ethnicity. Intuitively, the rule recovers the root cause of the discrimination: in a narrow mid-college rank, mid-experience range (14.6< col-rank ≤ 17.25, 11.99 < y-exp ≤ 14.84), switching to an unfavorable ethnicity increases the counterfactual college rank (17.85 < cf-col-rank ≤ 22.95) and flips the hiring decision. Because this rule explicitly captures the ethnicity-to-college rank-to-hire pathway, it demonstrates a very close match with the ground truth level root cause of the discrimination in this hiring program.

How to Mitigate Unfairness: We treat the learned rules as guardrails around the original program. On this program, inserting discriminatory rules as guardrails reduces the number of discriminatory instances from 2,250 to just 2, while preserving other similar classification metrics. We note that naively replacing the program with an unconstrained decision tree trained on counterfactual labels actually increases the number of fairness bugs to 2,430 on unseen samples.

3 Relational Explanation and Mitigation Problem

We consider any Decision-Making Program under Test (DPuT), which is essentially a function that maps an input describing an individual into a binary decision, such as *favorable* (e.g., loan approved) or *unfavorable* (e.g., loan denied). The input features for each individual can be grouped into two disjoint categories: *Protected attributes* (A_p) such as race or gender, which by fairness considerations should not affect the decision, and *Non-protected attributes* (A_{np}) such as income, age, or education level. Formally, the DPuT can be represented as a function

$$f(A_p, A_{np}) \ \rightarrow \ Y, \quad Y \in \{\text{favorable}, \text{unfavorable}\}.$$

Fairness Notion. We use an individual fairness definition that requires two similar individuals should receive similar outcomes irrespective of their protected attributes [23]. Following the counterfactual fairness definition [38], a decision by DPuT is considered unfair (discriminatory) for an individual if only changing the protected attributes (and any logically dependent non-protected attributes) flips the decision, i.e., $f(x_p, x_{np}) \neq f(x'_p, x'_{np})$ where $x_p \neq x'_p \wedge x_{np} \sim x'_{np}$.

Relational Labeling for Discrimination. We are given a DPuT together with an input dataset $\mathcal{D}$. To evaluate fairness, we transform $\mathcal{D}$ into a relational dataset, where pairs of datapoints are explicitly labeled as fair or unfair. We formalize this notion below:

Definition 3.1 (Relational Dataset). A relational dataset is a pair $\mathcal{D}_R = (\mathcal{D}', R)$ where $\mathcal{D}' = \{x_1, \ldots, x_n\}$ is a set of datapoints and $R : \mathcal{D}' \times \mathcal{D}' \to \{+, -\}$ is a labeling function that assigns to each pair (x_i, x_j) either + (indicating a fair pair) or − (indicating an unfair pair).

For each datapoint $x_i = (a_{p_i}, a_{np_i}) \in \mathcal{D}$, where $a_{p_i} \in A_p$ are the protected attributes and $a_{np_i} \in A_{np}$ are the non-protected attributes, we construct a corresponding counterfactual datapoint x'_i. The counterfactual is obtained by modifying the protected attributes, denoted a'_{p_i}, while keeping the non-protected attributes unchanged, except in cases where a change in the protected attribute necessitates an adjustment to some non-protected attributes (e.g., changing sex from male to female requires adjusting the relationship attribute from "husband" to "wife"). Formally,

$$x'_i = (a'_{p_i}, \widehat{a_{np_i}}), \quad \text{where } a'_{p_i} \neq a_{p_i},$$

and $\widehat{a_{np_i}}$ denotes the non-protected attributes of x_i, possibly updated to maintain consistency with the change in a'_{p_i}.

We then evaluate the DPuT on both the original datapoint and its counterfactual, obtaining outcomes $y_i = f(x_i)$ and $y'_i = f(x'_i)$, where f is the function modelling the DPuT. This comparison induces a relational label on the pair (x_i, x'_i):

- **Agreement (+):** The outcomes agree, indicating fair treatment.

$$L(x_i, x'_i) = + \quad \text{if } y_i = y'_i$$

- **Disagreement (−):** The outcomes differ, indicating potential counterfactual unfairness.

$$L(x_i, x'_i) = - \quad \text{if } y_i \neq y'_i$$

In this construction, the set $\mathcal{D}'$ of the relatioanl dataset consists of all original datapoints together with their counterfactual counterparts, and the relation R is defined by the agreement or disagreement labels assigned to each pair. The key challenge is not to explain the overall decision function $f(x)$ where $x \in \mathcal{D}$. Instead, our focus is on explaining the relational label $L(x, x')$, which captures whether a pair of datapoints is treated fairly or unfairly.

Our goal is to learn a human-interpretable model E that, can characterize the conditions under which a pair receives a disagreement label (−), i.e., when the Decision-Making Program under Test (DPuT) exhibits unfair behavior. In other words, we want to answer the question: **Informal Problem Statement.** *For which regions of the input space, defined solely by non-protected attributes, does the DPuT exhibit counterfactual unfairness?*

Formal Problem Statement Let our relational dataset be

$$\mathcal{D}_{\mathrm{R}} = \{((x_1, x'_1), L_1), ((x_2, x'_2), L_2), \ldots, ((x_n, x'_n), L_n)\}$$

where each $x_i = (a_{p_i}, a_{np_i})$, $x'_i = (a'_{p_i}, \widehat{a_{np_i}})$, and the label $L_i \in \{+, -\}$ indicates whether the outcomes of the DPuT on (x_i, x'_i) agree or disagree. For convenience, we denote $Z_i = (x_i, x'_i)$.
Explanation. Our goal is to learn an interpretable explanation model E, instantiated as a decision-tree based learner (e.g., CART, C4.5, FIGS, RulesFit, or similar variants). The model E predicts whether

a pair Z_i will receive an agreement (+) or disagreement (−) label. In practice, decision tree training is framed as a binary classification problem, where the learning algorithm recursively partitions the feature space so as to minimize an impurity measure at each split. For classification, the impurity at a node is most commonly measured using the *cross-entropy* between the two possible classes - agreement and disagreement. If p_+ and p_- denote the empirical probabilities of the labels + and −, respectively, among the datapoints at a node, then the cross-entropy is then given by

$$H(p_+, p_-) = -p_+ \log p_+ - p_- \log p_-$$

Thus, the explanation model E is a set of human-interpretable rules that delineate regions of the feature space Z where the DPuT is most likely to exhibit counterfactual unfairness.

Mitigation. Since the learned model E characterizes the discriminatory regions purely in terms of attributes, the rules describing its disagreement (−) regions can be extracted as *guardrails*. Let $\mathcal{R}^-$ denote this rule set, inducing a guardrail predicate $g(a_{np}) = 1$ iff a_{np} satisfies some rule $r \in \mathcal{R}^-$, i.e., the input lies in a region where f is likely to discriminate. We then deploy a *guarded* decision function $f_g(x) = f(x)$ when $g(a_{np}) = 0$, and $f_g(x) = \perp$ when $g(a_{np}) = 1$, where $\perp$ denotes withholding the automatic outcome and deferring the case to human review (or, alternatively, relabeling it to restore a consistent outcome across the pair (x, x')). The mitigation objective is to choose $\mathcal{R}^-$ so that f_g minimizes the residual counterfactual unfairness while keeping both the benign inputs it unnecessarily blocks and the loss in predictive utility small.

4 Approach

We define a region as fair if the DPuT's outcome remains invariant under changes to protected attributes; otherwise, the region is unfair, indicating the existence of Individual Discriminatory Instances (IDIs). To localize these bugs, our tool Remi, employs an interpretable decision tree to classify fair versus unfair regions. This model captures feature interactions and enables the extraction of precise decision rules that characterize the discriminatory input space. As illustrated in Figure 1 and Algorithm 1, our framework consists of four primary steps:

- Section 4.1: Generating counterfactuals to detect IDIs.
- Section 4.2: Curating relational datasets from identified pairs.
- Section 4.3: Training an interpretable model to extract explanation rules for discrimination.
- Section 4.4: Deploying these rules as guardrails to mitigate unfairness.

4.1 Counterfactual Generation and Finding Discriminatory Instances

For each of our different relational dataset construction and training approaches and programs (DPuTs), this is a common step. We take as input a decision-making program under test `DPuT`, which has access to sensitive attributes from the original dataset (X_{orig}), the program will be applied to. After applying the program `DPuT` on X_{orig}, we can retrieve their target (favorable or unfavorable) outcome Y_{orig} (Line 1 in Algorithm 1). Then we create a counterfactual x_i' for each data point $x_i \in X_{orig}$ based on its protected attribute. For our symbolic programs, we just flip the protected attribute (and update accordingly related dependent non-protected attribute) of to get the corresponding counterfactual entry ($x_i' \in X_{cf}$), and retrieve the target outcome for counterfactual (Y_{cf}) based on the decision of DPuT (Line 2-5 in Algorithm 1). Here x_i' is the corresponding counterfactual $\text{CF}(x_i)$ of x_i (Line 3). Here, we consider a protected attribute in a binary setting, grouping it into privileged or unprivileged groups. Then we audit for fairness violations with respect to the protected attribute, checking whether the outcomes for the corresponding counterfactual pairs change. When it does, we call these fairness violation instances individual discriminatory instances (IDI). There are several methods [5, 9, 24, 60, 71] to find these discriminatory instances. For data-driven DPuTs, e.g., DNNs, we also use the Themis [9] implementation by Zhao et al. [71], ExpGA [24], and LIMI [63] to identify

and merge these IDIs. Here, based on the X_{orig}, Y_{orig}, DPuT, and the corresponding IDI search, we can get their corresponding counterfactual input/output X_{cf}, Y_{cf} (Line 2-5 in Algorithm 1).

4.2 Curating Relational Dataset

From the DPuT programs (symbolic or black-box), original instances, and counterfactuals (and IDIs), we get two types of pairs in combination with original (X_{orig}, Y_{orig}) and counterfactual (X_{cf}, Y_{cf}) data points. One set of inputs or data points, even when their protected attributes have been flipped (from privileged to unprivileged or vice versa), both get the same target outcome from the DPuT. We refer to them as fair (or agreement) pairs and assign the relational label + to them. Another set of inputs or data points, where their protected attributes have been flipped, and where both points do not receive the same target outcome (favorable vs unfavorable) from the predictive model or DPuT. We refer to them as discriminatory (or disagreeable) pairs, and assign the relational label − to them. We store these relational labels (fair vs discriminatory) corresponding to each data in Y_{disc} (Line 6-8 in Algorithm 1). And here we create our relational dataset $\mathcal{D}_{\text{R}}$.

$$\mathcal{D}_{\text{R}} = \{((x_1, x_1'), L_1), ((x_2, x_2'), L_2), \ldots, ((x_n, x_n'), L_n)\}$$

Remi transforms $\mathcal{D}_{\text{R}}$ into a supervised training dataset $\mathcal{SD}_{\text{m}}$ for the classification of discriminatory pairs from the fair pairs using one of the following curation operator $m \in \{DCNE, DCVE, DCHE\}$.

$$\mathcal{SD}_{\text{m}} \leftarrow \{(u_i, L_i)\}_{i=1}^{n}$$

4.2.1 Approach 1: Dataset curation with no extension (DCNE). For DCNE, only the original instance x_i from each counterfactual pair is retained, annotated with the relational label L_i encoding whether the pair (x_i, x_i') produced a fair or discriminatory outcome:

$$\mathcal{SD}_{\text{DCNE}} \leftarrow \{(x_i, L_i)\}_{i=1}^{n}$$

DCNE preserves relational information through labeling rather than explicit pairing: L_i encodes the outcome consistency between x_i and its counterfactual x_i', so the learned model captures which non-protected feature values trigger sensitivity to the protected attribute, without storing counterfactual features directly. The classification label Y_{orig} is dropped. This avoids introducing potentially unsound feature combinations, which is particularly important for black-box DPuTs where counterfactual validity cannot be verified.

4.2.2 Approach 2: Dataset Curation With Vertical Extension (DCVE). In this curation technique, both sets of input from each pair (x_i, x_i') are used, and the classification label is determined based on their relational label L_i (whether they are from a fair or unfair pair). But each element of the pair (original x_i and counterfactual i.e., x_i') is treated as a separate data point, and their corresponding relational labels are L_i. So we have one pair $((x_i, L_i))$ and another pair $((x_i', L_i))$ In this way our dataset size doubles in terms of rows. The intuition behind this form of training is that if we add more instances from both fair and unfair pairs, and a better pattern emerges in the classification boundary, it will help the interpretable model distinguish unfair pairs from fair ones.

$$\mathcal{SD}_{\text{DCVE}} \leftarrow \{(x_i, L_i), (x_i', L_i)\}_{i=1}^{n}$$

where size of the dataset is $N_{DCVE} = 2 \times n$

4.2.3 Approach 3: Dataset Curation With Horizontal Extension (DCHE). For this curation technique, we also take information from both set of input from each pair (x_i, x_i'), but they are merged into a single row. We add columns from both sources ($x_i \in X_{orig}$ and $x_i' \in X_{cf}$) as training features which has different values, dropping one of the columns that has the same values in both sources. Let

Algorithm 1: Remi Discrimination identification and verification

Input: Program under test DPuT, Dataset X_{orig}, Curation $appr$: 'DCNE', 'DCVE', or 'DCHE'
Output: mitigation rules $\mathcal{R}$, Deferred or fair prediction y_{pred}

$Y_{orig} \leftarrow$ getOutcome(X_{orig}, DPuT)
for $i \leftarrow 1$ **to** $|X_{orig}|$ **do**
  $x'_i \leftarrow \text{CF}(x_i)$; // Get corresponding counterfactual input
  $y'_i \leftarrow f(x'_i)$
  X_{cf}.add(x'_i), Y_{cf}.add(y'_i)
foreach $(y_i, y'_i) \in (Y_{orig}, Y_{cf})$ **do**
  $L_i \leftarrow \mathbf{1}[y_i \neq y'_i]$
  Y_{disc}.add(L_i) ; // Add aggreement/disagreement relational labels
if $appr == DCNE$ **then**
  $\mathcal{SD}_{\text{DCNE}} \leftarrow \{(x_i, L_i)\}_{i=1}^{n}$
else if $appr == DCVE$ **then**
  $\mathcal{SD}_{\text{DCVE}} \leftarrow \{(x_i, L_i), (x'_i, L_i)\}_{i=1}^{n}$
else if $appr == DCHE$ **then**
  $\mathcal{SD}_{\text{DCHE}} \leftarrow \{(h(x_i, x'_i), L_i)\}_{i=1}^{n}$
$\mathcal{E}, M \leftarrow \mathcal{IM}_{train}(S_{appr}, depth_{\max}, Leaf_{\max})$;
foreach *leaf* λ *in* $\mathcal{E}$ **do**
  if *isDiscriminatoryLeaf*(λ) **then**
    $\pi \leftarrow extractRuleForLeaf(\lambda)$
    $\mathcal{R}$.add(π) ;
debugAndGuardRail($\mathcal{R}$, M) ; // Using rules/explanation to guard rail the model
foreach x_{pred} *in* X_{pred} **do**
  if *doesFallUnderDiscriminatoryRule*($\mathcal{R}$,x_{pred}) **then**
    Deny output
  else
    $y_{pred} \leftarrow DPuT(x_{pred})$
return ($\mathcal{R}$, y_{pred})

$C = |x_i|$, where C is the number of features we have, for x_i. We can write $x_i = (v_{i,1}, ..., v_{i,C})$ and $x'_i = (v'_{i,1}, ..., v'_{i,C})$, where $v_{i,1}$ represent the first feature of the x_i instance. And we can have a mask $m_{i,k} = 1[v_{i,k} \neq v'_{i,k}] \in \{0, 1\}$ to drop the features from x'_i which are the same as x'_i. If we denote the DCHE horizontal feature set as $h(\cdot)$, then

$$h(x_i, x'_i) = (v_{i,1}, ..., v_{i,C}, m_{i,1} * v'_{i,1}, ..., m_{i,C} * v'_{i,C})$$

For $m_{i,k} = 1$, it will add the counterfactual values as they differ, and for $m_{i,k} = 0$, it will drop the identical features.

$$\mathcal{SD}_{\text{DCHE}} \leftarrow \{(h(x_i, x'_i), L_i)\}_{i=1}^{n}$$

And the classification label for each instance is its corresponding relation label L_i. So, we get with relational dataset $\mathcal{SD}_{\text{DCHE}}$. The intuition behind this form of training is that for each point, we add more information, so that in terms of both original and counterfactual values, the models get more intuition about where the fairness bugs are mostly located.

4.3 Interpretable Learning and Synthesizing the explanation

Given a curated relational dataset SD_m (where $m \in \{DCNE, DCVE, DCHE\}$), we train an interpretable model $\mathcal{M}$ (e.g., a decision tree) to classify fair versus unfair relational pairs (Alg. 1, Line 15). Interpretability is essential to capture the feature interactions that localize discriminatory regions within the DPuT. Following AFT [71], we constrain the model's complexity by limiting the number of leaves. We evaluate $\mathcal{M}$ using standard metrics—Accuracy, Precision, Recall, and ROC-AUC—where True Positives (TP) represent correctly identified IDIs, True Negatives TN stands for correctly identified non-IDI, and accuracy measures the fraction of all data points (both fair and unfair) that are correctly classified. Impurity metric depends on the specific tree learner e.g. ID3 and C4.5 algorithms use cross-entropy, whereas CART decision trees use Gini impurity.

Once $\mathcal{M}$ achieves sufficient discriminative performance, we extract the path predicates leading to leaves λ classified as discriminatory. These paths form a set of explanatory rules $\mathcal{R} = \{\phi_\pi\}$ that localize the specific attribute ranges where discrimination occurs (Alg. 1, Lines 16–19). We validate each rule ϕ_π by subsetting the relational dataset using the rule's conjunction of predicates:

$$SD_m[\phi_\pi] = \{(u_i, L_i) \in SD_m \mid \phi_\pi(u_i) = \text{true}\}$$

For each rule, we compute a few key performance metrics, e.g., impurity, Confidence_{leaf}, and Coverage_{leaf}, to assess the strength of each rule. Confidence_{leaf} describes the precision of the corresponding rule, i.e., the fraction of instances captured by the rule π that are IDI cases.

$$\text{Confidence}_{leaf}(\pi) = \frac{\text{TP}_{\text{leaf}}}{\text{TP}_{\text{leaf}} + \text{FP}_{\text{leaf}}}$$

Here TP_{leaf} stands for the number of actual IDIs that fall under this leaf, and FP_{leaf} stands for the number of points that were classified as IDI, but originally was not IDIs.

$$\text{Coverage}_{leaf} = \frac{\text{TP}_{\text{leaf}}}{\text{Total \#IDI}}$$

Coverage_{leaf} metric finds the fraction of all true discriminatory instances that fall within leaf-rule π.

4.4 Mitigation via Guardrail

With the extracted rules we have, we add them beside our DPuTs as guardrails to deny output in those stages. When an input is sent to DpuT, instead of sending it directly for prediction, we at first see if the input falls under any of the extracted discriminatory rules ($\mathcal{R}$). If so we refuse to pass it to our DPuT for a prediction and deny output for those datapoints (Alg. 1, Lines 20–26). The idea is to abstain from making a decision that could lead to discriminatory behavior.

5 Experiments

Benchmarks. We test our framework Remi on both symbolic and data-driven programs.

Symbolic DPuT Benchmarks (PG1–PG6): We evaluate our approach on six symbolic DPuTs (PG1-PG6) where the internal decision logic is known, allowing us to validate root-cause explanations against ground truth. Datasets for PG1, PG3, and PG4 are created following the program listed in Listing 1, following the work of Albarghouthi et al. [8]. For PG2, PG5, PG6, we use the Compas [50] dataset, we inherit the preprocessed version of the data from [39, 73]. Table 1 (left) shows the details. We evaluate our approach using six symbolic programs (PG1–PG6) representing diverse discriminatory scenarios. PG1, PG3, and PG4 model hiring decisions; PG1 [8] involves indirect discrimination via a proxy (college rank), while PG3 and its supposedly repaired variant PG4 [66] exhibit distinct decision logics. PG2 [54] explicitly incorporates a protected attribute (sex) alongside societal bias proxies like arrest records. Finally, PG5 and PG6 [55] utilize additive scoring systems where age and prior arrests influence high-risk classification thresholds (see detailed code and logic in [7]).

Table 1. The dataset and benchmark (symbolic DPuT, left; DNN, right).

DPuT	Dataset	Prot. Feat	#Inst.	#Feat.	Source
Pg1	[8]	ethnicity	30000	4	[8]
Pg2	[39, 50, 73]	sex	6908	4	[54]
Pg3	[8]	ethnicity	30000	4	[66]
Pg4	[8]	ethnicity	30000	4	[66]
Pg5	[39, 50, 73]	age	6908	4	[55]
Pg6	[39, 50, 73]	age	6908	5	[55]

Dataset	Prot. Feat	#Inst.	#Feat.	DNN	Source	#Layers	#Neurons	Acc (%)
Adult Census	Sex Race Age	32,561	13	AC1	[2]	4	45	85.24
				AC2	[2], [24, 60]	3	121	84.70
				AC3	[2]	3	71	84.52
				AC4	[2]	4	221	84.86
				AC5	[2]	4	149	85.19
				AC6	[2]	4	45	84.77
				AC7	[70]	7	145	84.85
				AC8	[43, 61]	4	10	82.15
				AC9	[43, 61]	6	12	81.22
				AC10	[43, 61]	6	20	78.56
				AC11	[43, 61]	6	40	79.25
				AC12	[43, 61]	11	45	81.46
Bank Marketing	Age	45,211	16	BM1	[2]	4	97	89.20
				BM2	[2]	4	65	88.76
				BM3	[2], [24, 60]	3	117	88.22
				BM4	[2]	5	318	89.55
				BM5	[2]	4	49	88.90
				BM6	[2]	4	35	88.94
				BM7	[2]	4	145	88.70
				BM8	[70]	7	141	89.20

Neural DPuT Benchmarks. To assess the generalization of our approach to black-box models, we employ 20 Deep Neural Networks (DNNs) curated by Biswas et al. [11] from literature [24, 43, 60, 61, 70] and Kaggle [2]. These black-box models encompass various feed-forward architectures with ReLU activations. Table 1 (right) shows the characteristics of benchmarks.

- *AC1–AC12:* Trained on the *Adult Census* dataset (32,561 records, 13 attributes) to predict if annual income exceeds $50,000 [22].
- *BM1–BM8:* Trained on the *Bank Marketing* dataset (45,211 entries, 16 features) to predict term deposit subscriptions for a Portuguese bank [47].

Technical Details. Remi was implemented in Python v3.8.20, tensorflow v2.13.0, and scikit-learn v1.3.1 We run all our experiments on an Ubuntu 22.04.5 LTS (jammy) served by an instance type of c5ad.2xlarge (4 cores, 8 vCPUs, 0 GPUs) in Amazon Web Services (AWS) Elastic Computing (EC2). We repeat our experiments 10 times and then report the mean and standard deviation. For RQ5, we run the Themis [9], ExpGA [24], etc., for 60 minutes to find ID instances in the DNN models.

Hyperparameter Selection. For decision trees with CART, following AFT [71], we also set the maximum leaf nodes parameter to 1000 for decision trees. For other models, we follow the same settings.

Common Comparison Metrics: Throughout our experiment section, we use the following metrics. The column labeled *Acc* refers to the accuracy of the model, *Prec* refers to the precision, *Rec* refers to the recall, $F1$ refers to the F1 score for classification, which is a harmonic mean of precision and recall, *ROCAUC* is the Area under the curve (AUC) of the receiver-operating characteristic (ROC) score. The other relevant metrics will described accordingly in the relevant RQs.

Baseline Selection and Scope. We select baselines that represent distinct paradigms in fairness testing and mitigation. **AFT** [71] is the most directly comparable baseline for the *explanation* task, as it also employs a surrogate decision tree to identify discriminatory input regions. AFT iteratively generates random inputs within valid bounds, trains decision trees on the DPuT's classification labels, and extracts path pairs to guide IDI discovery. The key distinction is that AFT trains its trees on the original classification task, whereas Remi trains on relational (fair vs. unfair) labels derived from counterfactual pairs. We compare both tools on their ability to accurately localize discriminatory input regions, using both symbolic and DNN benchmarks.

ExpGA [24] uses explanation-guided genetic algorithms to generate IDIs efficiently in a black-box setting. **LIMI** [63] generates realistic IDIs via latent-space imitation learning. Both tools focus on point-wise IDI discovery and mitigate unfairness through counterfactual data augmentation

and retraining — they do not extract interpretable regional invariants, making them unsuitable as explanation baselines. We therefore compare ExpGA and LIMI against Remi on the *mitigation* task only, using their discovered IDIs as input to Remi's relational pipeline. We exclude Themis, ExpGA, and LIMI from symbolic benchmark comparisons, as these tools are designed for black-box models.

In this research work, we approach the following research questions:

RQ1 Which relational dataset curation technique leads to accurate and precise fairness invariants?
RQ2 Which interpretable algorithms is the best at extracting discriminatory regions?
RQ3 What are the characteristics of extracted fairness invariants?
RQ4 Can we utilize the extracted discriminatory rules as guardrails for mitigating unfairness?
RQ5 Does Remi generalize to address individual discrimination in black-box deep neural networks?

5.1 RQ1. Effectiveness of Relational Dataset Curation

Table 2. Comparison between different data pre-processing training approaches

Prog	Appr	Depth	Acc	Prec	Rec	F1	ROC AUC	T_{Train}	$\#IDI_{Loc}$
Pg1 (SMB)	most-freq	NA	0.92 (± 0.0)	0.0 (± 0.0)	0.0 (± 0.0)	0.0 (± 0.0)	0.5 (± 0.0)	NA	0.0 (± 0.0)
	AFT [71]	11.2 (± 1.14)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.32 (± 0.02)	0.0 (± 0.0)
	DCNE	18.0 (± 0.0)	1.0 (± 0.0)	**0.99** (± 0.0)	0.99 (± 0.0)	0.99 (± 0.0)	1.0 (± 0.0)	0.38 (± 0.01)	**2233.5** (± 1.18)
	DCVE	30.0 (± 0.0)	0.99 (± 0.0)	0.95 (± 0.0)	0.96 (± 0.0)	0.95 (± 0.0)	0.98 (± 0.0)	4.17 (± 0.05)	4329.6 (± 2.32)
	DCHE	17.0 (± 0.0)	1.0 (± 0.0)	**0.99** (± 0.0)	0.99 (± 0.0)	0.99 (± 0.0)	1.0 (± 0.0)	0.37 (± 0.01)	**2231.5** (± 1.65)
Pg2 (SMB)	most-freq	NA	0.97 (± 0.0)	0.0 (± 0.0)	0.0 (± 0.0)	0.0 (± 0.0)	0.5 (± 0.0)	NA	0.0 (± 0.0)
	AFT [71]	4.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.07 (± 0.02)	100.0 (± 0.0)
	DCNE	2.0 (± 0.0)	1.0 (± 0.0)	**1.0** (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.02 (± 0.0)	**206.0** (± 0.0)
	DCVE	2.0 (± 0.0)	1.0 (± 0.0)	**1.0** (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.02 (± 0.0)	**412.0** (± 0.0)
	DCHE	2.0 (± 0.0)	1.0 (± 0.0)	**1.0** (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.02 (± 0.0)	**206.0** (± 0.0)
Pg3 (SMB)	most-freq	NA	0.97 (± 0.0)	0.0 (± 0.0)	0.0 (± 0.0)	0.0 (± 0.0)	0.5 (± 0.0)	NA	0.0 (± 0.0)
	AFT [71]	2.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.25 (± 0.02)	0.0 (± 0.0)
	DCNE	13.0 (± 0.0)	1.0 (± 0.0)	**1.0** (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.07 (± 0.0)	**965.5** (± 1.58)
	DCVE	24.0 (± 0.0)	1.0 (± 0.0)	**0.97** (± 0.0)	0.97 (± 0.0)	0.97 (± 0.0)	0.98 (± 0.0)	0.94 (± 0.02)	1872.4 (± 2.27)
	DCHE	3.0 (± 0.0)	1.0 (± 0.0)	**1.0** (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.05 (± 0.0)	**968.0** (± 0.0)
Pg4 (SMB)	most-freq	NA	0.98 (±0.0)	0.0 (± 0.0)	0.0 (± 0.0)	0.0 (± 0.0)	0.5 (± 0.0)	NA	0 (± 0.0)
	AFT [71]	2.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.25 (± 0.02)	0.0 (± 0.0)
	DCNE	11.0 (± 0.0)	1.0 (± 0.0)	**1.0** (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.06 (± 0.0)	**580.1** (± 1.45)
	DCVE	21.0 (± 0.0)	1.0 (± 0.0)	**0.97** (± 0.0)	0.98 (± 0.0)	0.98 (± 0.0)	0.99 (± 0.0)	0.5 (± 0.01)	1141.8 (± 1.55)
	DCHE	3.0 (± 0.0)	1.0 (± 0.0)	**1.0** (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.04 (± 0.0)	**582.0** (± 0.0)
Pg5 (Scr)	most-freq	NA	0.52 (± 0.0)	0.0 (± 0.0)	0.0 (± 0.0)	0.0 (± 0.0)	0.5 (± 0.0)	NA	0 (± 0.0)
	AFT [71]	26.3 (± 2.0)	0.75 (± 0.01)	**0.77** (± 0.02)	0.76 (± 0.03)	0.77 (± 0.01)	0.75 (± 0.01)	3.85 (± 0.08)	1178.5 (± 152.92)
	DCNE	21.0 (± 0.0)	0.63 (± 0.0)	**0.64** (± 0.0)	0.53 (± 0.0)	0.58 (± 0.0)	0.63 (± 0.0)	2.61 (± 0.02)	**1732.0** (± 0.0)
	DCVE	21.0 (± 0.0)	0.62 (± 0.0)	**0.61** (± 0.0)	0.52 (± 0.0)	0.56 (± 0.0)	0.61 (± 0.0)	3.14 (± 0.03)	3448.2 (± 1.14)
	DCHE	21.0 (± 0.0)	0.63 (± 0.0)	**0.64** (± 0.0)	0.53 (± 0.0)	0.58 (± 0.0)	0.63 (± 0.0)	2.6 (± 0.03)	**1732.0** (± 0.0)
Pg6 (Scr)	most-freq	NA	0.68 (± 0.0)	0.0 (± 0.0)	0.0 (± 0.0)	0.0 (± 0.0)	0.5 (± 0.0)	NA	0 (± 0.0)
	AFT [71]	3.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.08 (± 0.01)	100.0 (± 0.0)
	DCNE	6.0 (± 0.0)	1.0 (± 0.0)	**1.0** (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.03 (± 0.0)	**2216.0** (± 0.0)
	DCVE	5.0 (± 0.0)	1.0 (± 0.0)	**1.0** (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.03 (± 0.0)	**4432.0** (± 0.0)
	DCHE	6.0 (± 0.0)	1.0 (± 0.0)	**1.0** (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.03 (± 0.0)	**2216.0** (± 0.0)

Baseline. We consider two approaches: i) the most frequent labels and ii) AFT [71] as our baseline for localization of discrimination. We choose AFT [71] because it is a state-of-the-art individual fairness testing technique that uses a decision tree as a surrogate to infer the individual discriminatory regions. Specifically, it uses decision tree path pairs to decide if a subspace is promising for generating individual discriminatory instances. The key difference remains that the AFT does not apply a relational input alignment in synthesizing decision trees. Also, for AFT, instead of multiple iterations of training the decision tree to extract paths for generating tests for individual discrimination (IDs), we stop after one, because our proposed approach trains the interpretable decision tree only once with the corresponding data-curation technique

To determine which relational dataset curation approach is most effective in localizing discrimination, we run experiments with the CART [12] (decision tree) interpretable model to compare their efficacy. The results are summarized in Table 2. On the left side of the table, *Prog* lists relevant DPuTs

described in the benchmark. T_{Train} refers to the time (in seconds) taken to complete the training, $\#IDI_{Loc}$ refers to the number of individual discriminatory instances (IDI) localized by the approach. Due to the class imbalance, $Prec$, Rec, $F1$, ROC AUC, and $\#IDI_{Loc}$ are terms that are more important to focus on rather than accuracy. The most important evaluation metric here is $\#ID_{Loc}$ and $Prec$.
From Table 2, we find that our three proposed data-curation techniques localize the largest number of IDIs around 83% of the time, compared to the considered baselines. In more than 80% of cases, we achieve very good performance metrics for accuracy, precision, recall, F1, and ROC for localizing IDI instances. Although in one case (Pg5), we can see AFT [71] performs better than the proposed curation techniques. AFT identifies more IDIs in PG5, but these AFT-identified IDIs do not necessarily overlap with those considered by the other data curation techniques, due to the nature of the AFT algorithm. At each iteration, AFT generates random data within the valid input bounds to find a possible region of interest to generate the IDIs, which is less deterministic and can also generate unrealistic input. In a few cases, AFT has a better F1 score, but it is worth noting that the decision trees of AFT were trained for an actual classification task of a favorable outcome, whereas our trees were trained for *IDI* localization, instead of the actual classification tasks. In most cases, the training approaches with DCNE (no extension) and DCHE (horizontal extension) have superior performance in terms of $\#ID$, accuracy, precision, recall, f1-score, all the time. In terms of the depth of the decision tree, DCHE training needed trees with fewer depths (50% of cases) to localize the discrimination, whereas the other approaches needed trees with higher depths to achieve similar performance.

Answer RQ1: Our proposed approaches of creating a relation dataset via pair alignments and training decision tree inference outperform the baselines in 80% of cases. The horizontal and no-extension alignments are similarly effective at accurately explaining individual discrimination.

5.2 RQ2. Performance of Interpretable Algorithms in Inferring Fairness Invariants

We evaluate nine tree-based algorithms to identify the optimal learner for fairness invariant synthesis: CART [12], GBC [25], XGB [17], C4.5 [51], TaoTree [13], RuleFit [26], FIGS [58], OneR [33], and GreedyRuleList [1]. Table 3 summarizes the training time (T_{Tr}) and the number of localized IDIs ($\#IDI_{Loc}$) across symbolic DPuTs.
For `PG1`, C4.5, CART, and XGB (using DCNE) localized over 2,230 IDIs. While XGB was the fastest, its ensemble nature limits direct interpretability. In `PG3` and `PG4`, all major learners showed comparable performance; however, FIGS and TaoTree provided the shortest training times. RuleFit achieved slightly higher localization in PG3, but at the cost of training latency and precision. In `PG5`, which exhibits low separability due to its probabilistic Bernoulli logic, OneR localized the most IDIs (1,857) but with the lowest precision. FIGS and CART provided a superior balance of precision and localization. For `PG2` and `PG6`, simpler decision logic and smaller datasets led to high performance across all algorithms.
Overall, *C4.5*, *FIGS*, and *CART* emerged as the top performers, outperforming other methods in 83% of cases. While XGB and GBC are computationally efficient, they lack the inherent interpretability required for precise bug localization. Conversely, OneR and GreedyRuleList frequently underperformed, resulting in low precision and F1 scores.

Answer RQ2: C4.5, FIGS, and CART are the most effective for precisely inferring fairness invariants, achieving top-tier performance in 83% of benchmarks.

5.3 RQ3. Various Characteristics of the Relational Explanation Models.

For symbolic programs where the root cause of discrimination is verifiable, we measure how closely Remi's extracted predicates align with the actual discriminatory conditions. Table 4 summarizes these findings using the following metrics:

Table 3. Comparison among different interpretable models with training approach with relational dataset.

Prog	Mod	Acc	Prec	Rec	F1	ROC AUC	T_{Tr}	$\#ID_{Loc}$
Pg1 SMB	CART	1.0 (± 0.0)	**0.99** (± 0.0)	0.99 (± 0.0)	0.99 (± 0.0)	1.0 (± 0.0)	0.38 (± 0.01)	**2233.5** (± 1.18)
	GBC	1.0 (± 0.0)	0.98 (± 0.0)	0.98 (± 0.0)	0.98 (± 0.0)	0.99 (± 0.0)	0.89 (± 0.02)	2196.1 (± 9.52)
	XGBoost	1.0 (± 0.0)	**0.98** (± 0.0)	0.99 (± 0.0)	0.98 (± 0.0)	0.99 (± 0.0)	0.15 (± 0.19)	**2233.0** (± 0.0)
	RuleFit	0.98 (± 0.0)	0.93 (± 0.01)	0.8 (± 0.01)	0.86 (± 0.01)	0.9 (± 0.01)	64.75 (± 1.45)	1807.7 (± 27.51)
	FIGS	0.95 (± 0.01)	0.61 (± 0.06)	0.84 (± 0.05)	0.7 (± 0.03)	0.9 (± 0.02)	0.61 (± 0.01)	1898.8 (± 110.81)
	C4.5Tree	1.0 (± 0.0)	**0.99** (± 0.0)	0.99 (± 0.0)	0.99 (± 0.0)	1.0 (± 0.0)	20.36 (± 1.33)	**2232.0** (± 4.06)
	TaoTree	0.96 (± 0.0)	0.69 (± 0.01)	0.82 (± 0.04)	0.75 (± 0.01)	0.89 (± 0.02)	2.35 (± 0.56)	1839.4 (± 93.1)
	OneR	0.93 (± 0.0)	0.0 (± 0.0)	0.0 (± 0.0)	0.0 (± 0.0)	0.5 (± 0.0)	0.26 (± 0.01)	0.0 (± 0.0)
	GreedyRuleList	0.93 (± 0.0)	0.0 (± 0.0)	0.0 (± 0.0)	0.0 (± 0.0)	0.5 (± 0.0)	0.14 (± 0.0)	0.0 (± 0.0)
Pg2 SMB	CART	1.0 (± 0.0)	**1.0** (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.02 (± 0.0)	**206.0** (± 0.0)
	GBC	1.0 (± 0.0)	**1.0** (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.05 (± 0.0)	**206.0** (± 0.0)
	XGBoost	1.0 (± 0.0)	**1.0** (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.03 (± 0.0)	**206.0** (± 0.0)
	RuleFit	1.0 (± 0.0)	**1.0** (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	4.97 (± 1.38)	**206.0** (± 0.0)
	FIGS	1.0 (± 0.0)	**1.0** (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.04 (± 0.01)	**206.0** (± 0.0)
	C4.5Tree	1.0 (± 0.0)	**1.0**(± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.21 (± 0.02)	**206.0** (± 0.0)
	TaoTree	1.0 (± 0.0)	**1.0** (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.07 (± 0.0)	**206.0** (± 0.0)
	OneR	1.0 (± 0.0)	0.94 (± 0.0)	1.0 (± 0.0)	0.97 (± 0.0)	1.0 (± 0.0)	0.05 (± 0.0)	**206.0** (± 0.0)
	GreedyRuleList	1.0 (± 0.0)	0.94 (± 0.0)	1.0 (± 0.0)	0.97 (± 0.0)	1.0 (± 0.0)	0.03 (± 0.0)	**206.0** (± 0.0)
Pg3 SMB	CART	1.0 (± 0.0)	**1.0** (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.07 (± 0.0)	**965.5** (± 1.58)
	GBC	1.0 (± 0.0)	**0.99** (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.29 (± 0.0)	**967.3** (± 0.67)
	XGBoost	1.0 (± 0.0)	**0.98** (± 0.0)	1.0 (± 0.0)	0.99 (± 0.0)	1.0 (± 0.0)	0.09 (± 0.03)	**967.0** (± 0.0)
	RuleFit	1.0 (± 0.0)	**0.99** (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	37.19 (± 2.15)	**967.6** (± 1.26)
	FIGS	1.0 (± 0.0)	**1.0** (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.54 (± 0.08)	**966.3** (± 1.25)
	C4.5Tree	1.0 (± 0.0)	**1.0** (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	5.61 (± 0.02)	**966.2** (± 1.48)
	TaoTree	1.0 (± 0.0)	**1.0** (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.48 (± 0.02)	**966.5** (± 1.08)
	OneR	0.97 (± 0.0)	0.53 (± 0.0)	1.0 (± 0.0)	0.69 (± 0.0)	0.98 (± 0.0)	0.16 (± 0.0)	**968.0** (± 0.0)
	GreedyRuleList	0.97 (± 0.0)	0.53 (± 0.0)	1.0 (± 0.0)	0.69 (± 0.0)	0.98 (± 0.0)	0.08 (± 0.0)	**968.0** (± 0.0)
Pg4 SMB	CART	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.06 (± 0.0)	**580.1** (± 1.45)
	GBC	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.26 (± 0.01)	**581.3** (± 0.82)
	XGBoost	1.0 (± 0.0)	0.97 (± 0.0)	1.0 (± 0.0)	0.98 (± 0.0)	1.0 (± 0.0)	0.08 (± 0.01)	**580.0** (± 0.0)
	RuleFit	1.0 (± 0.0)	0.99 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	42.1 (± 3.0)	**581.8** (± 0.42)
	FIGS	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.44 (± 0.05)	**580.4** (± 1.17)
	C4.5Tree	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	5.52 (± 0.03)	**580.9** (± 1.29)
	TaoTree	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.4 (± 0.02)	**580.8** (± 1.81)
	OneR	0.98 (± 0.0)	0.0 (± 0.0)	0.0 (± 0.0)	0.0 (± 0.0)	0.5 (± 0.0)	0.16 (± 0.0)	0.0 (± 0.0)
	GreedyRuleList	0.98 (± 0.0)	0.0 (± 0.0)	0.0 (± 0.0)	0.0 (± 0.0)	0.5 (± 0.0)	0.09 (± 0.01)	0.0 (± 0.0)
Pg5 Scr	CART	0.63 (± 0.0)	**0.64** (± 0.0)	0.53 (± 0.0)	0.58 (± 0.0)	0.63 (± 0.0)	2.61 (± 0.02)	**1732.0** (± 0.0)
	GBC	0.6 (± 0.0)	0.61 (± 0.01)	0.47 (± 0.03)	0.53 (± 0.01)	0.6 (± 0.0)	0.18 (± 0.0)	1531.6 (± 87.62)
	XGBoost	0.61 (± 0.0)	0.61 (± 0.0)	0.49 (± 0.0)	0.55 (± 0.0)	0.61 (± 0.0)	0.04 (± 0.01)	1624.0 (± 0.0)
	RuleFit	0.53 (± 0.0)	0.58 (± 0.21)	0.02 (± 0.01)	0.03 (± 0.02)	0.5 (± 0.0)	6.36 (± 0.64)	54.8 (± 43.07)
	FIGS	0.63 (± 0.0)	**0.63** (± 0.01)	0.54 (± 0.03)	0.58 (± 0.01)	0.63 (± 0.0)	109.12 (± 8.8)	**1785.7** (± 113.57)
	C4.5Tree	0.63 (± 0.0)	0.65 (± 0.01)	0.5 (± 0.02)	0.57 (± 0.01)	0.63 (± 0.0)	1.75 (± 0.06)	1646.2 (± 61.05)
	TaoTree	0.63 (± 0.0)	0.65 (± 0.01)	0.49 (± 0.02)	0.56 (± 0.01)	0.63 (± 0.0)	4.83 (± 0.14)	1617.5 (± 71.01)
	OneR	0.55 (± 0.0)	0.52 (± 0.0)	0.57 (± 0.01)	0.54 (± 0.01)	0.55 (± 0.0)	0.07 (± 0.0)	**1857.6** (± 35.63)
	GreedyRuleList	0.55 (± 0.0)	0.52 (± 0.0)	0.52 (± 0.1)	0.52 (± 0.06)	0.54 (± 0.01)	0.04 (± 0.01)	1698.3 (± 317.8)
Pg6 Scr	CART	1.0 (± 0.0)	**1.0** (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.03 (± 0.0)	**2216.0** (± 0.0)
	GBC	1.0 (± 0.0)	**1.0** (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.09 (± 0.0)	**2215.6** (± 0.52)
	XGBoost	1.0 (± 0.0)	**1.0** (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.04 (± 0.01)	**2215.0** (± 0.0)
	RuleFit	1.0 (± 0.0)	**1.0** (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	4.46 (± 0.38)	**2215.7** (± 0.48)
	FIGS	1.0 (± 0.0)	**1.0** (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.14 (± 0.02)	**2215.9** (± 0.32)
	C4.5Tree	1.0 (± 0.0)	**1.0** (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.53 (± 0.03)	**2215.8** (± 0.42)
	TaoTree	1.0 (± 0.0)	**1.0** (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	0.22 (± 0.01)	**2215.7** (± 0.48)
	OneR	0.81 (± 0.0)	0.63 (± 0.0)	1.0 (± 0.0)	0.78 (± 0.0)	0.86 (± 0.0)	0.06 (± 0.0)	**2216.0** (± 0.0)
	GreedyRuleList	0.81 (± 0.0)	0.63 (± 0.0)	1.0 (± 0.0)	0.78 (± 0.0)	0.86 (± 0.0)	0.04 (± 0.0)	**2216.0** (± 0.0)

Table 4. Comparison between the strength of the extracted rules of using proposed relational dataset (RQ3)

Prog	Appr	Depth	*Rule.Len*	Imp	Conf	Tot.Disc.Cov	Tot.$\#IDI_{Loc}$	Bst.Lf.Rule.Len	Bst.Lf.Disc.Cov	Bst.Lf.Conf	Bst.Lf.$\#IDI_{Loc}$
Pg1 SMB	DCNE	18.0 (± 0.0)	11.56 (± 2.77)	0.0 (± 0.0)	0.97 (± 0.09)	**0.98** (± 0.0)	**2233.5** (± 1.18)	16.0 (± 0.0)	0.13 (± 0.0)	1.0 (± 0.0)	**301.0** (± 0.0)
	DCHE	17.0 (± 0.0)	10.67 (± 2.69)	0.0 (± 0.0)	0.98 (± 0.09)	**0.98** (± 0.0)	**2231.5** (± 1.65)	15.0 (± 0.0)	0.13 (± 0.0)	1.0 (± 0.0)	**301.0** (± 0.0)
	AFT [71]	11.0 (± 0.0)	7.23 (± 1.79)	0.0 (± 0.0)	1.0 (± 0.0)	0.0 (± 0.0)	0.0 (± 0.0)	3.4 (± 0.52)	0 (± 0.0)	1.0 (± 0.0)	0 (± 0.0)
Pg2 SMB	DCNE	2.0 (± 0.0)	2.0 (± 0.0)	0.0 (± 0.0)	1.0 (± 0.0)	**1.0** (± 0.0)	**206.0** (± 0.0)	2.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	**206.0** (± 0.0)
	DCHE	2.0 (± 0.0)	2.0 (± 0.0)	0.0 (± 0.0)	1.0 (± 0.0)	**1.0** (± 0.0)	**206.0** (± 0.0)	2.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	**206.0** (± 0.0)
	AFT [71]	4.0 (± 0.0)	2.67 (± 1.53)	0.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	100.0 (± 0.0)	4 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	100.0 (± 0.0)
Pg3 SMB	DCNE	13.0 (± 0.0)	9.4 (± 2.5)	0.0 (± 0.0)	0.99 (± 0.01)	**0.99** (± 0.1)	**965.5** (± 1.58)	5.0 (± 0.0)	0.89 (± 0.0)	1.0 (± 0.0)	863.0 (± 0.0)
	DCHE	**3.0** (± 0.0)	3.0 (± 0.0)	0.0 (± 0.0)	1.0 (± 0.0)	**1.0** (± 0.0)	**968.0** (± 0.0)	3.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	**968.0** (± 0.0)
	AFT [71]	2.0 (± 0.0)	1.5 (± 0.71)	0.0 (± 0.0)	1.0 (± 0.0)	0.0 (± 0.0)	0 (± 0.0)	1 (± 0.0)	0 (± 0.0)	1.0 (± 0.0)	0 (± 0.0)
Pg4 (SMB)	DCNE	11.0 (± 0.0)	8.43 (± 2.07)	0.0 (± 0.0)	1.0 (± 0.01)	**0.99** (± 0.0)	**580.1** (± 1.45)	5.0 (± 0.0)	0.89 (± 0.0)	1.0 (± 0.0)	516.0 (± 0.0)
	DCHE	**3.0** (± 0.0)	3.0 (± 0.0)	0.0 (± 0.0)	1.0 (± 0.0)	**1.0** (± 0.0)	**582.0** (± 0.0)	3.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	**582.0** (± 0.0)
	AFT [71]	2.0 (± 0.0)	1.5 (± 0.71)	0.0 (± 0.0)	1.0 (± 0.0)	0 (± 0.0)	0 (± 0.0)	1 (± 0.0)	0 (± 0.0)	1.0 (± 0.0)	0.0 (± 0.0)
Pg5 (Scr)	DCNE	21.0 (± 0.0)	12.75 (± 3.47)	0.2 (± 0.23)	0.77 (± 0.2)	0.54 (± 0.0)	1732.0 (± 0.0)	9.0 (± 0.0)	0.02 (± 0.0)	0.5 (± 0.0)	63.0 (± 0.0)
	DCHE	21.0 (± 0.0)	12.79 (± 3.49)	0.2 (± 0.23)	0.77 (± 0.2)	0.54 (± 0.0)	1732.0 (± 0.0)	9.0 (± 0.0)	0.02 (± 0.0)	0.5 (± 0.0)	63.0 (± 0.0)
	AFT [71]	25.0 (± 0.0)	13.61 (± 3.7)	0.27 (± 0.41)	0.9 (± 0.15)	**0.9** (± 0.0)	**1843** (± 0.0)	4 (± 0.0)	0.49 (± 0.0)	1.0 (± 0.0)	**999** (± 0.0)
Pg6 (Scr)	DCNE	6.0 (± 0.0)	4.6 (± 1.14)	0.0 (± 0.0)	1.0 (± 0.0)	**1.0** (± 0.0)	**2216.0** (± 0.0)	3.0 (± 0.0)	0.62 (± 0.0)	1.0 (± 0.0)	**1378.0** (± 0.0)
	DCHE	6.0 (± 0.0)	4.6 (± 1.14)	0.0 (± 0.0)	1.0 (± 0.0)	**1.0** (± 0.0)	**2216.0** (± 0.0)	3.0 (± 0.0)	0.62 (± 0.0)	**1.0** (± 0.0)	**1378.0** (± 0.0)
	AFT [71]	3.0 (± 0.0)	2.0 (± 0.0)	0.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	100.0 (± 0.0)	3 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)	100.0 (± 0.0)

- **Rule Length** (*Rul.Len*)**:** Average number of predicates in rules reaching a discriminatory leaf.

- **Impurity (*Imp*):** Average node impurity across all discriminatory leaves.
- **Confidence (*Conf*):** Average precision of rules in identifying IDIs.
- **Coverage (*Tot.Disc.Cov* / *Bst.Lf.Disc.Cov*):** Total IDIs captured by all rules versus the single most representative leaf rule.

In PG1 (Hiring), both DCNE and DCHE trees hit zero impurity and high confidence around 0.97, with *Tot.Disc.Cov* around 0.98 and locates around 2233 IDIs. The single best leaf rule (out of a total 95 rules) covers around 13% of the discriminatory instances considered (301 out of 2235). Average rule lengths are longer, ranging from 10 to 12 literals, indicating that the unsafe region is a narrow area under several tight predicate conditions. AFT [71] does not find any IDIs for PG1 in the current settings, so we do not have any effective rule to compare with. For PG1, our framework discovers a compact discriminatory rule that matches the discrimination encoded in the DPuT, where the outcomes are based on ethnicity. After simplification, the best leaf rule becomes `{ 14.6 < col-rank ≤ 17.25 ∧ 11.99 < y-exp ≤ 14.84 ∧ cf-ethnicity > 10.5 ∧ 17.85< cf-col-rank ≤ 22.95 }` which isolates a narrow region of (col-rank, y-exp) space when the model becomes significantly discriminatory with respect to the protected attribute (ethnicity). The rule recovers the root cause of the discrimination lies in a narrow college rank, mid-experience band (14.6< col-rank ≤ 17.25, 11.99 < y-exp ≤ 14.84) for a favorable ethnicity. Now, flipping the ethnicity to an unfavorable group increases the counterfactual college rank (17.85 < cf-col-rank ≤ 22.95) and flips the hiring decision. The rule explicitly captures the ethnicity to college rank to hire pathway, demonstrating the match with the ground truth level root cause of the discrimination of the DPuT PG1.

For PG2 (Recidivism), Remi achieves 1.0 confidence and 100% coverage (206 IDIs) with a simple two-literal rule: `age ≤ 20.5 ∧ priors-count ≤ 3.5`. This matches the ground truth logic regarding gender-based discrimination in young defendants. AFT [71] localizes only 100 IDIs using a similar predicate structure.

For PG3, Remi achieves 0.99 confidence and 100% coverage (968 IDIs), identifying unfair behavior when `y-exp < 15` and `col-rank` differ between groups. For PG4 (the repaired variant of the hiring algorithm), Remi maintains 0.99 confidence and 100% coverage (582 IDIs), revealing that discriminatory behavior persists within specific `col-rank` and `y-exp` ranges. AFT [71] failed to identify IDIs for either program.

In PG5, Remi achieves 0.77 confidence and 53% total coverage (1,732 instances). The extracted rules correctly identify the age bracket (`20.5 < age ≤ 21.5`) as the primary discriminatory driver. In PG 6, Remi achieves 100% confidence and 100% coverage. A single extracted rule—incorporating `age`, `priors-count`, and `juv-misd-count`—localizes 62% (1,378) of total IDIs. Remi significantly outperforms AFT, which localizes only 100 instances.

Overall, Remi precisely characterizes the root cause of discrimination in 83% of cases. In all cases, except PG5, Remi significantly outperforms the state-of-the-art baseline, AFT [71]. Specifically, in 50% of cases (PG1, PG2, PG4), AFT fails to find any ID instances, rendering its rules inapplicable. This underscores the necessity of relational explanation, as our approach provides greater applicability where standard IDI search methods fail.

Answer RQ3: Remi outperformed the state-of-the-art technique AFT [71] in 83% of cases. In more than 66% of cases, Remi identified all the discrimination regions from the symbolic programs.

5.4 RQ4. Performance of Bias Mitigation via Fairness Invariant Guardrails

A common mitigation approach involves retraining the model with individual discrimination instances (IDIs) so it learns to avoid discrimination in local regions [24, 60, 70, 71]. However, our framework provides rules that define the discriminatory region and its root causes. These rules act as guardrails: when an input falls within a localized discriminatory region, the system can delay inference for human

review. This ensures the model maintains relational consistency and reduces errors in real-time deployment.

Table 5 reports the mitigation results across three program versions, focusing on the number of remaining IDIs (#*IDI*) and the number of non-discriminatory points incorrectly flagged as IDIs (FP_{IDI}). Applying our rules as guardrails reduced #*IDI* in 100% of cases, demonstrating the strong efficacy of the rule-based approach. Performance metrics remained consistently close to those of the original DPuTs. REMI identifies non-discriminatory points as IDIs in a few instances ($FP_{IDI} > 0$). This occurs when discriminatory and non-discriminatory points share similar attributes or reside across a narrow boundary that the decision tree's hyper-rectangular separation struggles to classify.

Table 5. Comparing performance during Mitigation

Prog	Appr	Acc	Prec	Rec	F1	ROC AUC	#*IDI*	#FP_{IDI}
Pg1 SMB	Ini. Pg	1.00	1.00	1.00	1.00	1.00	2250	-
	DT w IDI	0.99	0.99	0.99	0.99	0.99	2430	-
	Ini.Pg + REMI	1.00	1.00	1.00	1.00	1.00	**2**	2670
Pg2 SMB	Init. Pg	1.00	1.00	1.00	1.00	1.00	206	-
	DT w IDI	1.00	1.00	1.00	1.00	1.00	206	-
	Ini.Pg + REMI	1.00	1.00	1.00	1.00	1.00	**0**	**0**
Pg3 SMB	Init. Pg	1.00	1.00	1.00	1.00	1.00	968	-
	DT w IDI	1.00	1.00	1.00	1.00	1.00	968	-
	Ini.Pg + REMI	1.00	1.00	1.00	1.00	1.00	**0**	8
Pg4 SMB	Init. Pg	1.00	1.00	1.00	1.00	1.00	582	-
	DT w IDI	1.00	1.00	1.00	1.00	1.00	582	-
	Ini.Pg + REMI	1.00	1.00	1.00	1.00	1.00	**0**	5
Pg5 SC	Init. Pg	1.00	1.00	1.00	1.00	1.00	3286	-
	DT w IDI	0.68	0.61	0.79	0.69	0.69	2895	-
	Ini.Pg + REMI	1.00	1.00	1.00	1.00	1.00	**1338**	1567
Pg6 SC	Init. Pg	1.00	1.00	1.00	1.00	1.00	2216	-
	DT w IDI	1.00	1.00	1.00	1.00	1.00	2216	-
	Ini.Pg + REMI	1.00	1.00	1.00	1.00	1.00	**0**	32

Answer RQ4: The rule-based guardrails reduced the number of individual discriminatory instances from 2,250 to 2 in one case and from hundreds/thousands of instances to 0 in four cases (out of 6).

5.5 RQ5. Generalization of REMI to Black-Box Neural Networks (DNNs).

Once we establish the precision and usefulness of REMI relative to verifiable symbolic programs, we apply it to black-box deep neural networks to study the generalization. Specifically, we evaluate (a) whether REMI can perform rule inference, and (b) whether the extracted rules reduce discriminatory decisions while preserving the model's original classification utility.

Table 6. DNN rule extraction results using ExpGA [24] as the IDI source under DCNE curation strategy (RQ5).

Model	Depth	*Rule.Len*	*Tot.Rules*	Imp	Conf	Tot.Disc.Cov	Tot.#IDI_{Loc}	Bst.Lf.Rule.Len	Bst.Lf.Disc.Cov	Bst.Lf.Conf	Bst.Lf.#IDI_{Loc}
AC1	35.0 (± 0.0)	18.77 (± 7.08)	449	0.1 (± 0.17)	0.85 (± 0.18)	0.69	11496	4	0.5	1.0	7793
AC2	24.0 (± 0.0)	14.17 (± 3.98)	467	0.12 (± 0.18)	0.85 (± 0.18)	0.77	15583	7	0.43	1.0	8287
AC3	29.0 (± 0.0)	15.54 (± 4.73)	439	0.11 (± 0.18)	0.83 (± 0.19)	0.78	20531	5	0.66	1.0	16291
AC4	24.0 (± 0.0)	14.45 (± 3.91)	436	0.09 (± 0.16)	0.87 (± 0.17)	0.7	13864	2	0.58	1.0	10738
AC5	30.0 (± 0.0)	15.47 (± 4.51)	464	0.11 (± 0.18)	0.85 (± 0.19)	0.76	18223	5	0.64	1.0	14384
AC6	30.0 (± 0.0)	16.35 (± 4.83)	455	0.1 (± 0.17)	0.85 (± 0.19)	0.53	5333	2	0.23	1.0	2215
AC7	25.0 (± 0.0)	14.65 (± 3.98)	469	0.11 (± 0.17)	0.85 (± 0.18)	0.68	8942	2	0.17	0.99	2095
AC8	30.0 (± 0.0)	14.98 (± 5.05)	459	0.1 (± 0.18)	0.84 (± 0.19)	0.6	7231	9	0.34	1.0	3845
AC9	30.0 (± 0.0)	16.96 (± 4.81)	449	0.11 (± 0.18)	0.83 (± 0.19)	0.41	3729	2	0.1	1.0	851
AC10	24.0 (± 0.0)	14.39 (± 3.68)	441	0.11 (± 0.18)	0.84 (± 0.19)	0.84	23704	5	0.7	1.0	18997
AC11	29.0 (± 0.0)	16.17 (± 4.68)	457	0.11 (± 0.17)	0.85 (± 0.17)	0.8	20979	3	0.66	1.0	16368
AC12	33.0 (± 0.0)	17.37 (± 5.64)	443	0.11 (± 0.18)	0.84 (± 0.18)	0.75	14825	2	0.57	1.0	10649

Localization via Rule Extraction for DNNs. Table 6 reports rule extraction results for the 12 Adult Census DNN benchmarks using ExpGA-generated IDIs [24]. REMI produces interpretable trees of moderate depth (24–35), yielding 430–500 rules with average lengths of 14–18 conditions. Extracted rules achieve high average confidence (0.83–0.88), indicating reliable mapping between input features and relational fairness labels. Discrimination coverage ranges from 41–84%, confirming that the rules capture a substantial majority of the discovered discriminatory input space.

Mitigation Results (Qualitative Differences between REMI vs. Baselines): We compare REMI against three baselines: THEMIS [9], EXPGA [24], and LIMI [63]. While EXPGA and LIMI are IDI-generation tools focused on discovering individual discrimination instances, REMI targets explanation and mitigation. REMI operates as a post-hoc wrapper that preserves original model utility, whereas

Table 7. Comparison of mitigation techniques using IDIs generated by black-box DNN fairness testing tools.

Model	Mitigation	Acc	F1	#IDI$_{bef}$	#IDI$_{aft}$	Red. (%)	#FP_{IDI}
AC1	THEMIS [9]	0.85	0.66	7,173	7,571	-5.55	–
	REMI	0.85	0.64	7,173	**4,111**	42.69	1,323
	THEMIS [9] + REMI	0.85	0.66	7,571	**4,370**	42.28	1,208
	EXPGA [24]	0.84	0.61	26,647	15,687	41.13	–
	REMI	0.85	0.64	26,647	**4,234**	84.11	1,187
	EXPGA [24] + REMI	0.84	0.61	15,687	**4,191**	73.28	1,310
	LIMI [63]	0.85	0.65	38,049	13,338	64.95	–
	REMI	0.85	0.64	38,049	**4,294**	87.84	1,901
	LIMI [63]+ REMI	0.85	0.65	13,338	**4,515**	66.15	1,468
AC2	THEMIS [9]	0.84	0.60	7,532	7,891	-4.77	–
	REMI	0.85	0.61	7,532	**4,140**	45.03	1,309
	THEMIS [9] + REMI	0.84	0.60	7,891	**4,236**	46.32	1,259
	EXPGA [24]	0.77	0.65	27,002	19,148	29.09	–
	REMI	0.85	0.61	27,002	**4,101**	84.81	1,377
	EXPGA [24] + REMI	0.77	0.65	19,148	**3,565**	81.38	2,497
	LIMI [63]	0.82	0.42	29,132	16,940	41.85	–
	REMI	0.85	0.61	29,132	**3,838**	86.83	2,179
	LIMI [63]+ REMI	0.82	0.42	16,940	**3,922**	76.85	1,999
AC3	THEMIS [9]	0.85	0.64	7,525	7,094	5.73	–
	REMI	0.85	0.66	7,525	**4,290**	42.99	1,244
	THEMIS [9] + REMI	0.85	0.64	7,094	**4,754**	32.99	988
	EXPGA [24]	0.84	0.62	20,481	24,652	-20.37	–
	REMI	0.85	0.66	20,481	**4,235**	79.32	1,317
	EXPGA [24] + REMI	0.84	0.62	24,652	**4,121**	83.28	1,495
	LIMI [63]	0.85	0.64	46,081	11,014	76.10	–
	REMI	0.85	0.66	46,081	**4,233**	90.81	2,478
	LIMI [63]+ REMI	0.85	0.64	11,014	**4,120**	62.59	1,749
AC4	THEMIS [9]	0.84	0.65	7,491	7,999	-6.78	–
	REMI	0.85	0.62	7,491	**3,957**	47.18	1,512
	THEMIS [9] + REMI	0.84	0.65	7,999	**4,151**	48.11	1,363
	EXPGA [24]	0.84	0.66	21,503	18,465	14.13	–
	REMI	0.85	0.62	21,503	**4,094**	80.96	1,420
	EXPGA [24] + REMI	0.84	0.66	18,465	**4,601**	75.08	1,096
	LIMI [63]	0.83	0.67	38,043	34,463	9.41	–
	REMI	0.85	0.62	38,043	**3,683**	90.32	2,480
	LIMI [63]+ REMI	0.83	0.67	34,463	**4,142**	87.98	3,008
AC5	THEMIS [9]	0.84	0.65	7,206	7,202	0.06	–
	REMI	0.85	0.66	7,206	**4,054**	43.74	1,358
	THEMIS [9] + REMI	0.84	0.65	7,202	**4,331**	39.86	1,229
	EXPGA [24]	0.83	0.62	21,798	22,543	-3.42	–
	REMI	0.85	0.66	21,798	**4,293**	80.31	1,140
	EXPGA [24] + REMI	0.83	0.62	22,543	**4,320**	80.84	1,518
	LIMI [63]	0.82	0.66	41,114	21,810	46.95	–
	REMI	0.85	0.66	41,114	**4,066**	90.11	2,433
	LIMI [63]+ REMI	0.82	0.66	21,810	**4,244**	80.54	2,351
AC6	THEMIS [9]	0.84	0.67	7,559	10,522	-39.20	–
	REMI	0.85	0.61	7,559	**3,851**	49.05	1,666
	THEMIS [9] + REMI	0.84	0.67	10,522	**3,042**	71.09	2,495
	EXPGA [24]	0.85	0.63	25,959	9,502	63.40	–
	REMI	0.85	0.61	25,959	**3,803**	85.35	1,665
	EXPGA [24] + REMI	0.85	0.63	9,502	**4,169**	56.13	1,249
	LIMI [63]	0.85	0.67	34,822	15,302	56.06	–
	REMI	0.85	0.61	34,822	**3,854**	88.93	2,206
	LIMI [63]+ REMI	0.85	0.67	15,302	**4,338**	71.65	1,854
BM1	THEMIS [9]	0.88	0.00	3,136	3,788	-20.79	–
	REMI	0.89	0.48	3,136	**1,076**	65.69	479
	THEMIS [9] + REMI	0.88	0.00	3,788	**1,161**	69.35	499
BM2	THEMIS [9]	0.88	0.00	3,291	3,638	-10.54	–
	REMI	0.89	0.54	3,291	**1,063**	67.70	489
	THEMIS [9] + REMI	0.88	0.00	3,638	**1,073**	70.51	512
BM3	THEMIS [9]	0.88	0.00	3,506	3,573	-1.91	–
	REMI	0.88	0.58	3,506	**1,082**	69.14	513
	THEMIS [9] + REMI	0.88	0.00	3,573	**1,076**	69.89	513
BM4	THEMIS [9]	0.23	0.23	3,121	3,445	-10.38	–
	REMI	0.90	0.53	3,121	**1,060**	66.04	470
	THEMIS [9] + REMI	0.23	0.23	3,445	**1,034**	69.99	486
AC7	THEMIS [9]	0.84	0.68	7,445	8,140	-9.34	–
	REMI	0.85	0.64	7,445	**4,058**	45.49	1,437
	THEMIS [9] + REMI	0.84	0.68	8,140	**3,996**	50.91	1,826
	EXPGA [24]	0.76	0.00	23,326	12,311	47.22	–
	REMI	0.85	0.64	23,326	**4,105**	82.40	1,425
	EXPGA [24] + REMI	0.76	0.00	12,311	**3,369**	72.63	1,842
	LIMI [63]	0.76	0.00	37,705	11,208	70.27	–
	REMI	0.85	0.64	37,705	**4,001**	89.39	2,419
	LIMI [63]+ REMI	0.76	0.00	11,208	**3,094**	72.39	2,112
AC8	THEMIS [9]	0.84	0.56	7,566	7,689	-1.63	–
	REMI	0.83	0.66	7,566	**4,302**	43.14	1,277
	THEMIS [9] + REMI	0.84	0.56	7,689	**3,927**	48.93	1,645
	EXPGA [24]	0.84	0.64	20,053	11,306	43.62	–
	REMI	0.83	0.66	20,053	**4,411**	78.00	1,093
	EXPGA [24] + REMI	0.84	0.64	11,306	**4,075**	63.96	1,451
	LIMI [63]	0.85	0.67	24,386	17,034	30.15	–
	REMI	0.83	0.66	24,386	**4,188**	82.83	2,346
	LIMI [63]+ REMI	0.85	0.67	17,034	**4,362**	74.39	1,733
AC9	THEMIS [9]	0.84	0.68	8,348	8,260	1.05	–
	REMI	0.83	0.56	8,348	**3,642**	56.37	1,927
	THEMIS [9] + REMI	0.84	0.68	8,260	**3,691**	55.31	1,980
	EXPGA [24]	0.84	0.65	20,006	8,120	59.41	–
	REMI	0.83	0.56	20,006	**3,776**	81.13	1,727
	EXPGA [24] + REMI	0.84	0.65	8,120	**4,391**	45.92	1,168
	LIMI [63]	0.83	0.58	51,399	22,330	56.56	–
	REMI	0.83	0.56	51,399	**3,696**	92.81	2,379
	LIMI [63]+ REMI	0.83	0.58	22,330	**4,160**	81.37	2,054
AC10	THEMIS [9]	0.85	0.63	7,509	7,250	3.45	–
	REMI	0.78	0.33	7,509	**4,357**	41.98	1,164
	THEMIS [9] + REMI	0.85	0.63	7,250	**4,129**	43.05	1,339
	EXPGA [24]	0.84	0.57	28,342	27,314	3.63	–
	REMI	0.85	0.66	28,342	**4,220**	85.11	1,218
	EXPGA [24] + REMI	0.84	0.57	27,314	**3,610**	86.78	1,803
	LIMI [63]	0.83	0.65	42,942	21,551	49.81	–
	REMI	0.85	0.66	42,942	**4,141**	90.36	2,175
	LIMI [63]+ REMI	0.83	0.65	21,551	**4,512**	79.06	1,932
AC11	THEMIS [9]	0.84	0.62	7,808	7,481	4.19	–
	REMI	0.81	0.67	7,808	**4,529**	42.00	1,171
	THEMIS [9] + REMI	0.84	0.62	7,481	**3,957**	47.11	1,596
	EXPGA [24]	0.80	0.63	16,445	24,718	-50.31	–
	REMI	0.81	0.67	16,445	**4,746**	71.14	931
	EXPGA [24] + REMI	0.80	0.63	24,718	**3,739**	84.87	1,844
	LIMI [63]	0.76	0.00	65,598	11,209	82.91	–
	REMI	0.81	0.67	65,598	**3,677**	94.39	4,723
	LIMI [63]+ REMI	0.76	0.00	11,209	**3,264**	70.88	1,968
AC12	THEMIS [9]	0.76	0.00	7,667	11,208	-46.18	–
	REMI	0.84	0.65	7,667	**3,972**	48.19	1,580
	THEMIS [9] + REMI	0.76	0.00	11,208	**3,092**	72.41	2,115
	EXPGA [24]	0.83	0.61	23,401	18,645	20.32	–
	REMI	0.84	0.65	23,401	**4,209**	82.01	1,432
	EXPGA [24] + REMI	0.83	0.61	18,645	**3,820**	79.51	1,642
	LIMI [63]	0.76	0.00	74,169	11,208	84.89	–
	REMI	0.84	0.65	74,169	**3,927**	94.71	2,898
	LIMI [63]+ REMI	0.76	0.00	11,208	**3,086**	72.47	2,120
BM5	THEMIS [9]	0.15	0.21	3,288	3,862	-17.46	–
	REMI	0.89	0.58	3,288	**1,028**	68.73	582
	THEMIS [9] + REMI	0.15	0.21	3,862	**908**	76.49	578
BM6	THEMIS [9]	0.87	0.01	3,175	3,862	-21.64	–
	REMI	0.89	0.53	3,175	**1,232**	61.20	361
	THEMIS [9] + REMI	0.87	0.01	3,862	**1,032**	73.28	572
BM7	THEMIS [9]	0.57	0.27	3,244	3,692	-13.81	–
	REMI	0.89	0.56	3,244	**1,074**	66.89	471
	THEMIS [9] + REMI	0.57	0.27	3,692	**1,058**	71.34	516
BM8	THEMIS [9]	0.81	0.09	3,244	3,859	-18.96	–
	REMI	0.89	0.43	3,244	**1,074**	66.89	471
	THEMIS [9] + REMI	0.81	0.09	3,859	**1,006**	73.93	528

counterfactual retraining modifies the model itself and risks degrading accuracy or collapsing when discovered IDIs are out of distribution. Crucially, REMI is agnostic to the upstream IDI-finding method: it takes IDIs discovered by THEMIS, EXPGA, or LIMI, aligns them into relational datasets, infers interpretable discriminatory regions, and deploys those regions as guardrails.

Table 7 reports standard classification utility metrics (Accuracy, F1) and four fairness-specific columns: #IDI_{bef}, the number of IDIs before mitigation; #IDI_{aft}, the number of IDIs after mitigation; *Red* (%), the percentage reduction between the two; and #FP_{IDI}, the number of falsely rejected instances by the guardrail.

- *Data Augmentations vs. REMI.* Across all AC models, REMI consistently reduces unfair outputs. REMI reduces IDIs by 42–56% on the Adult Census models for THEMIS IDIs, by 71–85% for

ExpGA-based IDIs, and by 87–95% for LIMI-based IDIs. The larger reductions for ExpGA and LIMI stem from the greater volume (often 2-10x more vs Themis) of initial IDIs these tools discover (16,000–74,000 vs. Themis's 7,000–8,000), which provides richer relational training data helping to create more precise rules. On the BM models, reductions are around 61–69%. For BM1, we get around 65.7% reduction from 3,136 to 1,076. For BM3, the reduction is from 3506 to 1082, or around 69.1%.

- $\#FP_{IDI}$ *(false-positive guardrails)*. Remi incurs the cost of selectively denying some benign inputs. For AC models, $\#FP_{IDI}$ typically ranges from 1,100–3,000 (e.g., 1,323 for AC1; 1,927 for AC9), but each denial is offset by 2.2–2.6× as many unfair cases prevented. For instance, AC1 prevents 3,062 unfair outputs against 1,323 benign blocks.
- *Data Augmentations with Guardrails*. When Remi applied after retraining with the baseline techniques (*tool* + Remi), reliably lowers *#IDI* well below the unguarded retrained (*tool*) models by 39-84% (with Themis AC3: reduced from 7094 to 4754, for AC6 reduced from 10522 to 3042, for AC7 reduced from 8140 to 3996). Also, when CF retraining reduce IDIs e.g., AC1 with ExpGA reduces from 26,647 to 15,687), adding Remi guardrails further reduces IDIs to 4,191 (73% beyond retrained ExpGA alone).

Answer RQ5: Remi generalizes to improve individual fairness for black-box deep neural networks. The full pipeline of data curation, rule extraction, and guardrail deployment produces interpretable fairness invariants with high confidence (0.83–0.88) and substantial discrimination coverage (41–84%) on DNNs. Remi consistently reduces individual discriminatory instances by at least 40% and up to 84%, significantly outperforming the baseline mitigation techniques.

6 Discussions

Runtime and Performance Overhead. For symbolic programs, the full Remi pipeline (IDI finding, data curation, interpretable model training, rule extraction, and guardrail application) completes in under 120 seconds per program. For DNN benchmarks, Stage 1 IDI finding is the primary overhead, capped at 60 minutes (3,600 seconds) per model for all evaluated tools. The remaining pipeline stages are completed in under 30 seconds per model.

Component-Wise Ablation Analysis. Although Remi's evaluation does not use a single ablation table, each RQ is structured to isolate the contribution of one pipeline component while holding others fixed. For example, RQ1 ablates *data curation*: fixing the learner (CART) and varying the alignment strategy (DCNE, DCVE, DCHE vs. AFT and most-frequent baselines) to measure the contribution of relational alignment to localization quality. Similarly, RQ2 ablates the *interpretable learner*: fixing the curation method and comparing tree-based algorithms to isolate the impact of learner choice.

Limitation. Remi can be integrated into existing ML pipelines as a post-processing wrapper requiring no modification to the underlying model. After the DPuT produces a prediction, the guardrail module checks whether the input falls within a known discriminatory region and, if so, defers the decision to human review, making Remi compatible with any classifier. However, several limitations bound the current scope of applicability. First, Remi should be viewed as an auditing and guardrail-synthesis tool rather than a proof of global fairness: its rules describe unfair regions represented in the sampled relational dataset, and repeated audits may be needed as the DPuT or data distribution evolves. Second, rule quality is bounded by the coverage of the IDI-finding stage; input regions not represented during training—whether fair or discriminatory—will not be captured by the extracted rules. If the IDI-finding stage discovers no IDIs, Remi cannot proceed, though this outcome does not itself guarantee model fairness. Third, for natural language or vision systems, protected-attribute changes can affect semantics in subtle ways; applying Remi in such settings would require domain-specific

counterfactual generators and semantic-validity filters. We therefore do not claim that the current implementation directly addresses fairness debugging for all large-scale systems.

Threats to Validity. To ensure reproducibility and deterministic results, we repeat experiments multiple times, reporting averages and standard deviations. To mitigate the impact of randomness in fairness testing and decision tree initialization, we employ seed values, acknowledging the resulting seed dependency. Regarding generalizability, we evaluate our framework on 20 DNNs and diverse symbolic programs, including logic-based and scoring-based models. Internal validity concerns related to hyperparameter selection for IDI searching and interpretable model training are addressed by following established standards. Finally, to prevent variability from sampling symbolic datasets, we generate and fix our datasets once to maintain consistency across all experimental runs.

Validity of Counterfactual Inputs. Counterfactual generation in Remi follows a two-step process to ensure semantic validity. First, the protected attribute is flipped within its valid domain (e.g., sex, ethnicity). Second, any non-protected attributes causally dependent on the protected attribute are updated according to domain-specific symbolic rules. For symbolic DPuTs, these constraints are derived directly from the benchmark program logic. For DNN benchmarks, we inherit input domains and preprocessing constraints from the corresponding dataset (e.g., flipping sex from male to female triggers an update of the relationship attribute). However, for DNN benchmarks, a potential limitation is that causal dependencies between features may not be fully known, meaning some generated counterfactuals may be invalid due to the limitations of preprocessing rules.

Validity of Explanations.: Remi Explanations are quantitatively validated in different parts of the paper. In RQ3 and RQ5, we measure impurity, confidence, and coverage of the extracted rules against IDI localization and coverage metrics. Qualitative correctness is confirmed by benchmarking extracted invariants against ground-truth symbolic logic. In RQ4–RQ5, we demonstrate that applying these rules as guardrails effectively reduces IDIs, which would not be possible if the explanations were incorrect. A limitation is that for DNNs, where the ground-truth discriminatory logic is unknown, we can only validate explanations indirectly through their mitigation effectiveness.

Interpretability in Symbolic Setting. Interpretability is not for identifying protected attributes, but for mapping discriminatory logic to regions of non-protected features. This enables generalization from point-wise violations to systematic patterns, which is critical for deploying automated mitigation guardrails, especially in a black-box setting. Besides, interpretability in a symbolic setting serves as a validation step to assess whether our approach accurately explains the ground-truth discrimination.

Concrete Examples of Fairness Violations, Rules, and Mitigation. To illustrate Remi in a realistic fairness-debugging workflow, we consider a COMPAS-like recidivism risk assessment tool, which has been widely documented to produce racially and sexually disparate predictions [34]. Our symbolic benchmark PG2 encodes a simplified recidivism scoring rule (PG2 Listing in [7] from [54]).

- *Stage 1: IDI Discovery.* From 6,908 individuals sampled from the preprocessed COMPAS dataset [39, 50, 73], Remi find 206 IDIs by flipping `sex` (male ↔ female).
- *Stage 2: Relational Data Curation.* Using DCNE, Remi constructs a relational dataset of 6,908 instances, each labeled fair (+) or discriminatory (−) based on relational data points.
- *Stage 3: Rule Extraction.* A CART decision tree trained on the relational labels extracts the rule $\{\texttt{age} \leq 20.5 \wedge \texttt{priors\_count} \leq 3.5\}$, which captures all 206 IDIs with high confidence.
- *Stage 4: Guardrail Mitigation.* Deploying this rule as a guardrail eliminates all 206 IDIs, achieving a full reduction in discriminatory behavior without changing the logic of the decision-making program.

7 Related Work

Fairness Testing: When Galhotra et al. [27] proposed and popularized the causal fairness definition for individual fairness testing, a substantial line of work [4, 5, 9, 24, 45] focused on discovering discriminatory inputs (IDIs) through testing at scale following that fairness definition. Some of the tools e.g., Themis [9], AEQUITAS [60], SG [5], ExpGA [24], LIMI [63], AFT [71], considered black-box settings, where the inner working knowledge of classifiers was not necessary for finding ID instances. Some other test generation algorithms ADF [70], EIDIG [68], NeuronFair [72], DICE [46], MAFT [62] considers a white-box setting, which requires knowledge of the classifier being tested Many of these tools can be used in the first stage of our framework to find IDIs for DPuTs under test if the DPuT matches the target model for these tools. The problem of testing for group fairness has been explored extensively in existing literature [10, 14, 15, 18, 59, 67].
Formal Methods. Formal tools and techniques have been significantly studied in the literature [6, 36, 37, 41, 44] The FairSquare tool [8] targets probabilistic programs, employing volume-based computations to verify their fairness properties. A formal methodology for certifying individual fairness in standard machine learning architectures was established by John et al. [30]. Fairify [11] examines individual fairness and its variants in neural networks by translating pre-trained models into Satisfiability Modulo Theories (SMT) problems.
Explainable-AI and Interpretability: Riberio et al.[52] provide a local explanation model, LIME, to explain a locally faithful non-linear model via a sparse linear model Lundberg. et al., [42] propose a unified approach with SHAP and SHAPly values to interpret the global behavior of the model. Mothilal et al. [48] provide a tool called DiCE, which utilizes diverse counterfactual explanations to understand the decision boundary of the model, more specifically, changes in which feature will lead to the flip of the decision from one side of the boundary to the other side. FairLay-ML [64] is a debugging tool to explain the fairness implications of data-driven software. Parfait-ML [53, 59] explores the explanation behind hyperparameter configuration, which might lead to unfair models via decision trees. Instead, we used interpretable models to explain the root cause of unfairness.
Bias Mitigation To address bias in machine learning outputs, several researchers have proposed different types of mitigation algorithms [3, 35, 65]. Our work is closer to post-processing techniques [20, 21, 28, 40, 57, 69]. While these works aim to modify the decision logic of black-box models, our approach leverages guardrails, derived from the explanation models, to improve the fairness of symbolic and data-driven black-box software.

8 Conclusion and Future Work

In this paper, we presented Remi, a novel framework that treats individual fairness as a relational invariant discovery problem. By transforming counterfactual pairs into a relational dataset, Remi successfully localizes discriminatory regions using interpretable, rule-based explainers. These extracted fairness invariants provide both a precise explanation of the root cause of discrimination and a robust mitigation strategy via real-time guardrails. Our evaluation across symbolic, scoring, and deep neural network programs demonstrates that Remi is highly effective and outperforms the state-of-the-art baselines. Future work includes extending this relational framework to unstructured domains, such as natural language [19].

9 Data Availability

Our open-source tool Remi with all experimental subjects is available at Figshare [7] and Github.

Acknowledgments

This project has been partially supported by NSF under grants CCF-2536640 and CNS-2230061.

References

[1] 2024. Imodels Greedy Rule List. https://csinva.io/imodels/rule_list/greedy_rule_list.html. online.

[2] 2024. Kaggle. https://www.kaggle.com. online.

[3] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudik, John Langford, and Hanna Wallach. 2018. A Reductions Approach to Fair Classification. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 60–69. https://proceedings.mlr.press/v80/agarwal18a.html

[4] Aniya Agarwal, Pranay Lohia, Seema Nagar, Kuntal Dey, and Diptikalyan Saha. 2018. Automated Test Generation to Detect Individual Discrimination in AI Models. *arXiv e-prints*, Article arXiv:1809.03260 (Sept. 2018), arXiv:1809.03260 pages. arXiv:1809.03260 [cs.AI] doi:10.48550/arXiv.1809.03260

[5] Aniya Aggarwal, Pranay Lohia, Seema Nagar, Kuntal Dey, and Diptikalyan Saha. 2019. Black Box Fairness Testing of Machine Learning Models. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2019)*. 625–635. doi:10.1145/3338906.3338937

[6] Ranit D. Akash, Ashish Kumar, Verya Monjezi, Ashutosh Trivedi, Gang Tan, and Saeid Tizpaz-Niari. 2025. Uncovering Discrimination Clusters: Quantifying and Explaining Systematic Fairness Violations. In *2025 40th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 1680–1692. doi:10.1109/ASE63991.2025.00141

[7] Ranit Debnath Akash, Ashish Kumar, Gang Tan, and Saeid Tizpaz Niari. 2026. Supplementary Materials of paper "Fairness Invariants: A Relational Approach to Explaining and Mitigating Fairness Bugs". doi:10.6084/m9.figshare.33061115.v5 DOI: https://doi.org/10.6084/m9.figshare.33061115.v5.

[8] Aws Albarghouthi, Loris D'Antoni, Samuel Drews, and Aditya V. Nori. 2017. FairSquare: probabilistic verification of program fairness. *Proc. ACM Program. Lang.* 1, OOPSLA, Article 80 (Oct. 2017), 30 pages. doi:10.1145/3133904

[9] Rico Angell, Brittany Johnson, Yuriy Brun, and Alexandra Meliou. 2018. Themis: automatically testing software for discrimination. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Lake Buena Vista, FL, USA) *(ESEC/FSE 2018)*. Association for Computing Machinery, New York, NY, USA, 871–875. doi:10.1145/3236024.3264590

[10] Rachel K. E. Bellamy, Kuntal Dey, Michael Hind, Samuel C. Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, Aleksandra Mojsilovic, Seema Nagar, Karthikeyan Natesan Ramamurthy, John Richards, Diptikalyan Saha, Prasanna Sattigeri, Moninder Singh, Kush R. Varshney, and Yunfeng Zhang. 2018. AI Fairness 360: An Extensible Toolkit for Detecting, Understanding, and Mitigating Unwanted Algorithmic Bias. *arXiv e-prints*, Article arXiv:1810.01943 (Oct. 2018), arXiv:1810.01943 pages. arXiv:1810.01943 [cs.AI] doi:10.48550/arXiv.1810.01943

[11] Sumon Biswas and Hridesh Rajan. 2023. Fairify: Fairness Verification of Neural Networks. In *Proceedings of the 45th International Conference on Software Engineering* (Melbourne, Victoria, Australia) *(ICSE '23)*. IEEE Press, 1546–1558. doi:10.1109/ICSE48619.2023.00134

[12] L. Breiman, Jerome H. Friedman, Richard A. Olshen, and C. J. Stone. 2000. Classification and Regression Trees. https://api.semanticscholar.org/CorpusID:29458883

[13] Miguel A. Carreira-Perpinan and Pooya Tavallali. 2018. Alternating optimization of decision trees, with application to learning sparse oblique trees. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2018/file/185c29dc24325934ee377cfda20e414c-Paper.pdf

[14] Joymallya Chakraborty, Suvodeep Majumder, and Tim Menzies. 2021. Bias in Machine Learning Software: Why? How? What to Do?. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2021)*. Association for Computing Machinery, New York, NY, USA, 429–440. doi:10.1145/3468264.3468537

[15] Joymallya Chakraborty, Suvodeep Majumder, Zhe Yu, and Tim Menzies. 2020. Fairway: a way to build fair ML software. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Virtual Event, USA) *(ESEC/FSE 2020)*. Association for Computing Machinery, New York, NY, USA, 654–665. doi:10.1145/3368089.3409697

[16] Joymallya Chakraborty, Tianpei Xia, Fahmid M. Fahid, and Tim Menzies. 2019. Software Engineering for Fairness: A Case Study with Hyperparameter Optimization. *arXiv e-prints*, Article arXiv:1905.05786 (May 2019), arXiv:1905.05786 pages. arXiv:1905.05786 [cs.SE] doi:10.48550/arXiv.1905.05786

[17] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. ACM, New York, NY, USA, 785–794. doi:10.1145/2939672.2939785

[18] Zhenpeng Chen, Jie M. Zhang, Federica Sarro, and Mark Harman. 2022. MAAT: A Novel Ensemble Approach to Addressing Fairness and Performance Bugs for Machine Learning Software *(ESEC/FSE 2022)*. 1122–1134. doi:10.1145/

3540250.3549093

[19] Rajashree Dahal, Pardis Hossein Pour, Pranathi Kamisetty, Satwik Pamulaparthy, Saeid Tizpaz-Niari, and Natalie Parde. 2026. Investigating Stigmatizing Language in Clinical Documentation with Open-Source Large Language Models. In *BioNLP 2026*, Dina Demner-Fushman, Sophia Ananiadou, Kirk Roberts, and Junichi Tsujii (Eds.). Association for Computational Linguistics, San Diego, California, 490–501. doi:10.18653/v1/2026.bionlp-1.39

[20] Vishnu Asutosh Dasu, Ashish Kumar, Saeid Tizpaz-Niari, and Gang Tan. 2024. NeuFair: Neural Network Fairness Repair with Dropout. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis* (Vienna, Austria) *(ISSTA 2024)*. Association for Computing Machinery, New York, NY, USA, 1541–1553. doi:10.1145/3650212.3680380

[21] Vishnu Asutosh Dasu, Md Rafi Ur Rashid, Vipul Gupta Saeid Tizpaz-Niari, and Gang Tan. 2026. Attention Pruning: Automated Fairness Repair of Language Models via Surrogate Simulated Annealing. In *48th International Conference on Software Engineering (ICSE)*. To appear.

[22] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/census+income

[23] 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* (Cambridge, Massachusetts) *(ITCS '12)*. Association for Computing Machinery, New York, NY, USA, 214–226. doi:10.1145/2090236.2090255

[24] Ming Fan, Wenying Wei, Wuxia Jin, Zijiang Yang, and Ting Liu. 2022. Explanation-guided fairness testing through genetic algorithm. In *Proceedings of the 44th International Conference on Software Engineering* (Pittsburgh, Pennsylvania) *(ICSE '22)*. Association for Computing Machinery, New York, NY, USA, 871–882. doi:10.1145/3510003.3510137

[25] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232. doi:10.1214/aos/1013203451

[26] Jerome H. Friedman and Bogdan E. Popescu. 2008. Predictive Learning via Rule Ensembles. *The Annals of Applied Statistics* 2, 3 (2008), 916–954. http://www.jstor.org/stable/30245114

[27] Sainyam Galhotra, Yuriy Brun, and Alexandra Meliou. 2017. Fairness testing: testing software for discrimination. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering* (Paderborn, Germany) *(ESEC/FSE 2017)*. Association for Computing Machinery, New York, NY, USA, 498–510. doi:10.1145/3106237.3106277

[28] Xuanqi Gao, Juan Zhai, Shiqing Ma, Chao Shen, Yufei Chen, and Qian Wang. 2022. FairNeuron: improving deep neural network fairness with adversary games on selective neurons. In *Proceedings of the 44th International Conference on Software Engineering* (Pittsburgh, Pennsylvania) *(ICSE '22)*. Association for Computing Machinery, New York, NY, USA, 921–933. doi:10.1145/3510003.3510087

[29] Pranav Garg, Daniel Neider, P. Madhusudan, and Dan Roth. 2016. Learning invariants using decision trees and implication counterexamples. *SIGPLAN Not.* 51, 1 (Jan. 2016), 499–512. doi:10.1145/2914770.2837664

[30] Philips George John, Deepak Vijaykeerthy, and Diptikalyan Saha. 2020. Verifying Individual Fairness in Machine Learning Models. In *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI) (Proceedings of Machine Learning Research, Vol. 124)*, Jonas Peters and David Sontag (Eds.). PMLR, 749–758. https://proceedings.mlr.press/v124/george-john20a.html

[31] Usman Gohar, Sumon Biswas, and Hridesh Rajan. 2023. Towards Understanding Fairness and its Composition in Ensemble Machine Learning. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. 1533–1545. doi:10.1109/ICSE48619.2023.00133

[32] 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* (Barcelona, Spain) *(NIPS'16)*. 3323–3331.

[33] Robert C. Holte. 1993. Very Simple Classification Rules Perform Well on Most Commonly Used Datasets. *Machine Learning* 11 (1993), 63–90. https://api.semanticscholar.org/CorpusID:6596

[34] Surya Mattu Julia Angwin, Jeff Larson and Lauren Kirchne. 2021. Machine Bias. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing. Online.

[35] Faisal Kamiran, Asim Karim, and Xiangliang Zhang. 2012. Decision Theory for Discrimination-Aware Classification. In *2012 IEEE 12th International Conference on Data Mining*. 924–929. doi:10.1109/ICDM.2012.45

[36] Brian Hyeongseok Kim, Jacqueline L. Mitchell, and Chao Wang. 2026. Analyzing Fairness of Neural Network Prediction via Counterfactual Dataset Generation. arXiv:2602.10457 [cs.LG] https://arxiv.org/abs/2602.10457

[37] Brian Hyeongseok Kim, Jingbo Wang, and Chao Wang. 2025. FairQuant: Certifying and Quantifying Fairness of Deep Neural Networks. In *Proceedings of the IEEE/ACM 47th International Conference on Software Engineering* (Ottawa, Ontario, Canada) *(ICSE '25)*. IEEE Press, 527–539. doi:10.1109/ICSE55347.2025.00016

[38] Matt Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. 2017. Counterfactual fairness. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 4069–4079.

[39] Jeff Larson, Surya Mattu, Lauren Kirchner, and Julia Angwin. 2016. How we analyzed the COMPAS recidivism algorithm. *ProPublica (5 2016)* 9, 1 (2016), 3–3.

[40] Tianlin Li, Xiaofei Xie, Jian Wang, Qing Guo, Aishan Liu, Lei Ma, and Yang Liu. 2023. Faire: Repairing Fairness of Neural Networks via Neuron Condition Synthesis. *ACM Trans. Softw. Eng. Methodol.* 33, 1, Article 21 (Nov. 2023), 24 pages. doi:10.1145/3617168

[41] Yannan Li, Jingbo Wang, and Chao Wang. 2023. Certifying the Fairness of KNN in the Presence of Dataset Bias. In *Computer Aided Verification: 35th International Conference, CAV 2023, Paris, France, July 17–22, 2023, Proceedings, Part II* (Paris, France). Springer-Verlag, Berlin, Heidelberg, 335–357. doi:10.1007/978-3-031-37703-7_16

[42] Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 4768–4777.

[43] Denis Mazzucato and Caterina Urban. 2021. Reduced Products of Abstract Domains for Fairness Certification of Neural Networks. In *Static Analysis: 28th International Symposium, SAS 2021, Chicago, IL, USA, October 17–19, 2021, Proceedings* (Chicago, IL, USA). Springer-Verlag, Berlin, Heidelberg, 308–322. doi:10.1007/978-3-030-88806-0_15

[44] Verya Monjezi, Ashish Kumar, Ashutosh Trivedi, Gang Tan, and Saeid Tizpaz-Niari. 2026. On the Robustness of Fairness Practices: A Causal Framework for Systematic Evaluation. In *48th International Conference on Software Engineering (ICSE)*. To appear.

[45] Verya Monjezi, Ashutosh Trivedi, Vladik Kreinovich, and Saeid Tizpaz-Niari. 2025. Fairness Testing through Extreme Value Theory. In *Proceedings of the IEEE/ACM 47th International Conference on Software Engineering* (Ottawa, Ontario, Canada) *(ICSE '25)*. IEEE Press, 1501–1513. doi:10.1109/ICSE55347.2025.00070

[46] Verya Monjezi, Ashutosh Trivedi, Gang Tan, and Saeid Tizpaz-Niari. 2023. Information-Theoretic Testing and Debugging of Fairness Defects in Deep Neural Networks. *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)* (2023), 1571–1582. https://api.semanticscholar.org/CorpusID:258048637

[47] S. Moro, P. Rita, and P. Cortez. 2014. Bank Marketing. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5K306.

[48] Ramaravind K. Mothilal, Amit Sharma, and Chenhao Tan. 2020. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency* (Barcelona, Spain) *(FAT* '20)*. Association for Computing Machinery, New York, NY, USA, 607–617. doi:10.1145/3351095.3372850

[49] Giang Nguyen, Sumon Biswas, and Hridesh Rajan. 2023. Fix Fairness, Don't Ruin Accuracy: Performance Aware Fairness Repair using AutoML. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (San Francisco, CA, USA) *(ESEC/FSE 2023)*. Association for Computing Machinery, New York, NY, USA, 502–514. doi:10.1145/3611643.3616257

[50] ProPublica. 2021. Compas Software Ananlysis. https://github.com/propublica/compas-analysis. Online.

[51] J. Ross Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[52] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. Association for Computing Machinery, New York, NY, USA, 1135–1144. doi:10.1145/2939672.2939778

[53] Salvador Robles Herrera, Verya Monjezi, Vladik Kreinovich, Ashutosh Trivedi, and Saeid Tizpaz-Niari. 2024. Predicting Fairness of ML Software Configurations. In *Proceedings of the 20th International Conference on Predictive Models and Data Analytics in Software Engineering* (Porto de Galinhas, Brazil) *(PROMISE 2024)*. Association for Computing Machinery, New York, NY, USA, 56–65. doi:10.1145/3663533.3664040

[54] Cynthia Rudin. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence* 1, 5 (2019), 206–215. doi:10.1038/s42256-019-0048-x

[55] Cynthia Rudin and Berk Ustun. 2018. Optimized Scoring Systems: Toward Trust in Machine Learning for Healthcare and Criminal Justice. *Interfaces* 48, 5 (Oct. 2018), 449–466. doi:10.1287/inte.2018.0957

[56] Armando Solar-Lezama, Liviu Tancau, Rastislav Bodik, Sanjit Seshia, and Vijay Saraswat. 2006. Combinatorial sketching for finite programs. In *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems* (San Jose, California, USA) *(ASPLOS XII)*. Association for Computing Machinery, New York, NY, USA, 404–415. doi:10.1145/1168857.1168907

[57] Bing Sun, Jun Sun, Long H. Pham, and Jie Shi. 2022. Causality-based neural network repair. In *Proceedings of the 44th International Conference on Software Engineering* (Pittsburgh, Pennsylvania) *(ICSE '22)*. Association for Computing Machinery, New York, NY, USA, 338–349. doi:10.1145/3510003.3510080

[58] Yan Shuo Tan, Chandan Singh, Keyan Nasseri, Abhineet Agarwal, James Duncan, Omer Ronen, Matthew Epland, Aaron Kornblith, and Bin Yu. 2025. Fast Interpretable Greedy-Tree Sums. *Proceedings of the National Academy of Sciences* 122, 7 (2025), e2310151122. arXiv:https://www.pnas.org/doi/pdf/10.1073/pnas.2310151122 doi:10.1073/pnas.2310151122

[59] Saeid Tizpaz-Niari, Ashish Kumar, Gang Tan, and Ashutosh Trivedi. 2022. Fairness-aware configuration of machine learning libraries. In *Proceedings of the 44th International Conference on Software Engineering* (Pittsburgh, Pennsylvania) *(ICSE '22)*. Association for Computing Machinery, New York, NY, USA, 909–920. doi:10.1145/3510003.3510202

[60] Sakshi Udeshi, Pryanshu Arora, and Sudipta Chattopadhyay. 2018. Automated directed fairness testing. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering* (Montpellier, France) *(ASE '18)*. Association for Computing Machinery, New York, NY, USA, 98–108. doi:10.1145/3238147.3238165

[61] Caterina Urban, Maria Christakis, Valentin Wüstholz, and Fuyuan Zhang. 2020. Perfectly parallel fairness certification of neural networks. *Proc. ACM Program. Lang.* 4, OOPSLA, Article 185 (nov 2020), 30 pages. doi:10.1145/3428253

[62] Zhaohui Wang, Min Zhang, Jingran Yang, Bojie Shao, and Min Zhang. 2024. MAFT: Efficient Model-Agnostic Fairness Testing for Deep Neural Networks via Zero-Order Gradient Search. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering* (Lisbon, Portugal) *(ICSE '24)*. Association for Computing Machinery, New York, NY, USA, Article 121, 12 pages. doi:10.1145/3597503.3639181

[63] Yisong Xiao, Aishan Liu, Tianlin Li, and Xianglong Liu. 2023. Latent Imitator: Generating Natural Individual Discriminatory Instances for Black-Box Fairness Testing. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis* (Seattle, WA, USA) *(ISSTA 2023)*. Association for Computing Machinery, New York, NY, USA, 829–841. doi:10.1145/3597926.3598099

[64] Normen Yu, Luciana Carreon, Gang Tan, and Saeid Tizpaz-Niari. 2025. FairLay-ML: Intuitive Debugging of Fairness in Data-Driven Social-Critical Software. In *2025 IEEE/ACM 47th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)* (Ottawa, ON, Canada). IEEE Press, 25–28. doi:10.1109/ICSE-Companion66252.2025.00016

[65] 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. doi:10.1145/3278721.3278779

[66] Jiang Zhang, Ivan Beschastnikh, Sergey Mechtaev, and Abhik Roychoudhury. 2022. Fair decision making via automated repair of decision trees. In *Proceedings of the 2nd International Workshop on Equitable Data and Technology* (Pittsburgh, Pennsylvania) *(FairWare '22)*. Association for Computing Machinery, New York, NY, USA, 9–16. doi:10.1145/3524491.3527306

[67] Jie M. Zhang and Mark Harman. 2021. "Ignorance and Prejudice" in Software Fairness. In *43rd IEEE/ACM International Conference on Software Engineering, ICSE 2021, Madrid, Spain, 22-30 May 2021*. IEEE, 1436–1447. doi:10.1109/ICSE43902.2021.00129

[68] Lingfeng Zhang, Yueling Zhang, and Min Zhang. 2021. Efficient white-box fairness testing through gradient search. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis* (Virtual, Denmark) *(ISSTA 2021)*. Association for Computing Machinery, New York, NY, USA, 103–114. doi:10.1145/3460319.3464820

[69] Mengdi Zhang and Jun Sun. 2022. Adaptive fairness improvement based on causality analysis. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Singapore, Singapore) *(ESEC/FSE 2022)*. Association for Computing Machinery, New York, NY, USA, 6–17. doi:10.1145/3540250.3549103

[70] Peixin Zhang, Jingyi Wang, Jun Sun, Guoliang Dong, Xinyu Wang, Xingen Wang, Jin Song Dong, and Ting Dai. 2020. White-box fairness testing through adversarial sampling. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering* (Seoul, South Korea) *(ICSE '20)*. Association for Computing Machinery, New York, NY, USA, 949–960. doi:10.1145/3377811.3380331

[71] Zhenjiang Zhao, Takahisa Toda, and Takashi Kitamura. 2024. Approximation-guided Fairness Testing through Discriminatory Space Analysis. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering* (Sacramento, CA, USA) *(ASE '24)*. Association for Computing Machinery, New York, NY, USA, 1007–1018. doi:10.1145/3691620.3695481

[72] Haibin Zheng, Zhiqing Chen, Tianyu Du, Xuhong Zhang, Yao Cheng, Shouling Ji, Jingyi Wang, Yue Yu, and Jinyin Chen. 2022. NeuronFair: interpretable white-box fairness testing through biased neuron identification. In *Proceedings of the 44th International Conference on Software Engineering* (Pittsburgh, Pennsylvania) *(ICSE '22)*. Association for Computing Machinery, New York, NY, USA, 1519–1531. doi:10.1145/3510003.3510123

[73] Chudi Zhong, Zhi Chen, Jiachang Liu, Margo Seltzer, and Cynthia Rudin. 2023. Exploring and interacting with the set of good sparse generalized additive models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '23)*. Curran Associates Inc., Red Hook, NY, USA, Article 2476, 27 pages. doi:10.52202/075280-2476

Received 2026-01-30; accepted 2026-06-25