MAAT: A Novel Ensemble Approach to Addressing Fairness and Performance Bugs for Machine Learning Software

Zhenpeng Chen
zp.chen@ucl.ac.uk
University College London
London, United Kingdom

Jie M. Zhang
jie.zhang@kcl.ac.uk
King's College London
London, United Kingdom

Federica Sarro
f.sarro@ucl.ac.uk
University College London
London, United Kingdom

Mark Harman
mark.harman@ucl.ac.uk
University College London
London, United Kingdom

ABSTRACT

Machine Learning (ML) software can lead to unfair and unethical decisions, making software fairness bugs an increasingly significant concern for software engineers. However, addressing fairness bugs often comes at the cost of introducing more ML performance (e.g., accuracy) bugs. In this paper, we propose MAAT, a novel ensemble approach to improving fairness-performance trade-off for ML software. Conventional ensemble methods combine different models with identical learning objectives. MAAT, instead, combines models optimized for different objectives: fairness and ML performance. We conduct an extensive evaluation of MAAT with 5 state-of-the-art methods, 9 software decision tasks, and 15 fairness-performance measurements. The results show that MAAT significantly outperforms the state-of-the-art. In particular, MAAT beats the trade-off baseline constructed by a recent benchmarking tool in 92.2% of the overall cases evaluated, 12.2 percentage points more than the best technique currently available. Moreover, the superiority of MAAT over the state-of-the-art holds on all the tasks and measurements that we study. We have made publicly available the code and data of this work to allow for future replication and extension.

CCS CONCEPTS

• **Software and its engineering → Software creation and management**; • **Computing methodologies → Machine learning**.

KEYWORDS

Software fairness, bias mitigation, fairness-performance trade-off, ensemble learning, machine learning software

ACM Reference Format:
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. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '22), November 14–18, 2022, Singapore, Singapore.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3540250.3549093

1 INTRODUCTION

Software systems have been widely adopted to make decisions in social-critical, human-related tasks, including credit assessment [1], disease detection [55], criminal justice [4], and hiring [31]. The wide adoption of such decision-making software raises concerns about software fairness bugs (i.e., unfair software decisions) [27]. These bugs have been frequently reported, related to protected attributes such as race [5, 9] and sex [7, 17]. They may particularly disadvantage minorities and protected groups, resulting in unethical and unacceptable consequences.

The issue in fairness has been studied in both Software Engineering (SE) [40] and Machine Learning (ML) [62] research communities since 2008. From the SE perspective, fairness is a non-functional software property, as such can be the subject of testing to find fairness bugs [79]. In recent years, the surging ML software (i.e., software that relies on ML to tackle decision problems), increases the prevalence of software fairness bugs [58, 67]. Such prevalence attracts increasing attention from the SE community, and researchers have called for actions by the SE research to tackle fairness bugs [27]. As a result, the SE literature has witnessed a large number of recent results on addressing fairness bugs (i.e., software bias[1] mitigation) [25, 26, 28, 29, 46, 47, 78].

Although bias mitigation methods aim to address fairness bugs, many theoretical and empirical studies [23, 34, 36, 47, 73] have revealed that the reduction of fairness bugs can come at the cost of introducing more performance (e.g., accuracy) bugs into ML software, causing decrease in ML performance. Therefore, fairness and ML performance can be conflicting goals in software development. Software engineers are often in a dilemma between the functional property (i.e., ML performance) and the non-functional property (i.e., fairness), known as "*fairness-performance trade-off*".

We present **MAAT**, a *fairness-performance ensemble* approach, for improving M̲L fa̲irness-performa̲nce t̲rade-off. Conventional ensemble methods combine different ML models with identical learning objectives (e.g., accuracy). In this paper, we propose a **novel ensemble method** that combines ML models with different learning objectives (i.e., fairness and performance) to optimize the trade-off between them. Specifically, MAAT learns individual models with fairness and ML performance as optimization goals separately, and combines the learned knowledge to make specific decisions. To get optimized fairness, we also design a *data debugging* technique to mitigate selection bias and label bias in the training data of ML software, both of which are demonstrated to be major causes of software unfairness [28, 73].

We conduct an extensive evaluation of MAAT on nine widely-adopted software decision tasks, which cover social, financial, and

[1] We treat "bias" and "unfairness" as synonyms, referring to the opposite of "fairness".

medical application domains, as well as scenarios with single and multiple protected attributes. We compare MAAT with five state-of-the-art bias mitigation methods from the ML and SE communities, using 15 types of fairness-performance measurements (i.e., combinations of three fairness metrics and five ML performance metrics).

We observe that MAAT is more effective in improving fairness-performance trade-off than existing methods. According to the state-of-the-art benchmarking tool [47], MAAT surpasses the trade-off baseline constructed by the tool in 92.2% of the studied cases, while existing methods achieve it in between 33.3% and 80.0% of the cases. When considering multiple protected attributes at the same time, MAAT beats the trade-off baseline in 96.5% of the cases evaluated, 46.3 percentage points more than the state-of-the-art. Furthermore, MAAT is widely applicable, as it outperforms existing methods on all the software decision tasks and fairness-performance measurements that we consider.

In summary, this paper makes the following contributions:

- We propose MAAT, a novel fairness-performance ensemble approach, to improve fairness-performance trade-off for ML software.
- We conduct a large-scale study of MAAT and 5 existing bias mitigation methods on 9 software decision tasks with 15 fairness-performance measurements. The results show that MAAT significantly outperforms the state-of-the-art.
- We make available the scripts and data used in our study [18] to the research community for other researchers to adopt MAAT or replicate and extend this work.

The rest of this paper is structured as follows. Section 2 introduces the preliminaries about software fairness. Section 3 presents the MAAT approach. Section 4 describes the evaluation settings and research questions. Section 5 reports and analyzes the results. Section 6 discusses the advantages and implications of MAAT as well as the threats to validity, followed by concluding remarks in Section 7.

2 PRELIMINARIES

We start by introducing the background and related work about software fairness.

2.1 Background

Classification tasks are the most widely-adopted research subjects in the software fairness literature [25, 26, 28, 29, 47, 78]. For such tasks, ML software can be considered a function that maps feature vectors to class labels. Among the features, some (such as sex and race; termed *protected attributes*) need to be protected against unfairness. Based on the value of a protected attribute, a population is partitioned into the *privileged* and *unprivileged* groups. It is recognized that ML software tends to produce the favorable class label for the members in the privileged group [58], thus making the unprivileged group at disadvantage. For example, the recidivism assessment software that US courts used has been shown to be particularly likely to falsely flag black defendants as future criminals compared to white defendants [5].

Building responsible software with *group fairness* has been an important ethical duty for software engineers. It requires that the protected attributes do not affect the decision outcomes and thus ML software treats the privileged and unprivileged groups equally. Group fairness has been advocated in legal regulations such as the four-fifths rule in US law [73], and widely studied in software bias mitigation research [58]. In this paper, we focus on classification tasks and group fairness.

2.2 Related Work

Fairness has been an important non-functional property that software engineers need to meet in software development practice. Therefore, major software companies have started to put significant efforts into software fairness. For instance, Meta (formerly Facebook) developed the Fairness Flow tool to detect bias in ML software [11]; Microsoft established the FATE group [8] to promote software fairness, and published the ethical principles of artificial intelligence [15], stating that ML software must be fair in real-life applications.

Meanwhile, fairness has been a hot research topic in the SE community. In particular, at ESEC/FSE 2018, researchers [27] set out a vision on how SE research can help reduce fairness bugs, fostering a series of SE efforts in software fairness. Next, we introduce related work about fairness bug detection and resolution.

Detecting fairness bugs (fairness testing): Chen et al. [33] provided a comprehensive survey of existing research on fairness testing. Galhotra et al. [43] proposed Themis to generate test suites for detecting causal discrimination in ML software. Udeshi et al. [69] leveraged the inherent robustness property in ML models for scalable fairness test generation. Angell et al. [20] presented an automated test suite generator to measure causal relationships between sensitive inputs and program behaviour. Aggarwal et al. [19] combined symbolic execution and local explainability for fairness test input generation. Zhang et al. [81] used gradient computation and clustering to generate discriminatory instances. Chakraborty et al. [30] used trustworthy explanation for uncovering underlying fairness bugs. Zhang et al. [80] generated diverse discriminatory seeds and individual discriminatory instances around these seeds through gradient search. Asyrofi et al. [21] generated bias-uncovering test cases by text mutation for sentiment analysis software.

Addressing fairness bugs (bias mitigation): Some researchers provided implications for bias mitigation through empirical studies. For example, Zhang and Harman [78] investigated potential influencing factors of software fairness and found that enlarging the feature set was a possible way to improve fairness. Valentim et al. [70] and Biswas and Rajan [26] explored the impact of different pre-processing techniques on fairness and derived insights for choosing appropriate techniques to improve software fairness.

Additionally, researchers proposed numerous bias mitigation methods to address fairness bugs, including pre-processing, in-processing, and post-processing methods [45, 58]. *Pre-processing* methods processed training data to reduce data bias; *in-processing* methods mitigated bias by optimizing training algorithms; *post-processing* methods modified prediction outcomes of ML software to improve fairness. Recently, IBM has launched a toolkit named AIF360 [14] to integrate popular pre-processing, in-processing, and post-processing methods, such as Reweighing [49], Adversarial Debiasing [77], and Reject Option Classification [51].

Furthermore, researchers proposed ensemble techniques that combined different bias mitigation methods/models [24, 28, 29, 48, 53] to address fairness bugs. These techniques often combined methods/models with the identical objective (i.e., fairness) to achieve better fairness than any of the individuals. For example, Chakraborty et al. [29] combined a pre-processing method (i.e., situation testing) and an in-processing method (i.e., multi-objective optimization), and demonstrated that the ensemble performed better than each of the individuals; similarly, their follow-up work [28] combined two pre-processing bias mitigation techniques, i.e., situation testing and data distribution balancing. Different from them, we propose an ensemble approach that combines models with different objectives (i.e., fairness and ML performance) to deal with the trade-off between them.

With the emergence of various bias mitigation methods, some work focused on empirical evaluation of them. For example, Biswas and Rajan [25] applied seven bias mitigation methods on real-world ML models collected from a crowd-sourced platform. However, existing work often measured the changes caused by bias mitigation methods in fairness and ML performance separately. In this way, it was unclear whether the improved fairness was simply the consequence of ML performance loss. To tackle this problem, Hort et al. [47] proposed a benchmarking tool named Fairea, which provided a unified baseline to evaluate and compare the fairness-performance trade-off of different bias mitigation methods. In this paper, we adopt Fairea to compare our approach and existing bias mitigation methods. We use ***"trade-off baseline"*** to refer to the baseline that Fairea provides, which is expected to be surpassed by any reasonable bias mitigation methods.

3 THE MAAT APPROACH

In this section, we first introduce the workflow of MAAT, then present a detailed description of its key components.

3.1 MAAT: In a Nutshell

MAAT is a fairness-performance ensemble approach for improving the fairness-performance trade-off in ML software development. Inspired by the ensemble theory [56, 66] that ensembles tend to yield better results when there is a diversity among the involved models, MAAT makes use of ML models with different objectives (i.e., fairness and performance), and combines the behaviours of them to make the final decisions.

Figure 1 presents the overview of MAAT. First, we train two individual models, named the **Fairness Model** and the **Performance Model**. The fairness model is optimized for fairness; the performance model is optimized for ML performance (e.g., accuracy). Each model maps the input feature vector to a probability vector, indicating the probability that the input belongs to each class label. We then combine the output probability vectors of the fairness model and the performance model to produce the final predictions.

MAAT is a general framework for tackling fairness-performance trade-off for ML software. Researchers and practitioners can design the fairness model, the performance model, and the combination strategy according to their applications. In the following, we introduce a default configuration of MAAT, with a fairness model that we propose using data debugging.

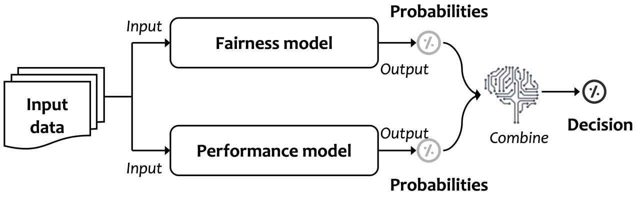


Figure 1: Overview of MAAT.

3.2 Fairness Model

We propose a training data debugging approach for obtaining the fairness model.

ML software is developed following the data-driven programming paradigm, and the training data determines its decision logic to a large extent [79]. As a result, bias in the training data is considered a root cause of ML software bias [28]. For example, in the Adult dataset [6], which is commonly used for predicting income of individuals, 31% of men are labeled with "high income" and 11% of women, a difference of almost three times. Therefore, ML software developed on this dataset tends to favor men.

Data debugging is an emerging technique, which aims to locate and modify the data that causes program bugs. For example, Wu et al. [74] proposed a data debugging approach, which allows users to complain about queries' output of database integrating ML inference and returns the smallest set of training data that can be removed to fix the database bugs; Kirschner et al. [54] proposed an approach to maximizing the subset of input that can be processed by the program, thus debugging as much input data as possible. Different from these work, we propose a data debugging approach for addressing fairness bugs, i.e., improving software fairness.

Specifically, we debug the training data by encoding the "We're All Equal" (WAE) worldview [41, 76] into it. The WAE worldview holds the belief that there is no statistical association between the outcome decision and the protected attribute, and has been widely advocated in the literature [58] and law [76]. To encode the worldview, we first divide the training data into four subgroups based on the values of the outcome label (*Favorable* or *Unfavorable*) and the protected attribute (*Privileged* or *Unprivileged*), and use PF, PU, UF, and UU to denote the number of samples in the Privileged & Favorable, Privileged & Unfavorable, Unprivileged & Favorable, and Unprivileged & Unfavorable subgroups. Then, we need to make the favorable rates of privileged and unprivileged groups equal, i.e., $\frac{PF}{PF+PU} = \frac{UF}{UF+UU}$, to satisfy the WAE criteria. However, in fact, the two rates in current training data are often unequal, or even very different (as shown in the aforementioned Adult dataset).

We design a debugging strategy according to the recognized major factors of training data bias. Existing work [28, 73] attributes training data bias to two factors: *selection bias*, which occurs when sampling real-world data as the training data in a way that happens to introduce an unexpected correlation between the protected attribute and the outcome label, and *label bias*, which occurs when the process that produces the labels is influenced by factors that are not germane to the determination of labels. To mitigate the selection bias, we aim to undersample the training data to remove the unexpected correlation, making $\frac{PF}{PF+PU} = \frac{UF}{UF+UU}$. Since many research results and real-world cases have shown that ML software tends to falsely produce the favorable label for the privileged and the unfavorable label for the unprivileged [58], label bias mainly exists in the Privileged & Favorable and Unprivileged & Unfavorable subgroups. Therefore, we perform undersampling by reducing data samples in the two subgroups to alleviate the label bias as well as the selection bias. Meanwhile, we keep the ratio of privileged and unprivileged individuals during the sampling process, as we expect that the final training data still shares the same privileged/unprivileged ratio as the real-world. We use a and b to denote the number of samples that we need to remove from the Privileged & Favorable and Unprivileged & Unfavorable subgroups, and formulate the aforementioned constraints as follows:

$$\begin{cases} \frac{PF-a}{PF-a+PU} = \frac{UF}{UF+UU-b}, \\ \frac{PF+PU}{UF+UU} = \frac{PF-a+PU}{UF+UU-b}, \\ a \geq 0, b \geq 0. \end{cases} \tag{1}$$

Based on the constraints, we calculate the numbers a and b. Then, we randomly remove a samples from the Privileged & Favorable subgroup and b samples from the Unprivileged & Unfavorable subgroup. Finally, based on the remaining training data, we train the fairness model to predict favorable or unfavorable outcome.

In Section 5.3, we also use the models obtained by applying existing bias mitigation methods as the fairness model, to investigate how different fairness models affect MAAT.

3.3 Performance Model

By default, the training process of ML models uses ML performance as the optimization goal. Therefore, we use the models obtained by traditional ML algorithms on the original training data, as the performance model.

In Section 5.3, we employ the models trained using different ML algorithms as the performance model, to investigate how different performance models affect MAAT.

3.4 Combination

The fairness model and the performance model produce their respective probability vectors based on identical inputs. Our default combination strategy is to average the produced probabilities of the two models. Let us consider a binary classification task with unfavorable and favorable labels (i.e., 0 and 1), and denote the two probability vectors obtained by the fairness model and the performance model as $[p_{0f}, p_{1f}]$ and $[p_{0p}, p_{1p}]$, where p_{0f} and p_{0p} indicate the probabilities that the input belongs to the class 0, p_{1f} and p_{1p} the class 1. Then, we average the two vectors to obtain the final probability vector $[\frac{p_{0f}+p_{0p}}{2}, \frac{p_{1f}+p_{1p}}{2}]$. If $\frac{p_{0f}+p_{0p}}{2}$ is greater than $\frac{p_{1f}+p_{1p}}{2}$, we predict the input as the class 0, otherwise the class 1. Averaging is a common strategy in ensemble learning [83], and we use it as the default combination strategy of MAAT.

In Section 5.4, we also explore other combination strategies to investigate how different strategies affect MAAT.

3.5 For Multiple Protected Attributes

Software systems may have multiple protected attributes that need to be considered at the same time. For such multi-attribute tasks, it is difficult for software engineers to improve fairness for each protected attribute simultaneously. For example, existing ML software, which has been removed gender bias [50], is demonstrated to exhibit severe racial bias [43].

MAAT, as a general framework, can be easily adopted for multi-attribute tasks. Specifically, we can train an individual fairness model for each protected attribute according to the steps in Section 3.2. Each fairness model predicts the same target (favorable or unfavorable). Then, given the input data, the performance model and each fairness model produce a probability vector, respectively. We average these vectors to obtain the final probability vector and then make the decision based on it.

In Section 5.5, we evaluate MAAT in multi-attribute decision tasks to demonstrate its effectiveness in dealing with multiple protected attributes simultaneously.

4 EVALUATION

In this section, we describe the evaluation design and propose our research questions.

4.1 Benchmark Datasets

We consider five representative benchmark datasets with different protected attributes (as shown in Table 1). The five datasets cover financial, social, and medical application domains. They have been the most widely-adopted datasets in the fairness research [26, 28, 29, 47, 58, 78] and integrated in the IBM AIF360 toolkit [22]. The number of datasets that we use aligns with the fairness literature, as previous work [47] points out that 90% of the fairness research uses no more than three datasets. Next, we introduce each dataset briefly.

- **Adult** [6, 39] dataset is used to predict whether individuals have annual income over $50K based on their demographic and financial information.
- **Compas** [4] dataset is used to predict whether defendants will be re-offended within two years based on their demographic information and criminal histories.
- **German** [1, 39] dataset is used to predict credit risk levels of people based on their demographic and credit information.
- **Bank** [2, 39] dataset is used to predict whether clients will subscribe a term deposit based on their demographic, financial, and social information.
- **Mep** [3] dataset is used to predict health care needs of individuals based on how Americans pay for medical care, health insurance, and out-of-pocket spending.

We covert the five datasets to nine benchmark tasks. First, we consider one protected attribute at a time, thus having *seven uni-attribute tasks* (e.g., Adult-Sex and Adult-Race). Second, we consider more than one protected attribute at the same time, thus having *two multi-attribute tasks* (i.e., Adult and Compas).

Table 1: Benchmark datasets.

Name	Protected attribute(s)	#Features	Favorable label	Majority label	Size
Adult	Sex, Race	14	1 (income > 50K)	0 (75.2%)	45,222
Compas	Sex, Race	10	0 (no recidivism)	0 (54.5%)	6,167
German	Sex	20	1 (good credit)	1 (70.0%)	1,000
Bank	Age	20	1 (subscriber)	0 (87.3%)	30,488
Mep	Race	41	1 (utilizer)	0 (82.8%)	15,830

4.2 Existing Methods

We use five existing bias mitigation methods for comparison. On the one hand, we consider three state-of-the-art methods proposed in the ML community: Reweighing (REW) [49], Adversarial Debiasing (ADV) [77], and Reject Option Classification (ROC) [51]. They have been integrated into the IBM AIF360 toolkit [22] and widely adopted in previous SE studies [25, 29, 47]. On the other hand, we employ two state-of-the-art methods recently proposed in SE venues: Fairway [29] at ESEC/FSE 2020 and Fair-SMOTE [28] at ESEC/FSE 2021. The five methods cover pre-processing, in-processing, post-processing, and ensemble bias mitigation methods. Next, we introduce each method briefly.

- **REW** [49] is a pre-processing method that calculates weights for training samples in each (group, label) combination.
- **ADV** [77] is an in-processing method that uses adversarial techniques to reduce evidence of the protected attribute in predictions while simultaneously maximizing ML performance.
- **ROC** [51] is a post-processing method that targets predictions with high uncertainty and tends to assign favorable outcomes to the unprivileged and unfavorable outcomes to the privileged.
- **Fairway** [29] combines pre- and in-processing techniques. First, it removes ambiguous data points from training data via situation testing. Second, it employs multi-objective optimization to improve fairness while maximizing ML performance.
- **Fair-SMOTE** [28] combines two pre-processing strategies. First, it generates new data points to make the numbers of training data in different subgroups equal. Second, it removes ambiguous data points from training data like Fairway.

4.3 Metrics and Measurements

We evaluate MAAT and existing methods in terms of 15 fairness-performance measurements, i.e., combinations of three fairness metrics and five ML performance metrics. In this section, we first introduce the fairness and ML performance metrics, and then describe how to measure the fairness-performance trade-off.

4.3.1 Fairness metrics. To measure software bias, we use group fairness metrics that are widely adopted in the literature [25, 26, 28, 29, 47, 78]. Let A be a protected attribute, with 1 as the privileged group and 0 the unprivileged group; let Y be the original class label and $\hat{Y}$ the predicted label, with 1 as the favorable class and 0 the unfavorable class; let P denote the probability.

- **SPD** (Statistical Parity Difference) measures the difference in probabilities of favorable outcomes obtained by privileged and unprivileged groups:

$$SPD = P[\hat{Y} = 1|A = 0] - P[\hat{Y} = 1|A = 1]. \tag{2}$$

- **AOD** (Average Odds Difference) measures the average of the false-positive rate difference and the true-positive rate difference between privileged and unprivileged groups:

$$\begin{aligned} AOD = &\frac{1}{2}(|P[\hat{Y} = 1|A = 0, Y = 0] - P[\hat{Y} = 1|A = 1, Y = 0]| \\ &+ |P[\hat{Y} = 1|A = 0, Y = 1] - P[\hat{Y} = 1|A = 1, Y = 1]|). \end{aligned} \tag{3}$$

- **EOD** (Equal Opportunity Difference) measures the true-positive rate difference between privileged and unprivileged groups:

$$EOD = P[\hat{Y} = 1|A = 0, Y = 1] - P[\hat{Y} = 1|A = 1, Y = 1]. \tag{4}$$

There is another widely-adopted metric called Disparate Impact (DI). Like SPD, DI compares the probabilities of favorable outcomes in privileged and unprivileged groups. Specifically, DI computes the ratio of the two probabilities, while SPD computes their difference. Between SPD and DI, we follow previous work [26, 47] to use only SPD.

For all the fairness metrics, we use their absolute values. In this way, these metrics suggest the greatest fairness when they equal to 0, and larger values indicate more bias.

4.3.2 Performance metrics. To measure ML performance, we use traditional classification metrics, including **precision**, **recall**, **F1-score**, as well as **accuracy**.

For a given class, precision is the proportion of samples predicted as this class that actually belong to it; recall is the proportion of samples belonging to this class that are predicted as it; F1-score is the harmonic mean of precision and recall. Following previous work in SE [32, 60, 61], we report the macro-average values for precision, recall, and F1-score to enable quick comparison of the overall performance over favorable and unfavorable classes. Specifically, we first calculate precision, recall, and F1-score for each class, and then average the results of the two classes.

Accuracy is the most widely-adopted metric in the fairness literature [25, 26, 28, 47, 49, 78], which measures how often a method makes the correct prediction. However, it is often criticized as not being suitable for the imbalanced class distribution, because it is easy for an ML model to obtain a high accuracy just by predicting all samples as the majority class in such a distribution. Considering that some benchmark datasets (e.g., the Bank dataset) have an imbalanced class distribution, we use another metric called **MCC** (Matthews Correlation Coefficient), to mitigate the threat of accuracy. MCC has been demonstrated to be suitable for dealing with imbalanced scenarios [35] and widely adopted in SE research [64, 75]. It is calculated as:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}, \quad (5)$$

where TP, TN, FP, and FN denote the numbers of true positives, true negatives, false positives, and false negatives, respectively.

For all the five metrics, larger values indicate better ML performance. The value of MCC is between -1 and 1, where 1 indicates a perfect prediction, 0 no better than random prediction, and -1 total disagreement with observation; the values of other four metrics are between 0 and 1. By default, we use *performance* to refer to all the five metrics, and report the overall results for them.

4.3.3 Fairness-performance trade-off measurements. To compare the fairness-performance trade-off of bias mitigation methods, we adopt **Fairea** [47], a benchmarking tool proposed at ESEC/FSE 2021, which can classify the trade-off effectiveness of these methods into different levels. Specifically, it works as follows:

(1) Creation of trade-off baseline: Fairea constructs a trade-off baseline using the performance and bias (measured by the aforementioned metrics) of the original model and a series of pseudo models generated by randomly mutating model predictions (by replacing them with the majority class of data). Fairea considers different mutation degrees (i.e., the fraction of chosen predictions; 10%, 20%, ..., 100%) to obtain a series of pseudo models. The core insight of Fairea is that when it mutates the original model into a random guessing model gradually, the fairness will be improved as the predictive performance is equally worse in privileged and unprivileged groups. As any reasonable bias mitigation methods are expected to surpass these naive mutated models, Fairea uses them as the trade-off baseline.

(2) Division of effectiveness levels: The baseline classifies bias mitigation methods into five levels of trade-off effectiveness. A bias mitigation method falls in *win-win trade-off* if it improves both ML performance and fairness compared to the original model. On the contrary, if a method reduces both, it belongs to *lose-lose trade-off*. If a method improves ML performance but reduces fairness, it falls in *inverted trade-off*. There are another two levels of trade-off where methods reduce ML performance but improve fairness. Specifically, if a method achieves a better trade-off than the baseline, it belongs to *good trade-off*; otherwise, it falls in *poor trade-off*.

In the original paper [47], Fairea is applied to only SPD-accuracy and AOD-accuracy measurements. In this work, we extend our evaluation to 15 fairness-performance measurements, i.e., combinations of three fairness metrics and five ML performance metrics.

4.4 Experimental Settings

We describe the experimental settings in details to ensure the reproducibility of this work.

Implementation of datasets. As the five benchmark datasets have been integrated in the IBM AIF360, we use them by directly invoking off-the-shelf APIs provided by this toolkit. In addition, we follow previous work [28, 29, 47] to normalize all the feature values to be between 0 and 1.

Implementation of bias mitigation. For each benchmark task, we train the original models using three ML algorithms that are widely adopted in the fairness literature: Logistic Regression (LR) [25, 28, 29, 47, 78], Support Vector Machine (SVM) [25, 28, 47], and Random Forest (RF) [25, 28, 78]. Following previous work [28, 47, 78], we use the default configuration provided by the scikit-learn library [16] to implement each ML algorithm. We apply MAAT and existing bias mitigation methods to the original models, respectively. We apply REW, ADV, and ROC based on the IBM AIF360 [14]; we apply Fairway and Fair-SMOTE based on the code released by their authors [10, 12]. The experiments are repeated 50 times. Each time, we shuffle the dataset and randomly split it into 70% training data and 30% test data.

Implementation of Fairea. We adopt Fairea using the code released by its authors [13]. We create the trade-off baseline for each *(benchmark task, ML algorithm, fairness-performance measurement)* combination. Specifically, we train the original model 50 times; each time, based on the original model, we repeat the mutation procedure 50 times for each mutation degree. Finally, we construct the baseline using the mean result of the multiple runs, as suggested by Fairea [47].

Experimental environment. The experiments are implemented with Python 3.7.11 and TensorFlow 2.6.0, and executed on a Ubuntu 16.04 LTS with 128GB RAM, 2.3 GHz Intel Xeon E5-2653 v3 Dual CPU and two NVidia Tesla M40 GPUs.

4.5 Research Questions

We evaluate MAAT via answering the following research questions.

- **RQ1 (Trade-off effectiveness):** *What fairness-performance trade-off does MAAT achieve?* This RQ compares MAAT with existing bias mitigation methods by analyzing which trade-off effectiveness levels they belong to overall according to the benchmarking tool Fairea [47].
- **RQ2 (Applicability):** *How well does MAAT apply to different ML algorithms, decision tasks, and fairness-performance measurements?* In addition to the overall effectiveness, we further analyze the effectiveness of MAAT on different ML algorithms, decision tasks, and measurements to evaluate its applicability.
- **RQ3 (Influence of fairness and performance models):** *How do different fairness models and performance models affect MAAT?* RQ1 and RQ2 evaluate MAAT with the default fairness model and performance model. In this RQ, we adopt different fairness models and performance models to investigate the impacts of the two key components on MAAT.
- **RQ4 (Influence of combination strategies):** *How do different combination strategies affect MAAT?* We use averaging, a common ensemble strategy, as the default combination strategy of MAAT. In this RQ, we investigate other combination strategies to provide implications for further improvement of MAAT.
- **RQ5 (Multiple protected attributes):** *Is MAAT effective when dealing with multiple protected attributes at the same time?* In RQs1~4, we consider one protected attribute at a time for ease of comparison with previous work [25, 26, 29, 47, 78], because little previous work supports multiple attributes [28]. However, real-world software systems may need to consider multiple protected attributes at the same time. This RQ investigates whether MAAT provides an effective solution to such common but often overlooked application scenarios (i.e., multi-attribute tasks).

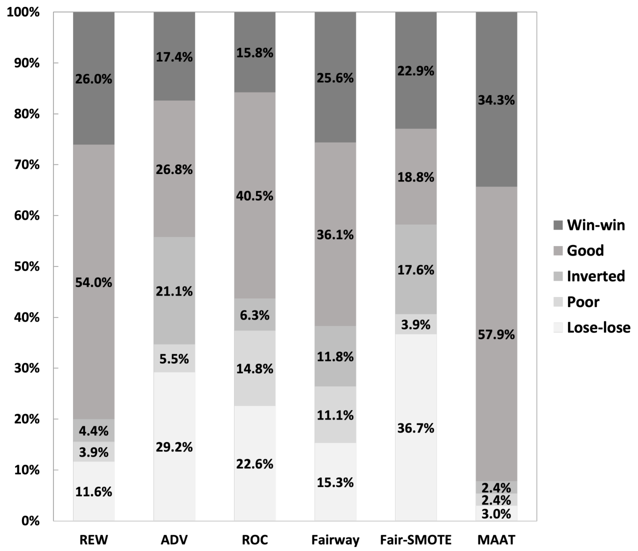


Figure 2: (RQ1) Effectiveness level distributions of MAAT and existing methods in uni-attribute benchmark tasks. Overall, MAAT achieves the best trade-off, with 92.2% of the mitigation cases falling in good or win-win trade-off.

5 RESULTS

In this section, we answer our RQs based on experimental results.

5.1 RQ1: Trade-off Effectiveness

This RQ evaluates the effectiveness of MAAT and existing methods in seven uni-attribute benchmark tasks. In each task, each method is applied with three ML algorithms 50 times. We treat each single run as an individual mitigation case. As a result, we have 6×7×3×50=6,300 cases in total. We answer RQ1 based on them.

First, we compare MAAT and existing methods in terms of the effectiveness levels classified by Fairea. Figure 2 shows the results. We observe that MAAT achieves good or win-win trade-off (i.e., beating the trade-off baseline constructed by Fairea) in the most cases, accounting for 92.2%. In contrast, the corresponding proportions of REW, ADV, ROC, Fairway, and Fair-SMOTE are 80.0%, 33.3%, 66.7%, 69.8%, and 33.3%, respectively. Moreover, MAAT achieves poor or lose-lose trade-off in much fewer cases (only 5.4%) than existing methods. For example, Fair-SMOTE suffers from poor or lose-lose trade-off in 40.6% of the cases, about seven times more than MAAT.

Then, we dive deeper to investigate the reason behind the effectiveness of MAAT, by analyzing its impact on fairness and ML performance, respectively. To this end, for each *(task, ML algorithm, fairness/performance metric)* combination scenario, we compare the metric values of the 50 original models and the 50 models after applying MAAT. We use the non-parametric Mann Whitney U-test [57] (which suits our purpose well as it does not assume normality) to test whether the fairness/performance is significantly improved/decreased. The fairness/performance change is considered statistically significant, only if the p-value of the computed statistic is lower than 0.05. We also compare the original models with the models applied existing bias mitigation methods. For each method, we calculate the proportions of scenarios where it significantly improves fairness and decreases performance, respectively.

Table 2: (RQ1) Proportions of scenarios where each method significantly improves fairness and decreases performance. MAAT significantly improves fairness in 96.8% of the scenarios, without decreasing ML performance too much.

	REW	ADV	ROC	Fairway	Fair-SMOTE	MAAT
Fairness ↑	87.3%	33.3%	66.7%	69.8%	33.3%	96.8%
Performance ↓	42.9%	51.4%	76.2%	61.9%	48.6%	44.8%

Table 2 presents the results. We observe that MAAT significantly improves fairness in 96.8% of the scenarios, while existing methods are in between 33.3% and 87.3%. Moreover, MAAT significantly decreases ML performance in 44.8% of the scenarios, only 1.9% more than the current best alternative (42.9%). In summary, the effectiveness of MAAT in fairness-performance trade-off is because that compared to existing methods, MAAT significantly improves fairness in much more scenarios without decreasing ML performance too much.

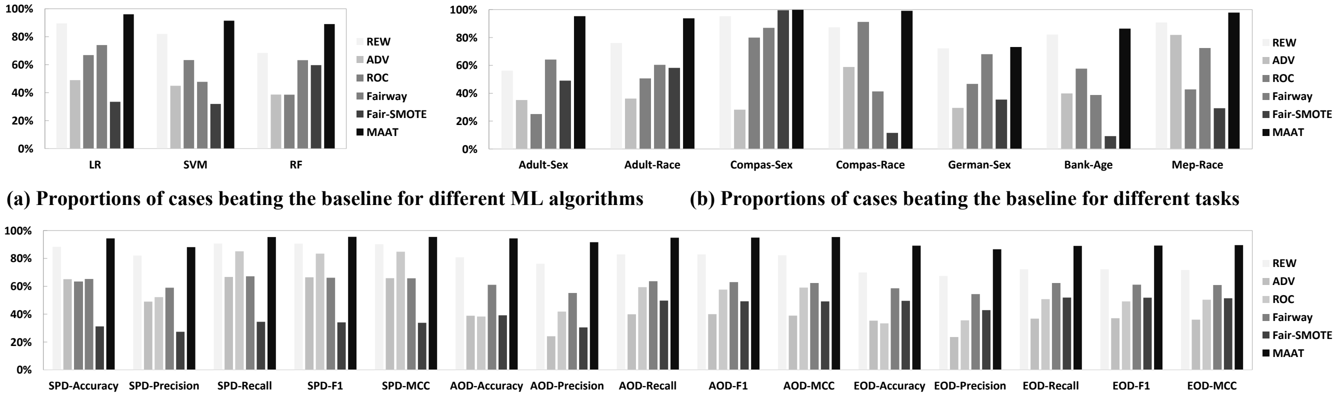
> **Ans. to RQ1:** MAAT surpasses the trade-off baseline in 92.2% of the overall cases evaluated, 12.2 percentage points more than the best technique currently available. This is because MAAT can significantly improve fairness in 96.8% of the scenarios, without decreasing ML performance too much.

5.2 RQ2: Applicability

To evaluate the applicability of MAAT, we further compare MAAT with existing methods on different ML algorithms, decision tasks, and fairness-performance measurements. For ease of illustration, we use the proportion of mitigation cases that surpass the trade-off baseline constructed by Fairea (i.e., falling in good or win-win trade-off) as the effectiveness indicator.

Figures 3(a), (b), and (c) show the results organized by different ML algorithms, decision tasks, and measurements, respectively. We observe that MAAT surpasses the trade-off baseline in a larger proportion of cases than existing methods on all the ML algorithms, tasks, and measurements that we consider.

In addition, compared to existing methods, MAAT shows wide effectiveness. From Figure 3, we observe that existing methods show unstable effectiveness across different decision tasks. For example, in the Compas-Sex task, REW and Fair-SMOTE beat the trade-off baseline in 95.3% and 99.6% of the cases respectively, while in the Adult-Sex task, the proportion obtained by the two methods is only 56.2% and 49.0%. In contrast, MAAT beats the trade-off baseline in 99.9% and 95.3% of the cases in Compas-Sex and Adult-Sex tasks, with a difference of only 4.6%. MAAT also works well for small datasets, although it employs a undersampling strategy. For the smallest task that we consider (i.e., the German-Sex task), only 34 samples (a = 24, b = 10) out of the 1,000 samples are removed by MAAT, because the data bias in the original training data is relatively minor. Since MAAT mitigates bias while retaining the majority of the data, it beats the trade-off baseline in 73.2% of the cases in the German-Sex task, surpassing existing methods.



(a) Proportions of cases beating the baseline for different ML algorithms **(b) Proportions of cases beating the baseline for different tasks**

(c) Proportions of cases beating the baseline for different fairness-performance measurements

Figure 3: (RQ2) Proportion of mitigation cases that beat the trade-off baseline, organized by different models (a), benchmark tasks (b), and fairness-performance measurements (c). MAAT surpasses the trade-off baseline in a larger proportion of cases than existing methods on all the ML algorithms, tasks, and measurements that we consider.

> **Ans. to RQ2:** The superiority of MAAT over the state-of-the-art holds on all the ML algorithms, decision tasks, and fairness-performance measurements that we study.

5.3 RQ3: Influence of Fairness and Performance Models

This RQ explores the impact of different fairness models and performance models on the effectiveness of MAAT.

5.3.1 Fairness models. Existing bias mitigation methods (described in Section 4.2) have been demonstrated to be able to improve fairness [25, 28, 29]. In this section, we use them to obtain the fairness model and compare with our training data debugging technique. MAAT requires the fairness model to produce the probability vector, but the IBM AIF360 toolkit does not provide this API for ADV and ROC. Therefore, here, we employ REW, Fairway, and Fair-SMOTE for experiments. As a result, we have three variant methods, denoted as M-REW, M-Faiway, and M-Fair-SMOTE, which differ from MAAT only in the fairness model. We compare them with their original versions and the default setting of MAAT. To ease the comparison, we use the proportion of cases that surpass the trade-off baseline as the effectiveness indicator.

Table 3 presents the results. Compared to their original versions, M-REW, M-Faiway, and M-Fair-SMOTE achieve 4.0%, 4.1%, and 2.2% more cases beating the trade-off baseline respectively, indicating the potential of MAAT in improving the trade-off effectiveness of existing bias mitigation methods. Then we compare the three variants with the default setting of MAAT. We find that our training data debugging technique is more effective in fairness-performance trade-off than existing methods under the framework of MAAT, although it is simpler than them. It achieves 8.2% more cases beating the trade-off baseline than the best variant (i.e., 92.2% vs. 84.0%). It is not the first time in SE research to observe that simple techniques produce better results [42, 63, 82]. It is important to stress that the point is not to deprecate the existing advanced techniques. We would like to demonstrate that, for MAAT, the training data debugging technique proposed by us is more effective.

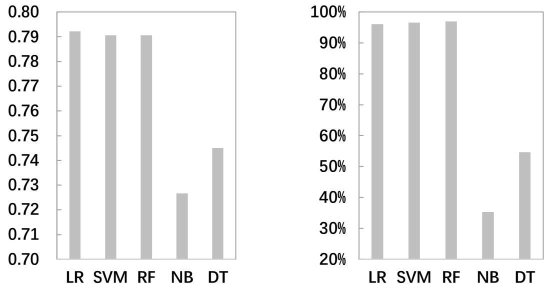
Table 3: (RQ3) Proportions of cases beating the trade-off baseline, achieved by existing bias mitigation methods, their combinations with MAAT, and the default setting of MAAT. The results show that the ensemble approach of MAAT can improve the trade-off for each method, but the default setting of MAAT still performs the best.

REW	Fairway	Fair-SMOTE	M-REW	M-Fairway	M-Fair-SMOTE	MAAT
80.0%	61.7%	41.7%	84.0%	65.8%	44.0%	92.2%

5.3.2 Performance models. To investigate the impact of the performance model on MAAT, we use different ML algorithms to obtain different performance models, while keeping the fairness model unchangeable. Specifically, we take the fairness model trained using the LR algorithm as the example, and change the performance model with the models trained using LR, SVM, RF, and two other very popular ML algorithms, i.e., Naive Bayes (NB) and Decision Tree (DT). Each variant of MAAT is evaluated on seven uni-attribute tasks 50 times. We calculate the average of ML performance of each performance model and the trade-off effectiveness (indicated as the proportion of cases beating the baseline) of the corresponding variant of MAAT over the 50 runs.

Figure 4 shows the results. Due to space limit, we show only the accuracy of these performance models in Figure 4(a).[2] We observe that the performance models trained using the NB and DT algorithms exhibit poor ML performance compared to LR, RF, and SVM; correspondingly, the variants of MAAT that use the two performance models yield lower proportions (35.3% and 54.6%) of cases surpassing the trade-off baseline than the others (96.1%, 96.6%, and 97.0%). It indicates that MAAT tends to have better effectiveness with more accurate performance models.

[2] The figure for all the performance metrics can be found in our repository [18].



(a) Accuracy of different performance models

(b) Proportions of cases beating the trade-off baseline, for different variants of MAAT

Figure 4: (RQ3) Impact of the performance model on MAAT. MAAT tends to have better effectiveness with more accurate performance models.

> **Ans. to RQ3:** Our training data debugging technique is more effective than existing bias mitigation methods in improving fairness-performance trade-off under the framework of MAAT. In addition, MAAT tends to have better effectiveness with more accurate performance models.

5.4 RQ4: Influence of Combination Strategies

This RQ investigates the impact of different combination strategies on the effectiveness of MAAT. To this end, we employ 11 strategies, i.e., *0-1*, *0.1-0.9*, ..., *0.9-0.1*, and *1-0*, which combine the output probability vectors of the performance model ($[p_{0p}, p_{1p}]$) and the fairness model ($[p_{0f}, p_{1f}]$) in different proportions. For example, the *0.1-0.9* strategy calculates the final probability vector as $0.1 * [p_{0p}, p_{1p}] + 0.9 * [p_{0f}, p_{1f}]$. We use the proportion of cases surpassing the trade-off baseline as the effectiveness indicator.

Figure 5 presents the result for each strategy in each benchmark task and over all tasks. From Figure 5(a), we observe that different benchmark tasks have different optimal strategies. For example, for the Adult-Sex task, *0.8-0.2* is the optimal strategy, with 99.7% of mitigation cases beating the baseline; for the Adult-Race task, *0.6-0.4* is the optimal, with 95.9% of cases beating the baseline. Overall, the averaging strategy (i.e., *0.5-0.5*) that we use as the default setting shows the best effectiveness, as shown in Figure 5(b).

We then compare the *0.5-0.5* strategy (i.e., MAAT) with the *0-1* strategy (i.e., the fairness model). We find that the fairness model itself improves fairness in 81.0% of the cases studied (96.8% for MAAT) and decreases performance in 65.7% of the cases (44.8% for MAAT). As a result, the fairness model is not as effective as MAAT, beating the trade-off baseline in only 79.0% of the cases (92.2% for MAAT). This finding supports the need for the ensemble approach.

The results provide the following implications for the adoption of MAAT in real-world applications: (1) The ensemble strategy in MAAT can be configured to balance the improvements between fairness and performance via adjusting the combination strategy. In fact, this is one advantage of our novel fairness-performance ensemble approach. Indeed, in practice software engineers may have different requirements regarding fairness and performance. They could try different strategies, and compare the effectiveness of these strategies on validation data to find the optimal one for them. Although the simple linear combination that we use can outperform existing methods, we still encourage future work to try more advanced strategies to further improve MAAT. (2) For the applications that do not have enough validation data for strategy selection, the default averaging strategy is a safe option.

> **Ans. to RQ4:** Different bias mitigation tasks have different optimal combination strategies. Overall, the averaging strategy that we adopt by default achieves the best effectiveness.

5.5 RQ5: Multiple Protected Attributes

The first four RQs explore bias mitigation with a single protected attribute, which is the focus of the current fairness literature [25, 26, 29, 47, 78]. Nevertheless, bias mitigation with multiple protected attributes is a demanding task. Therefore, this RQ explores the effectiveness of MAAT with multiple protected attributes, and compares it with Fair-SMOTE [28]. According to Chakraborty et al. [28], Fair-SMOTE is the only approach that reduces bias for multiple protected attributes at the same time.

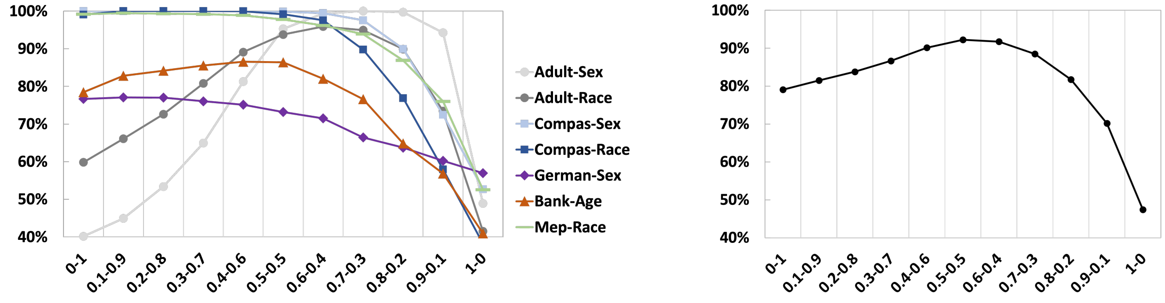
We compare MAAT and Fair-SMOTE in two multi-attribute tasks (i.e., Adult and Compas). For MAAT, we train a fairness model for each protected attribute, and then combine the fairness models with the performance model; for Fair-SMOTE, we balance the training data with respect to the class and each protected attribute by data generation, as suggested by its authors [28]. In each task, each method is implemented with LR, SVM, and RF 50 times. As a result, we have 2×2×3×50=600 mitigation cases in total. We answer RQ5 based on these cases.

We compare MAAT and Fair-SMOTE in terms of the trade-off effectiveness levels classified by Fairea. Figure 6 shows the results for each protected attribute in each task. On the one hand, MAAT achieves good or win-win trade-off (i.e., beating the trade-off baseline constructed by Fairea) in a larger proportion of the studied cases (MAAT: 96.5% vs. Fair-SMOTE: 50.2%). The gap between the two methods is particularly obvious in some tasks. For example, in the Adult task, MAAT beats the baseline for race in 93.9% of the cases, while Fair-SMOTE achieves it in only 1.7%. On the other hand, MAAT achieves poor or lose-lose trade-off in fewer cases. Specifically, the proportion of cases falling in poor or lose-lose trade-off achieved by MAAT ranges from 0.4% to 4.4%, while Fair-SMOTE is from 7.7% to 69.6%.

> **Ans. to RQ5:** When there is more than one protected attribute, MAAT still outperforms the state-of-the-art. Specifically, it beats the trade-off baseline in 96.5% of the studied cases, 45.2 percentage points more than the state-of-the-art.

6 DISCUSSION

In this section, we discuss the advantages of MAAT and the threats to the validity of our results.



(a) Proportions of cases beating the baseline, for different combination strategies on each benchmark task

(b) Overall proportions of cases beating the baseline, for different combination strategies

Figure 5: (RQ4) Impact of combination strategies on MAAT. Although different benchmark tasks have different optimal strategies, the averaging strategy (i.e., *0.5-0.5* in the figure) achieves the best effectiveness overall.

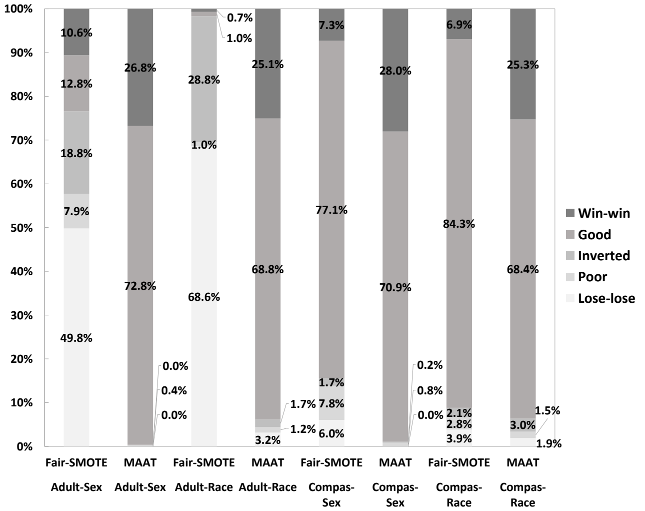


Figure 6: (RQ5) Effectiveness level distributions of MAAT and Fair-SMOTE in multi-attribute tasks. On average, MAAT achieves good or win-win trade-off in a much larger proportion of the mitigation cases than Fair-SMOTE (96.5% vs. 50.2%).

6.1 Why MAAT?

Uncompromising. MAAT improves fairness significantly in 96.8% of the scenarios studied, while not compromising ML performance much. In other words, it is more effective in fairness-performance trade-off than existing methods. The conclusion is supported by an extensive evaluation on 15 fairness-performance measurements and 9 decision tasks, which increases our confidence on the claim.
Widely applicable. MAAT can be easily adopted for different ML algorithms, because it just debugs the training data to find and remove bias, and does not rely on the internal logic of ML algorithms. In contrast, there have been many techniques that tackle the fairness-performance trade-off problem using optimization techniques during the training process [52, 77]. These techniques need to design different optimization strategies to align with the internal logic of different ML algorithms, and thus are algorithm-specific. In addition, in Section 5.2, we show the wide effectiveness of MAAT across different ML algorithms, decision tasks, and fairness-performance measurements.
Versatile. The core idea of MAAT is task-independent, although we evaluate it only in common classification tasks like previous work [25, 26, 28, 29, 47, 78]. MAAT combines the performance model and the fairness model obtained based on our training data debugging technique. This idea can be directly applied to regression tasks and even deep learning tasks (e.g., face recognition and text classification), to have a broad impact on a variety of real-world software systems.
Fast. MAAT is much faster, compared to the state-of-the-art ensemble bias mitigation methods (i.e., Fairway and Fair-SMOTE). We use the LR algorithm as the example. Table 4 shows the execution time of the 50 runs of MAAT and the state-of-the-art methods with LR for different benchmark tasks. On average, MAAT is 72 times faster than Fairway and 195 times faster than Fair-SMOTE. It is reasonable since Fairway needs much more time for multi-objective optimization, while Fair-SMOTE spends a lot of time on data generation. In contrast, MAAT is simple, fast, yet effective.

Overall, MAAT explores a new ensemble way to combine models with different objectives (i.e., fairness and performance) to deal with the trade-off between them, and an extensive evaluation demonstrates the effectiveness of this new ensemble way. Our successful practice of MAAT may also shed light on other trade-off problems in software development. For example, existing work [68] demonstrated that the improvement of robustness, an important non-functional property that has attracted enormous software testing efforts [38, 44, 72], often leads to a reduction of ML performance, thus making them conflicting software objectives. To achieve a good trade-off between robustness and ML performance, researchers can borrow the insight of MAAT to design robustness-performance ensemble learning, which combines models that use robustness and ML performance as respective optimization objectives.

Table 4: Execution time (in seconds) of Fairway, Fair-SMOTE, and MAAT, for the LR algorithm. MAAT needs much less execution time than Fairway and Fair-SMOTE.

Task	Fairway	Fair-SMOTE	MAAT
Adult-Sex	5,946	8,553	77
Adult-Race	6,169	16,337	79
Compas-Sex	1,428	2,139	20
Compas-Race	1,594	1,641	20
German-Sex	91	126	5
Bank-Age	3,192	20,413	47
Mep-Race	1,760	5,219	31

6.2 Threats to Validity

Selection of tasks. The choice of benchmark tasks may be a threat to validity of our results. To mitigate this threat, we use nine benchmark tasks that have been widely adopted in the fairness literature [26, 28, 29, 47, 58, 78] and integrated in the well-known AIF360 toolkit, covering financial, social, and medical application domains. The use of widely-studied tasks guarantees a fair comparison with the state-of-the-art.

Selection of existing methods. For a paper that proposes a new bias mitigation approach, it is sufficient and common to demonstrate an improvement over the state-of-the-art. To this end, we use representative bias mitigation methods from the ML and SE communities, covering pre-processing, in-processing, post-processing, and ensemble methods. These methods have been demonstrated to be state-of-the-art in addressing fairness bugs [25, 28, 29].

Selection of ML algorithms. Although MAAT can apply to both classic ML algorithms and Deep Learning (DL) algorithms, we use classic ML algorithms in the evaluation for four reasons: (1) The most widely-adopted datasets for fairness research (listed in Table 1) are tabular data, while DL is more suitable for complex unstructured data, e.g., text and images [65]. (2) These widely-adopted datasets are relatively small in size (e.g., German dataset has 1,000 samples), where DL may easily overfit due to its nature of complexity. (3) Decision-making scenarios that demand fairness often also require explainability, while low explainability is a big disadvantage of DL. (4) State-of-the-art group fairness work [25, 26, 28, 29, 47, 78] also uses classic ML algorithms. In the future, one could replicate our work with more ML algorithms.

Selection of evaluation criteria. The measurements that we use may also be a threat. To alleviate this threat, we use 15 fairness-performance measurements, the most in the literature to date. In the future, with more fairness metrics being proposed, one could replicate this work with more evaluation criteria.

Implementation of existing methods. To mitigate the threat in implementation of existing methods, we shared our code with the authors of Fairway [29] and Fair-SMOTE [28]. Their first author checked and confirmed the soundness of our source code.

Access to protected attributes. Studying group fairness requires access to the protected attributes of interest, but in practice, this information might be unavailable due to the recent released regulations such as GDPR (General Data Protection Regulation) [71]. GDPR requires users' consent for collecting and using their personal information. However, over 90% of users consent to legal terms and service conditions without reading them [59]. Moreover, GDPR applies to only Europe. Consequently, protected attributes are still prevalent in the training data that many companies collect. In addition, simulation has been commonly used in companies [37] to generate data, which may contain protected attributes. These can explain why almost all the recent fairness papers [19, 20, 25, 26, 28–30, 43, 47, 69, 78, 80] are still working on fairness issues based on protected attributes.

7 CONCLUSION

This paper presents MAAT, a widely applicable, versatile, and fast fairness-performance ensemble approach, which improves fairness-performance trade-off for ML software. MAAT first trains individual models optimized for fairness and ML performance, and then synthesizes their outcomes to make the final decision. An extensive evaluation demonstrates that MAAT outperforms existing bias mitigation methods from the ML and SE communities. Moreover, the superiority of MAAT over the state-of-the-art holds on all the software decision tasks, ML algorithms, and fairness-performance measurements that we study. Furthermore, the successful practice of MAAT opens up to further opportunities for software engineers dealing with other conflicting objectives in software development.

ACKNOWLEDGMENTS

Zhenpeng Chen, Federica Sarro, and Mark Harman are supported by the ERC Advanced Grant under the grant number 741278 (EPIC: Evolutionary Program Improvement Collaborators). Jie M. Zhang is partially supported by the UKRI Trustworthy Autonomous Systems Node in Verifiability, with Grant Award Reference EP/V026801/2.

REFERENCES

[1] 1994. The German Credit dataset. https://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29. Retrieved on November 25, 2021.
[2] 2014. The Bank dataset. https://archive.ics.uci.edu/ml/datasets/Bank+Marketing. Retrieved on November 25, 2021.
[3] 2015. The Mep dataset. https://meps.ahrq.gov/mepsweb/data_stats/download_data_files_detail.jsp?cboPufNumber=HC-181. Retrieved on November 25, 2021.
[4] 2016. The Compas dataset. https://github.com/propublica/compas-analysis. Retrieved on November 25, 2021.
[5] 2016. Machine bias. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing. Retrieved on November 25, 2021.
[6] 2017. The Adult Census Income dataset. https://archive.ics.uci.edu/ml/datasets/adult. Retrieved on November 25, 2021.
[7] 2017. Semantics derived automatically from language corpora contain human-like biases. https://www.science.org/doi/10.1126/science.aal4230. Retrieved on November 25, 2021.
[8] 2018. FATE: Fairness, Accountability, Transparency, and Ethics in AI. https://www.microsoft.com/en-us/research/theme/fate/. Retrieved November 25, 2021.
[9] 2018. Study finds gender and skin-type bias in commercial artificial-intelligence systems. https://news.mit.edu/2018/study-finds-gender-skin-type-bias-artificial-intelligence-systems-0212. Retrieved on November 25, 2021.
[10] 2020. The GitHub repository of Fairway. https://github.com/joymallyac/Fairway. Retrieved on November 25, 2021.
[11] 2021. Facebook says it has a tool to detect bias in its artificial intelligence. https://qz.com/1268520/facebook-says-it-has-a-tool-to-detect-bias-in-its-artificial-intelligence/. Retrieved November 25, 2021.
[12] 2021. The GitHub repository of Fair-SMOTE. https://github.com/joymallyac/Fair-SMOTE/tree/master/Fair-SMOTE. Retrieved on November 25, 2021.
[13] 2021. The GitHub repository of Fairea. https://github.com/maxhort/Fairea. Retrieved on November 25, 2021.
[14] 2021. IBM AI Fairness 360. https://aif360.mybluemix.net. Retrieved on November 25, 2021.
[15] 2021. Microsoft AI principles. https://www.microsoft.com/en-us/ai/responsible-ai?activetab=pivot1%3aprimaryr6. Retrieved November 25, 2021.
[16] 2021. Scikit-learn. https://scikit-learn.org. Retrieved on November 25, 2021.

[17] 2021. When good algorithms go sexist: Why and how to advance AI gender equity. https://ssir.org/articles/entry/when_good_algorithms_go_sexist_why_and_how_to_advance_ai_gender_equity. Retrieved on November 25, 2021.

[18] 2022. Replication package. https://github.com/chenzhenpeng18/FSE22-MAAT.

[19] Aniya Aggarwal, Pranay Lohia, Seema Nagar, Kuntal Dey, and Diptikalyan Saha. 2019. Black box fairness testing of machine learning models. In *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2019*. 625–635.

[20] Rico Angell, Brittany Johnson, Yuriy Brun, and Alexandra Meliou. 2018. Themis: Automatically testing software for discrimination. In *Proceedings of the 2018 ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2018*. 871–875.

[21] Muhammad Hilmi Asyrofi, Zhou Yang, Imam Nur Bani Yusuf, Hong Jin Kang, Ferdian Thung, and David Lo. 2021. BiasFinder: Metamorphic test generation to uncover bias for sentiment analysis systems. *IEEE Transactions on Software Engineering* (2021).

[22] 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 T. Richards, Diptikalyan Saha, Prasanna Sattigeri, Moninder Singh, Kush R. Varshney, and Yunfeng Zhang. 2019. AI fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias. *IBM Journal of Research and Development* 63, 4/5 (2019), 4:1–4:15.

[23] Richard Berk, Hoda Heidari, Shahin Jabbari, Michael Kearns, and Aaron Roth. 2021. Fairness in criminal justice risk assessments: The state of the art. *Sociological Methods & Research* 50, 1 (2021), 3–44.

[24] Dheeraj Bhaskaruni, Hui Hu, and Chao Lan. 2019. Improving prediction fairness via model ensemble. In *Proceedings of the 31st IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2019*. 1810–1814.

[25] Sumon Biswas and Hridesh Rajan. 2020. Do the machine learning models on a crowd sourced platform exhibit bias? An empirical study on model fairness. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2020*. 642–653.

[26] Sumon Biswas and Hridesh Rajan. 2021. Fair preprocessing: Towards understanding compositional fairness of data transformers in machine learning pipeline. In *Proceedings of the 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2021*. 981–993.

[27] Yuriy Brun and Alexandra Meliou. 2018. Software fairness. In *Proceedings of the 2018 ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2018*. 754–759.

[28] 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 European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2021*. 429–440.

[29] 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 European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2020*. 654–665.

[30] Joymallya Chakraborty, Kewen Peng, and Tim Menzies. 2020. Making fair ML software using trustworthy explanation. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering, ASE 2020*. 1229–1233.

[31] Jason Chan and Jing Wang. 2018. Hiring preferences in online labor markets: Evidence of a female hiring bias. *Management Science* 64, 7 (2018), 2973–2994.

[32] Zhenpeng Chen, Yanbin Cao, Huihan Yao, Xuan Lu, Xin Peng, Hong Mei, and Xuanzhe Liu. 2021. Emoji-powered sentiment and emotion detection from software developers' communication data. *ACM Transactions on Software Engineering and Methodology* 30, 2 (2021), 18:1–18:48.

[33] Zhenpeng Chen, Jie M. Zhang, Max Hort, Federica Sarro, and Mark Harman. 2022. Fairness testing: A comprehensive survey and analysis of trends. *CoRR* abs/2207.10223 (2022).

[34] Zhenpeng Chen, Jie M. Zhang, Federica Sarro, and Mark Harman. 2022. A comprehensive empirical study of bias mitigation methods for software fairness. *CoRR* abs/2207.03277 (2022).

[35] Davide Chicco and Giuseppe Jurman. 2020. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC genomics* 21, 1 (2020), 1–13.

[36] Sam Corbett-Davies, Emma Pierson, Avi Feller, Sharad Goel, and Aziz Huq. 2017. Algorithmic decision making and the cost of fairness. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2017*. 797–806.

[37] William de Paula Ferreira, Fabiano Armellini, and Luis Antonio de Santa-Eulalia. 2020. Simulation in industry 4.0: A state-of-the-art review. *Computers & Industrial Engineering* 149 (2020), 106868.

[38] Xiaoning Du, Yi Li, Xiaofei Xie, Lei Ma, Yang Liu, and Jianjun Zhao. 2020. Marble: Model-based robustness analysis of stateful Deep learning systems. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering, ASE 2020*. 423–435.

[39] Dheeru Dua and Casey Graff. 2019. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml. University of California, Irvine, School of Information and Computer Sciences.

[40] Anthony Finkelstein, Mark Harman, S. Afshin Mansouri, Jian Ren, and Yuanyuan Zhang. 2008. "Fairness analysis" in requirements assignments. In *Proceedings of the 16th IEEE International Requirements Engineering Conference, RE 2008*. 115–124.

[41] Sorelle A. Friedler, Carlos Scheidegger, and Suresh Venkatasubramanian. 2021. The (im)possibility of fairness: Different value systems require different mechanisms for fair decision making. *Commun. ACM* 64, 4 (2021), 136–143.

[42] Wei Fu and Tim Menzies. 2017. Easy over hard: A case study on deep learning. In *Proceedings of the 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017*. 49–60.

[43] 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, ESEC/FSE 2017*. 498–510.

[44] Xiang Gao, Ripon K. Saha, Mukul R. Prasad, and Abhik Roychoudhury. 2020. Fuzz testing based data augmentation to improve robustness of deep neural networks. In *Proceedings of the 42nd International Conference on Software Engineering, ICSE 2020*. 1147–1158.

[45] Max Hort, Zhenpeng Chen, Jie M. Zhang, Federica Sarro, and Mark Harman. 2022. Bias mitigation for machine learning classifiers: A comprehensive survey. *CoRR* abs/2207.07068 (2022).

[46] Max Hort and Federica Sarro. 2021. Did you do your homework? Raising awareness on software fairness and discrimination. In *Proceedings of the 36th IEEE/ACM International Conference on Automated Software Engineering, ASE 2021*.

[47] Max Hort, Jie M. Zhang, Federica Sarro, and Mark Harman. 2021. Fairea: A model behaviour mutation approach to benchmarking bias mitigation methods. In *Proceedings of the 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Athens, ESEC/FSE 2021*. 994–1006.

[48] Vasileios Iosifidis, Besnik Fetahu, and Eirini Ntoutsi. 2019. FAE: A fairness-aware ensemble framework. In *Proceedings of the 2019 IEEE International Conference on Big Data, IEEE BigData 2019*. 1375–1380.

[49] Faisal Kamiran and Toon Calders. 2011. Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems* 33, 1 (2011), 1–33.

[50] Faisal Kamiran, Toon Calders, and Mykola Pechenizkiy. 2010. Discrimination aware decision tree learning. In *Proceedings of the 10th IEEE International Conference on Data Mining, ICDM 2010*. 869–874.

[51] Faisal Kamiran, Asim Karim, and Xiangliang Zhang. 2012. Decision theory for discrimination-aware classification. In *Proceedings of the 12th IEEE International Conference on Data Mining, ICDM 2012*. 924–929.

[52] Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma. 2012. Fairness-aware classifier with prejudice remover regularizer. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases, ECML/PKDD 2012*. 35–50.

[53] Patrik Joslin Kenfack, Adil Mehmood Khan, S.M. Ahsan Kazmi, Rasheed Hussain, Alma Oracevic, and Asad Masood Khattak. 2021. Impact of model ensemble on the fairness of classifiers in machine learning. In *Proceedings of the 2021 International Conference on Applied Artificial Intelligence, ICAPAI 2021*. 1–6.

[54] Lukas Kirschner, Ezekiel O. Soremekun, and Andreas Zeller. 2020. Debugging inputs. In *Proceedings of the 42nd International Conference on Software Engineering, ICSE 2020*. 75–86.

[55] Chayakrit Krittanawong, Hafeez Ul Hassan Virk, Sripal Bangalore, Zhen Wang, Kipp W Johnson, Rachel Pinotti, HongJu Zhang, Scott Kaplin, Bharat Narasimhan, Takeshi Kitai, et al. 2020. Machine learning prediction in cardiovascular diseases: a meta-analysis. *Nature Scientific reports* 10, 1 (2020), 1–11.

[56] Ludmila I Kuncheva and Christopher J Whitaker. 2003. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning* 51, 2 (2003), 181–207.

[57] Henry B Mann and Donald R Whitney. 1947. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics* (1947), 50–60.

[58] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A survey on bias and fairness in machine learning. *Comput. Surveys* 54, 6 (2021), 115:1–115:35.

[59] Jayashree Mohan, Melissa Wasserman, and Vijay Chidambaram. 2019. Analyzing GDPR compliance through the lens of privacy policy. In *Proceedings of the Heterogeneous Data Management, Polystores, and Analytics for Healthcare - VLDB 2019 Workshops*. 82–95.

[60] Nicole Novielli, Fabio Calefato, Davide Dongiovanni, Daniela Girardi, and Filippo Lanubile. 2020. Can we use SE-specific sentiment analysis tools in a cross-platform setting?. In *Proceedings of the 17th International Conference on Mining Software Repositories, MSR 2020*. 158–168.

[61] Nicole Novielli, Daniela Girardi, and Filippo Lanubile. 2018. A benchmark study on sentiment analysis for software engineering research. In *Proceedings of the*

15th International Conference on Mining Software Repositories, MSR 2018. 364–375.

[62] Dino Pedreshi, Salvatore Ruggieri, and Franco Turini. 2008. Discrimination-aware data mining. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2008*. 560–568.

[63] Chanathip Pornprasit and Chakkrit Tantithamthavorn. 2021. JITLine: A simpler, better, faster, finer-grained just-in-time defect prediction. In *Proceedings of the 18th IEEE/ACM International Conference on Mining Software Repositories, MSR 2021*. 369–379.

[64] Daniel Rodríguez, Israel Herraiz, Rachel Harrison, José Javier Dolado, and José C. Riquelme. 2014. Preliminary comparison of techniques for dealing with imbalance in software defect prediction. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, EASE 2014*. 43:1–43:10.

[65] Ravid Shwartz-Ziv and Amitai Armon. 2022. Tabular data: Deep learning is not all you need. *Information Fusion* 81 (2022), 84–90.

[66] Peter Sollich and Anders Krogh. 1995. Learning with ensembles: How overfitting can be useful. *Advances in Neural Information Processing Systems* 8 (1995).

[67] Zeyu Sun, Jie M. Zhang, Mark Harman, Mike Papadakis, and Lu Zhang. 2020. Automatic testing and improvement of machine translation. In *Proceedings of the 42nd International Conference on Software Engineering, ICSE 2020*. 974–985.

[68] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. 2019. Robustness may be at odds with accuracy. In *Proceedings of the 7th International Conference on Learning Representations, ICLR 2019*.

[69] 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, ASE 2018*. 98–108.

[70] Inês Valentim, Nuno Lourenço, and Nuno Antunes. 2019. The impact of data preparation on the fairness of software systems. In *Proceedings of the 30th IEEE International Symposium on Software Reliability Engineering, ISSRE 2019*. 391–401.

[71] Paul Voigt and Axel Von dem Bussche. 2017. The EU general data protection regulation (GDPR). *A Practical Guide, 1st Ed., Cham: Springer International Publishing* 10, 3152676 (2017), 10–5555.

[72] Jingyi Wang, Jialuo Chen, Youcheng Sun, Xingjun Ma, Dongxia Wang, Jun Sun, and Peng Cheng. 2021. RobOT: Robustness-oriented testing for deep learning systems. In *Proceedings of the 43rd IEEE/ACM International Conference on Software Engineering, ICSE 2021*. 300–311.

[73] Michael L. Wick, Swetasudha Panda, and Jean-Baptiste Tristan. 2019. Unlocking fairness: A trade-off revisited. In *Proceedings of the Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*. 8780–8789.

[74] Weiyuan Wu, Lampros Flokas, Eugene Wu, and Jiannan Wang. 2020. Complaint-driven training data debugging for query 2.0. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD 2020*. 1317–1334.

[75] Jingxiu Yao and Martin J. Shepperd. 2020. Assessing software defection prediction performance: Why using the Matthews correlation coefficient matters. In *Proceedings of Evaluation and Assessment in Software Engineering, EASE 2020*. 120–129.

[76] Samuel Yeom and Michael Carl Tschantz. 2021. Avoiding disparity amplification under different worldviews. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency,FAccT 2021*. 273–283.

[77] 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, AIES 2018*. 335–340.

[78] Jie M. Zhang and Mark Harman. 2021. Ignorance and prejudice in software fairness. In *Proceedings of the 43rd IEEE/ACM International Conference on Software Engineering, ICSE 2021*. 1436–1447.

[79] Jie M. Zhang, Mark Harman, Lei Ma, and Yang Liu. 2019. Machine learning testing: Survey, landscapes, and horizons. *IEEE Transactions on Software Engineering* (2019).

[80] 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, ISSTA 2021*. 103–114.

[81] 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 42nd International Conference on Software Engineering, ICSE 2020*. 949–960.

[82] Pingyi Zhou, Jin Liu, Xiao Liu, Zijiang Yang, and John C. Grundy. 2019. Is deep learning better than traditional approaches in tag recommendation for software information sites? *Information and Software Technology* 109 (2019), 1–13.

[83] Zhi-Hua Zhou. 2021. Ensemble learning. In *Machine learning*. Springer, 181–210.