A Comprehensive Empirical Study of Bias Mitigation Methods for Machine Learning Classifiers

ZHENPENG CHEN, University College London, United Kingdom
JIE M. ZHANG, King's College London, United Kingdom
FEDERICA SARRO, University College London, United Kingdom
MARK HARMAN, University College London, United Kingdom

Software bias is an increasingly important operational concern for software engineers. We present a large-scale, comprehensive empirical study of 17 representative bias mitigation methods for Machine Learning (ML) classifiers, evaluated with 11 ML performance metrics (e.g., accuracy), 4 fairness metrics, and 20 types of fairness-performance trade-off assessment, applied to 8 widely-adopted software decision tasks. The empirical coverage is much more comprehensive, covering the largest numbers of bias mitigation methods, evaluation metrics, and fairness-performance trade-off measures compared to previous work on this important software property. We find that (1) the bias mitigation methods significantly decrease ML performance in 53% of the studied scenarios (ranging between 42%∼66% according to different ML performance metrics); (2) the bias mitigation methods significantly improve fairness measured by the 4 used metrics in 46% of all the scenarios (ranging between 24%∼59% according to different fairness metrics); (3) the bias mitigation methods even lead to decrease in both fairness and ML performance in 25% of the scenarios; (4) the effectiveness of the bias mitigation methods depends on tasks, models, the choice of protected attributes, and the set of metrics used to assess fairness and ML performance; (5) there is no bias mitigation method that can achieve the best trade-off in all the scenarios. The best method that we find outperforms other methods in 30% of the scenarios. Researchers and practitioners need to choose the bias mitigation method best suited to their intended application scenario(s).

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

Additional Key Words and Phrases: Machine learning, bias mitigation, fairness-performance trade-off

ACM Reference Format:
Zhenpeng Chen, Jie M. Zhang, Federica Sarro, and Mark Harman. 2023. A Comprehensive Empirical Study of Bias Mitigation Methods for Machine Learning Classifiers. *ACM Trans. Softw. Eng. Methodol.* 1, 1, Article 1 (January 2023), 30 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Machine Learning (ML) software has made its way into a wide range of critical decision-making applications, such as hiring, criminal justice, credit risk prediction, and admissions [58]. There are several widely-known examples of software exhibiting unfair behaviour, relating to protected

Zhenpeng Chen, Federica Sarro, and Mark Harman are with the Department of Computer Science, University College London, London, United Kingdom. Emails: {zp.chen, f.sarro, mark.harman}@ucl.ac.uk. Jie M. Zhang is with the Department of Informatics, King's College London, London, United Kingdom. E-mail: jie.zhang@kcl.ac.uk. Zhenpeng Chen is the corresponding author.

1049-331X/2023/1-ART1 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

attributes such as gender [15, 17] and race [10, 16]. Unfair software behaviour may result in unacceptable and unethical consequences that adversely affect users in minority and/or historically disadvantaged groups. Moreover, when software falls within legal or regulatory frameworks, unfair behaviour also incurs legal risks to software engineers.

The fairness issue has been studied for some time in Software Engineering (SE) research [44], pre-dating the recent upsurge in ML applications. The SE research community has increasingly focused on fairness since then. This increased focus is induced by increasing software systems' reliance on ML as a powerful generic technique to tackle complex decision and prediction problems, bringing with it the potential for unfairness as a result of software bias.[1] In recent years, the SE literature has witnessed a large number of results on software bias and fairness [19, 23, 24, 26, 30–32, 45, 49, 50, 71, 77, 81].

Software engineers tend to regard fairness as a non-functional property (typically most pertinent to ML software [78], although also applicable more generally in SE [44]). In SE nomenclature, unfairness can be thought of "fairness bugs" [35], thereby motivating software engineers to seek techniques for fixing fairness bugs to reduce software bias (i.e., bias mitigation) [23, 30, 31, 49, 50].

With the emergence of various bias mitigation methods, SE researchers have started to evaluate and compare these methods [23, 24, 30, 31, 50]. Such empirical evaluations enable the development of scientific knowledge about how useful different bias mitigation methods are in different application scenarios. There is a widespread belief in the research community that bias mitigation methods often improve fairness at the cost of ML performance (e.g., accuracy), known as "*fairness-performance trade-off*" [22, 36, 72]. An effective bias mitigation method should improve fairness without decreasing ML performance too much, i.e., achieving a good trade-off between fairness and ML performance. Therefore, to evaluate the effectiveness of a bias mitigation method, researchers tend to not only measure the fairness improved by it, but also consider its effect on ML performance.

However, existing evaluations of bias mitigation methods have yet to achieve full coverage and completeness to give a comprehensive picture. In particular, researchers often use one or two metrics to measure the effects of existing bias mitigation methods on ML performance, overlooking other metrics that are widely used in industry and academia. For example, Zhang and Harman [77] and Hort et al. [50] measure ML performance in terms of only accuracy; Chakraborty et al. [31] use recall on favorable and unfavorable classes as only ML performance metrics; Biswas and Rajan [23, 24] measure only accuracy as well as F1-score on the favorable class.

Nevertheless, different situations require different metrics. For example, in an online advertising task targeting users with lower income, software engineers would care about precision for this category of users. It is because, with a fixed budget, advertisers can place advertisements for a fixed number of users to view or click [56]. Among these users, advertisers want as many of them as possible to be low-income (i.e., a high precision of low-income users). However, existing work does not consider the corresponding metric (i.e., precision on the unfavorable class) when evaluating bias mitigation methods on benchmark tasks such as the income prediction task [30, 31, 50, 77]. Therefore, based on the results of existing work, software engineers will have no findings on which to base their decisions for such scenarios.

This gap motivates us to evaluate existing bias mitigation methods using comprehensive metrics. Since different metrics measure the functional or non-functional properties of ML software from different aspects, we believe that the results of this study can provide insightful implications for real-world applications as well as a foundational baseline for further research and follow-on studies.

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

We present a comprehensive study, evaluating 17 representative bias mitigation methods in 8 widely-adopted benchmark tasks with 11 ML performance metrics, 4 fairness metrics, and 20 fairness-performance trade-off measures. Our study covers the largest number of bias mitigation methods, evaluation metrics, and fairness-performance trade-off measures in software fairness literature. Specifically, our study yields the following implications for fairness research and practice:

(1) The values of all ML performance metrics that we use decrease in a large proportion (42%~66%) of scenarios after applying bias mitigation methods. Therefore, it is important for researchers to use a comprehensive set of task-relevant ML performance metrics in their evaluations, to be sure to capture any decrease in ML performance caused by bias mitigation methods.
(2) Existing bias mitigation methods improve fairness measured by the used metrics in 46% of the scenarios that we study, and even lead to decrease in both fairness and ML performance in 25% of the scenarios. Therefore, a community effort is required to bring software fairness improvement to a level where it becomes more effective and usable in practice.
(3) The effectiveness of existing bias mitigation methods is affected by tasks, models, the choice of protected attributes, and the set of metrics used to measure fairness and ML performance; there is no bias mitigation method that can achieve the best trade-off in all the scenarios. Therefore, researchers and practitioners need to choose the suitable method according to their intended application scenario(s). Our results provide empirical guidance for such choices.
(4) We have made publicly available all the scripts and data used in this study [37] to allow for future replication and extension of our work.

The rest of this paper is organized as follows. Section 2 describes the background knowledge and motivation of this study. Section 3 presents our research questions and methodology. Section 4 reports and analyzes the results. Section 5 discusses threats to the validity. Section 6 summarizes the related work, followed by concluding remarks in Section 7.

2 PRELIMINARIES

This section provides background knowledge on software fairness and motivates this study.

2.1 Definition of Fairness

There are two primary types of fairness that researchers pursue, i.e., individual fairness and group fairness [50, 69, 77]. Individual fairness requires an ML model to produce similar predictive outcomes for similar individuals, while group fairness requires an ML model to treat different groups equally. Since existing bias mitigation methods [24, 30, 31, 50, 72, 77] focus on group fairness, in this paper, we also focus on group fairness.

In the context of group fairness, a population is partitioned into the privileged and unprivileged groups based on the values of protected attributes, which refer to the sensitive characteristics (e.g., sex and race) that need to be protected against unfairness. Usually, an unfair ML model tends to favor the privileged group (i.e., inclined to produce the favorable class for its members), thereby putting the unprivileged group at disadvantage. For example, in the recidivism assessment task, race is a protected attribute yet existing recidivism assessment systems have been demonstrated to recommend favourable decisions for white defendants compared to otherwise equivalent black defendants [10].

2.2 Motivation

We take the Adult dataset [1], which is commonly used for predicting income of individuals, as a motivating example to explain what a group fairness issue looks like. In this dataset, 31% of men and 11% of women are labeled with “high income”, a difference of almost three times. As a result,

the prediction model trained on this dataset tends to favor men, i.e., it is more likely to predict men as high income than women [30]. In other words, the obtained prediction model exhibits unfairness regarding sex (a protected attribute in the Adult dataset).

To tackle the group fairness issue, an intuitive idea is to ignore protected attribute information in the training and decision-making process. However, existing work [31] finds that it results in a similar level of unfairness as before. This is because sometimes non-protected attributes employed in the training process may contain information correlated to protected attributes, thereby still leading to potential discrimination.

Researchers have proposed other bias mitigation approaches, including pre-processing, in-processing, and post-processing methods [48, 58]. Pre-processing methods process the training data to mitigate data bias. They take the original training data of ML models as the input, and output the training data with less bias. In-processing methods improve group fairness during the training process. They take the training data as the input, and are applied in the training process to output fairer models. Post-processing methods modify the prediction outcomes of ML models to improve fairness. The input is the prediction outcomes of ML models, and the output is fairer outcomes.

These methods can improve fairness at the cost of ML performance. For example, Chakraborty et al. [31] remove ambiguous data points that may contain bias from training data. This helps improve fairness of the obtained ML models, but the loss of information in these data points can result in the decrease of classification accuracy.

Previous studies [23, 24, 30, 31] consider both fairness and ML performance in the evaluation of bias mitigation methods. However, the changes in ML performance and fairness caused by bias mitigation methods are often measured and visualised separately. Therefore, it is difficult to judge whether the improved fairness is simply the consequence of ML performance loss. To tackle this problem, Hort et al. [50] propose a model behavior mutation method to quantitatively benchmark and evaluate the fairness-performance trade-off of different bias mitigation methods.

Nevertheless, existing evaluations of bias mitigation methods have yet to achieve full coverage and completeness to give a comprehensive picture:

- Existing evaluations are conducted in terms of limited metrics. In particular, researchers often use one or two ML performance metrics for evaluation. For example, most of the fairness work [27–29, 42, 50–52, 54, 74, 77] measures ML performance using only accuracy. Chakraborty et al. [31] use recall on the favorable class and false alarm (i.e., 1 minus recall on the unfavorable class) to compare different methods. Biswas and Rajan [23, 24] measure accuracy as well as F1-score on the favorable class. In contrast, Chakraborty et al. [30] employ the most ML performance metrics, including accuracy and precision/recall/F1-score on the favorable class, but they still ignore other metrics that measure ML performance on the unfavorable class and those that measure overall performance on favorable and unfavorable classes.
- Existing evaluations focus on limited bias mitigation methods. For example, Bias and Rajan [23] evaluate seven of the methods that we use; Chakraborty et al. [31] evaluate five; Chakraborty et al. [30] evaluate three. Hort et al. [50] evaluate the most bias mitigation methods, but they focus on only methods proposed in the ML community.

In practice, researchers and practitioners need to choose the bias mitigation method best suited to different application scenarios (i.e., different requirements and evaluation metrics). Existing evaluations cannot provide comprehensive implications for such choices. For evaluation metrics that are not considered by previous studies, engineers have no idea which bias mitigation method is the most suitable. For metrics that have been considered, existing studies use them to evaluate limited methods, so engineers may still have no idea whether the bias mitigation method suggested by the evaluation results is actually the most suitable. For example, because Hort et al. [50] evaluate

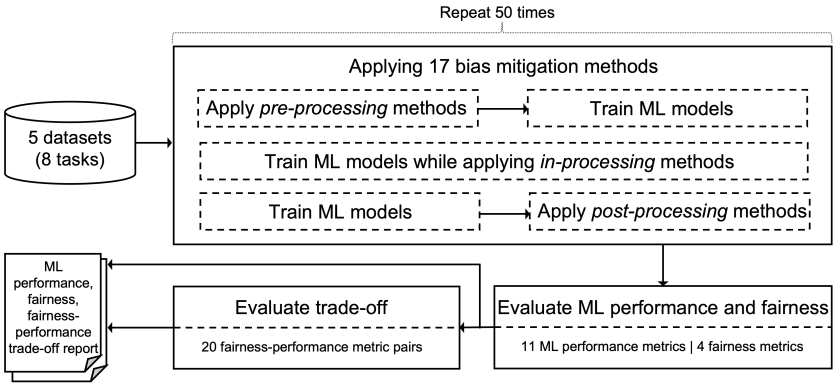


Fig. 1. Overview of experimental design.

only methods from the ML community, software engineers may have no idea whether the methods recently proposed in the SE community have better results.

To tackle these problems, we present an empirical study with comprehensive measurements and bias mitigation methods from both ML and SE communities. This large-scale study allows us to get a big picture of the literature as well as future research challenges and opportunities. Based on our results, engineers can select the most suitable bias mitigation method for their scenarios according to different requirements (i.e., different evaluation metrics).

While it is true that many practitioners and researchers may have their suspicions about current mitigation approaches, there is no scientific measurement of the effect size of these problems. In this paper, we provide an empirical study to properly measure and understand the magnitude of the problem. This is important also to provide a baseline against which to measure any future improvement. Without this, all we left with is a "belief" rather than quantitative scientific evidence.

3 EXPERIMENTAL SETUP

This section describes our research questions and experimental design.

3.1 Overview of Experimental Design

Fig. 1 illustrates our experimental design in a nutshell. First, we use five widely-adopted benchmark datasets (covering eight different software decision tasks) to train ML models. Second, we apply 17 representative bias mitigation methods from the ML and SE communities to these models. Third, we adopt 11 ML performance metrics and 4 fairness metrics to evaluate different methods. Finally, we consider ML performance and fairness together, and evaluate the fairness-performance trade-off of the 17 methods using 20 fairness-performance metric pairs. Specifically, we aim to answer the following research questions (RQs):

RQ1 (Influence on ML performance): *How do existing bias mitigation methods affect ML performance?* ML performance (e.g., accuracy and precision) represents important functional requirements of ML software, but bias mitigation methods can improve fairness at the cost of ML performance. Therefore, we first use various metrics to investigate how existing methods change ML performance.

RQ2 (Influence on fairness): *How do existing bias mitigation methods affect fairness?* Since the main goal of bias mitigation methods is to improve software fairness, we then use various fairness metrics to measure how well existing bias mitigation methods achieve this goal.

RQ3 (Fairness-performance trade-off): *What fairness-performance trade-off do existing bias mitigation methods achieve?* We finally consider fairness and ML performance together, and evaluate existing bias mitigation methods using different types of fairness-performance trade-off assessment, i.e., combinations of different fairness metrics and ML performance metrics.

In the following, we introduce the bias mitigation methods (Section 3.2), benchmark datasets (Section 3.3), metrics and measures (Section 3.4), and experimental settings (Section 3.5).

3.2 Bias Mitigation Methods

We focus our analysis on 17 representative bias mitigation methods proposed in the ML and SE communities. For the methods in the ML community, we follow previous work [23, 31, 50, 77] to employ the state-of-the-art ones implemented in the IBM AIF360 framework [9]. Specifically, we employ all the ten methods listed on its homepage [9], covering three types, i.e., pre-processing, in-processing, and post-processing methods. Next, we briefly introduce each method by type.

Pre-processing: Optimized Pre-processing (**OP**) [41] learns a probabilistic transformation to modify data features and labels. Learning Fair Representation (**LFR**) [75] learns fair representations by obfuscating information about protected attributes. Reweighting (**RW**) [51] generates different weights for samples in each (group, label) combination. Disparate Impact Remover (**DIR**) [43] modifies feature values to improve fairness while preserving rank-ordering within groups.

In-processing: Prejudice Remover (**PR**) [55] adds a discrimination-aware regularization term to the learning objective. Adversarial Debiasing (**AD**) [76] uses adversarial techniques to maximize accuracy and reduce evidence of protected attributes in the predictions simultaneously. Meta Fair Classifier (**MFC**) [29] takes the fairness metric as part of the input and returns a classifier optimized for the metric.

Post-processing: Reject Option Classification (**ROC**) [53] targets predictions with high uncertainty and tends to assign favorable outcomes to the unprivileged group and unfavorable outcomes to the privileged group. Calibrated Equalized Odds Post-processing (**CEO**) [64] optimizes over calibrated classifier score outputs to find probabilities with which to change output labels with an equalized odds objective. Equalized Odds Post-processing (**EOP**) [47] solves a linear program to find probabilities with which to change output labels to optimize equalized odds.

As for the methods proposed in the SE community, we use two methods recently published in top SE venues, including Fairway [31] at ESEC/FSE 2020 and Fair-SMOTE [30] at ESEC/FSE 2021.

Fairway [31] combines pre-processing and in-processing techniques to improve fairness. First, it evaluates the original labels of the training data and removes ambiguous data points that can eventually make the classifier biased. Then, it employs multi-objective optimization to maximize the model performance while making it fair.

Fair-SMOTE [30] is a pre-processing method that employs two strategies. First, it generates new data points to make the numbers of training data in different subgroups (i.e., combinations of different outcomes and protected attribute values) equal. Second, it uses the same method as Fairway to remove ambiguous data points from the training data.

In the AIF360 toolkit, MFC, ROC, and CEO are implemented with two, three, and three different metrics to guide the bias mitigation process, respectively. Specifically, MFC offers a choice between Disparate Impact (DI) and False Discovery Rate (FDR); ROC offers a choice among Statistical Parity Difference (SPD), Average Odds Difference (AOD), and Equal Opportunity Difference (EOD); CEO offers a choice among False Negative Rate (FNR), False Positive Rate (FPR), and a weighted metric to combine both. We implement and evaluate each of the settings. Therefore, we have a total of 17 bias mitigation methods for our study.

Table 1. Benchmark datasets for bias mitigation.

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

3.3 Benchmark Datasets

We follow previous work [77] to use five benchmark datasets implemented in the IBM AIF360 (as listed in Table 1). The five datasets cover social, financial, and medical domains, and are widely adopted in the fairness literature [24, 30, 31, 50, 58, 77]. The number of datasets used in this study compares favorably with the literature, as previous work [50] points out that 90% of fairness papers use no more than three datasets. Next, we briefly introduce each dataset.

Adult Income dataset [1] (a.k.a., **Adult** dataset) contains demographic and financial information about individuals extracted from the 1994 U.S. census data, and is used to predict whether the income of a person exceeds $50K a year or not.

ProPublica Recidivism dataset [4] (a.k.a., **Compas** dataset) contains demographic information and criminal histories of defendants from Broward County, and is used to predict whether a defendant will be re-offended within two years.

German Credit dataset [6] (a.k.a., **German** dataset) contains demographic and credit information of 1,000 individuals, and is used to predict people's credit risk levels.

Bank Marketing dataset [3] (a.k.a., **Bank** dataset) contains demographic, social, and financial information of clients of a Portuguese banking institution, and is used to predict whether a client will subscribe a term deposit.

Medical Survey 2015 dataset [11] (a.k.a., **Mep** dataset) contains data measuring how Americans use and pay for medical care, health insurance, and out-of-pocket spending, and is used to predict health care utilization of individuals.

As shown in Table 1, each dataset has its protected attribute(s) specified by its provider. We acknowledge that there may be other attributes that also need to be protected for each dataset. However, determining all protected attributes needs the assistance of requirements engineers, even legal practitioners, compliance officers, and policy makers, which is beyond the scope of this paper. Therefore, we follow previous work [24, 31, 50, 77] to use the specified protected attributes for study. In addition, although in practice software engineers may need to mitigate bias regarding all potentially protected attributes, most of existing bias mitigation methods treat each protected attribute individually for each task. Therefore, in line with previous work [24, 31, 50, 77], we consider one protected attribute each time and thus have eight dataset-attribute pairs (e.g., Adult-Sex and Adult-Race). We use the eight pairs as the eight bias mitigation tasks of this study and take each task as a binary classification problem.

3.4 Evaluation Metrics and Measures

We use 11 ML performance metrics, 4 fairness metrics, and 20 types of fairness-performance trade-off assessment for a comprehensive evaluation of existing bias mitigation methods. In the following, we briefly introduce the metrics and measures that we use.

Given a bias mitigation task, let a protected attribute be A, with 0 as the unprivileged group and 1 the privileged group; let the real classification label be Y and the predicted label $\hat{Y}$, with 0 as the unfavorable class and 1 the favorable class. In addition, we use Pr to denote probability.

3.4.1 ML Performance Metrics. For each bias mitigation method, we measure the ML performance changes caused by it on favorable and unfavorable classes in terms of precision, recall, and F1-score, which are widely employed in classification problems [33, 34, 61, 62].

Precision measures the exactness of a method. The precision for a given class c (i.e., 0 or 1) is calculated as:

$$Precision@c = Pr[Y = c|\hat{Y} = c]. \tag{1}$$

Recall measures the sensitivity of a method. The recall for a given class c is calculated as:

$$Recall@c = Pr[\hat{Y} = c|Y = c]. \tag{2}$$

F1-score measures a harmonic mean of precision and recall. The F1-score for a given class c is calculated as:

$$F1@c = \frac{2 \times Precision@c \times Recall@c}{Precision@c + Recall@c}. \tag{3}$$

In addition, to measure the overall performance, we follow previous classification work in SE [23, 24, 30, 50, 77] to use accuracy (**Acc**), which measures how often a method makes the correct prediction and is calculated as:

$$Acc = Pr[\hat{Y} = Y]. \tag{4}$$

Acc 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 datasets (e.g., the Bank dataset) that we use have an imbalanced class distribution, we follow previous SE work to use three additional macro-average metrics [34, 61, 62] and the Matthews Correlation Coefficient (**MCC**) metric [65, 73], which are all demonstrated to be suitable for dealing with imbalanced scenarios [38, 68]. For the macro-average metrics, we use macro-precision (**Mac-P**), macro-recall (**Mac-R**), and macro-F1 (**Mac-F1**), which take the average of precision, recall, and F1-score on the favorable and unfavorable classes, respectively. MCC is calculated as:

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

where TP, TN, FP, and FN denote the numbers of favorable samples predicted as favorable, unfavorable samples predicted as unfavorable, unfavorable samples predicted as favorable, and favorable samples predicted as unfavorable, respectively.

To summarize, we use 11 ML performance metrics, including F-P (precision for the favorable), F-R (recall for the favorable), F-F1 (F1-score for the favorable), UnF-P (precision for the unfavorable), UnF-R (recall for the unfavorable), UnF-F1 (F1-score for the unfavorable), Acc, Mac-P, Mac-R, Mac-F1, and MCC. The values of F-P, F-R, F-F1, UnF-P, UnF-R, UnF-F1, Acc, Mac-P, Mac-R, and Mac-F1 are between 0 and 1. The value of MCC is between -1 and 1, where 1 represents a perfect prediction, 0 no better than random prediction, and -1 total disagreement between prediction and observation. For all the metrics, larger values indicate better ML performance.

In practice, the choice of metrics depends on the intended applications. Specifically, requirements engineers can determine the metrics suitable for their applications without the need of considering all the 11 metrics. For example, for disease detection systems, engineers may pursue a high recall

on the unfavorable class (i.e., disease). Indeed, different types of datasets have different appropriate metrics. In our study, we use the full set of metrics for all the datasets due to the following reasons: 1) in practice, it is difficult to predict which metric is required, and thus using the full set of metrics in the study provides more comprehensive information for engineers to get reference when choosing metrics; 2) existing bias mitigation methods have used different types of metrics based on inconsistent rules, and thus using the full set of metrics for different datasets and methods provides a comprehensive and unified way to make comparisons.

3.4.2 Fairness Metrics. Based on different definitions of fairness, various fairness metrics have been proposed to measure the difference in classification between the privileged and unprivileged groups. In this work, we use the group fairness metrics that are most widely adopted in fairness research [23, 24, 30, 31, 50, 77].

Statistical Parity Difference (**SPD**) measures the difference in the acceptance rate of the favorable class between the privileged and unprivileged groups:

$$SPD = Pr[\hat{Y} = 1|A = 0] - Pr[\hat{Y} = 1|A = 1]. \tag{6}$$

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

$$\begin{aligned} AOD = &\frac{1}{2}(|Pr[\hat{Y} = 1|A = 0, Y = 0] - Pr[\hat{Y} = 1|A = 1, Y = 0]| \\ &+ |Pr[\hat{Y} = 1|A = 0, Y = 1] - Pr[\hat{Y} = 1|A = 1, Y = 1]|). \end{aligned} \tag{7}$$

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

$$EOD = Pr[\hat{Y} = 1|A = 0, Y = 1] - Pr[\hat{Y} = 1|A = 1, Y = 1]. \tag{8}$$

Error Rate Difference (**ERD**) measures the difference in the error rate between the privileged and unprivileged groups:

$$ERD = Pr[\hat{Y} \neq Y|A = 0] - Pr[\hat{Y} \neq Y|A = 1]. \tag{9}$$

There is another fairness metric called Disparate Impact (DI), which is also widely adopted in the fairness literature. DI and SPD both compare the probabilities of classifying samples as favorable in the privileged and unprivileged groups. Specifically, DI computes the ratio of the two probabilities, while SPD computes the difference of the two probabilities. Between SPD and DI, we follow previous work [24, 50] to use only SPD in our evaluation. In practice, the different acceptance rates of the favorable class between the privileged and unprivileged groups are allowed in certain scenarios, where SPD may be not a good metric. When it comes to fairness research, SPD is often adopted despite the existence of such threats because the main purpose of fairness research is to investigate the effectiveness of bias mitigation methods, in which scenario the validity of the bias in the data does not affect the evaluation of bias mitigation methods.

For all the fairness metrics, we use their absolute values. In this way, values equal to 0 indicate the greatest fairness; larger values indicate more bias.

3.4.3 Fairness-performance Trade-off Measures. It is difficult to evaluate which bias mitigation method is better based on fairness alone, since it is unclear whether the improved fairness is simply the consequence of ML performance loss. Therefore, here, we consider fairness and ML performance together and measure the fairness-performance trade-off of different methods.

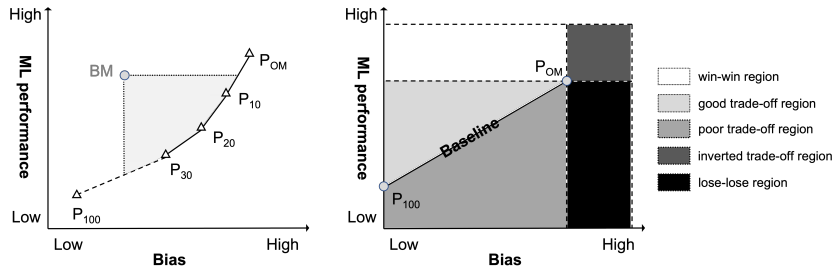


Fig. 2. Illustration of Fairea [50]. (a) presents the fairness-performance trade-off baseline, where P_{OM} represents the original model, BM the model after bias mitigation, and P_{10}, ..., $P100$ the points obtained by model behavior mutation. (b) presents the effectiveness regions of bias mitigation methods according to the changes in ML performance and bias.

To this end, we employ Fairea [50], a model behavior mutation method proposed at ESEC/FSE 2021, to benchmark and quantify the fairness-performance trade-off achieved by existing bias mitigation methods. Next, we briefly introduce Fairea, which is illustrated in Fig. 2.[2] Specifically, Fairea includes three steps as follows:

Step 1: Create trade-off baseline. Fairea presents the ML performance and fairness achieved by bias mitigation methods in a two-dimensional coordinate system as shown in Fig. 2(a). It constructs the fairness-performance trade-off baseline by connecting the original model (i.e., P_{OM}) and a series of pseudo mitigation models generated by model behavior mutation (i.e., P_{10}, ..., P_{100}). The pseudo mitigation models are mutated based on the original model by sacrificing ML performance to reduce bias in a naive way. Specifically, Fairea randomly chooses a subset of the predictions made by the original model and replaces them with the majority class of the data. It considers different mutation degrees (i.e., the fraction of chosen predictions) from 10% to 100%, with a step size of 10%, to obtain P_{10}, ..., P_{100}. The core insight of Fairea is that when the original model is mutated into a model that always predicts the same class, fairness will be greatly improved as predictive performance is equally worse in the privileged and unprivileged groups. The fairness-performance trade-off of these naive mutated models is expected to be surpassed by any reasonable bias mitigation method, so we use these models as the baseline.

Step 2: Divide effectiveness regions. The obtained baseline categorizes bias mitigation methods into five regions that represent different effectiveness levels. As illustrated in Fig. 2(b), the **win-win region** contains bias mitigation methods that improve ML performance and decrease bias, while the **lose-lose region** contains methods that decrease ML performance and increase bias. Methods that improve both ML performance and bias fall in the **inverted trade-off region**. The remaining two regions contain methods that reduce bias but decrease ML performance. Specifically, if a method achieves a better trade-off than the baseline constructed by Fairea, it falls within the **good trade-off region**. Otherwise, it belongs to the **poor trade-off region**. The region division of Fairea provides an overview of the overall effectiveness of a bias mitigation method.

Step 3: Quantify trade-off effectiveness. Fairea can also measure the trade-off effectiveness of a bias mitigation method in a quantitative way. Fig. 2(a) shows the area (indicated in grey) obtained by connecting the point of the model after applying a bias mitigation method (i.e., *BM*) and the Fairea baseline vertically and horizontally. Fairea calculates the size of the area as a quantitative

[2]The figure is taken from the original paper [50].

measure of the fairness-performance trade-off achieved by the bias mitigation method. A larger area indicates a better trade-off. Using the area as a measure of the trade-off enables a convenient comparison among different bias mitigation methods.

In the original paper [50], Fairea evaluates only two types of fairness-performance trade-offs (i.e., SPD&Acc and AOD&Acc) on 12 bias mitigation methods proposed in the ML community. In this study, we aim to extend our evaluation to the trade-off between more fairness and performance metrics on 17 bias mitigation methods from the ML and SE communities. Since we employ 11 ML performance metrics and 4 fairness metrics, we have a total of 48 fairness-performance metric pairs. However, as Fairea replaces predictions with the majority class, in terms of ML performance metrics for a certain class (e.g., recall on the majority class), we may observe that ML performance and fairness are both improved with the increased mutation degrees. For such metrics, we cannot obtain the trade-off baselines as in Fig. 2(a). Therefore, here we choose only 5 ML performance metrics that measure performance on both the favorable and unfavorable classes, i.e., Acc, Mac-P, Mac-R, Mac-F1, and MCC, to reflect the overall performance of each bias mitigation method. As a result, we have 20 types of fairness-performance trade-offs, i.e., combinations of 5 ML performance metrics and 4 fairness metrics.

3.5 Experimental Settings

To ensure the reproducibility of our study, we describe the experimental settings in details.

Implementation of datasets: We use the five benchmark datasets implemented in the IBM AIF360 via directly invoking off-the-shelf APIs provided by it [9]. Moreover, we follow previous work [30, 31, 50, 77] to normalize all feature values to be between 0 and 1.

Implementation of bias mitigation: For each bias mitigation task, we train original models using three traditional ML algorithms and four Deep Neural Networks (DNNs):

- For traditional ML algorithms: We use Logistic Regression (LR), Support Vector Machine (SVM), and Random Forest (RF), all of which are widely adopted in previous studies [23, 50, 77]. In line with these studies, we use the default configuration provided by the scikit-learn library [14] to implement each algorithm.
- For DNNs: We first use a fully-connected neural network composed of five hidden layers, which contain 64, 32, 16, 8, 4 units, respectively. This DNN is widely applied to our benchmark datasets in previous fairness research [80–82]. For a more comprehensive evaluation, we also use another three fully-connected neural networks used in a previous fairness testing study [79]. The hidden layer structures of the three DNNs are [50, 30, 15, 10, 5], [30, 20, 15, 10, 5], and [30, 20, 15, 15, 10], respectively. In line with these previous studies, we use ReLU and Softmax as the activation functions for the hidden layers and the output layer, respectively. In the rest of this paper, we denote the four DNNs as DL1, DL2, DL3, and DL4, respectively.

We apply 17 bias mitigation methods for the original models. Specifically, pre-processing and post-processing methods are applied before and after model training, while in-processing methods are applied during the training process to obtain new models. We implement the 15 bias mitigation methods from the ML community based on the IBM AIF360 framework [9], and implement the two methods from the SE community based on the code released by their authors [7, 8]. Since the IBM AIF360 does not support the OP method for the Bank and Mep datasets, we apply OP only for six tasks. For the remaining 16 methods, we apply each of them for all the eight bias mitigation tasks. Each application is repeated 50 times. Each time, the dataset is shuffled and randomly splitted into 70% training data and 30% test data.

Implementation of Fairea: We use Fairea to create the fairness-performance trade-off baseline for each *(task, model, fairness-performance metric pair)* combination. Specifically, we train the

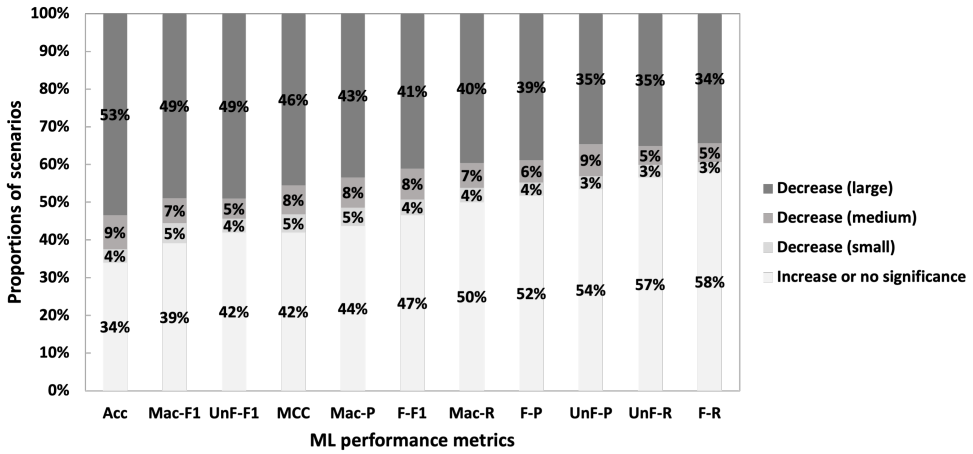


Fig. 3. **RQ1.1:** Effects of bias mitigation methods on different ML performance metrics. After applying existing bias mitigation methods, the values of the 11 ML performance metrics significantly decrease in an average of 53% of applications.

original model 50 times. Each time, based on the original model, we repeat the mutation procedure 50 times for each mutation degree, i.e., 10%, 20%, ..., and 100%. Finally, as suggested by Fairea [50], we construct the baseline using the mean value of the multiple runs.

Statistical analysis: To test whether the difference between two bias mitigation methods is statistically significant, we employ the non-parametric Mann Whitney U-test [57]. This test suits our purpose well as it does not assume normality. The difference is considered significant, only if the p-value of the computed statistic is lower than a pre-specified level (usually 0.05). For example, when we use the Mann Whitney U-test to compare two sets of accuracy values achieved by the 50 runs of methods A and B, the null hypothesis is that the accuracy of A is similar to B. If we find that p-value < 0.05, we can conclude with 95% confidence that our alternative hypothesis is true, which indicates that A achieves a significantly different accuracy than B. By default, we use 95% as the confidence level in this paper.

Furthermore, we compute the effect size with the Cohen's d [40], to check whether the difference has a meaningful effect. We consider the difference with $0 < d < 0.5$ a small effect, $0.5 \leq d < 0.8$ a medium effect, and $d \geq 0.8$ a large effect [67].

In addition, we employ the Spearman's rank correlation coefficient ρ [60] to investigate whether changes in the values of different metrics caused by bias mitigation methods are similar. Spearman's rank correlation coefficient does not assume normality, and thus suits our purpose. The value of ρ is between -1 to 1, where -1 indicates perfectly negative correlation, 0 no correlation, 1 perfectly positive correlation. Moreover, the p-value is reported together with the correlation coefficient. The correlation is considered statistically significant, only if the p-value is lower than 0.05.

Experimental environment: The experiments are implemented with Python 3.7.11 and TensorFlow 1.15.0, and executed on a Ubuntu 16.04 LTS with 128GB RAM, having 2.3 GHz Intel Xeon E5-2653 v3 Dual CPU and two NVidia Tesla M40 GPUs.

4 RESULTS

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

4.1 RQ1: Influence on ML Performance

This section presents the results for 11 ML performance metrics. Based on the results, we investigate RQ1 by answering two specific questions:

RQ1.1 (Effects on ML performance metric values): *How do the values of ML performance metrics change after bias mitigation?* First, we investigate whether the values of ML performance metrics are significantly changed after we apply existing bias mitigation methods. Additionally, we explore whether the changes in different ML performance metrics are significantly correlated. If so, researchers and practitioners can employ the ML performance metrics used in previous work as proxies for the unconsidered ones.

RQ1.2 (ML performance comparison among methods): *How do different bias mitigation methods affect ML performance?* Second, we compare existing bias mitigation methods in terms of different ML performance metrics. The results provide implications for the choice of bias mitigation methods in application scenarios where ML performance is critical.

4.1.1 RQ1.1: Effects on ML Performance Metric Values. To answer RQ1.1, we use the original models that do not apply any bias mitigation method as the baselines. Specifically, for each bias mitigation task, we use LR, SVM, RF, DL1, DL2, DL3, and DL4 to train the original models, with each model repeated 50 times. Then, for each task-model pair, we use the average level (i.e., mean values of ML performance metrics) of the corresponding 50 original models as the baseline. Since we apply each bias mitigation method in each task-model pair, we have a total of $16 \times 8 \times 7 + 1 \times 6 \times 7 = 938$ applications.[3] We repeat each application 50 times. For each application, we calculate the difference in the mean value of each ML performance metric achieved by the 50 models after applying the bias mitigation method and the corresponding 50 original models. Then we analyze the significance and effect size of the difference using Mann Whitney U-test and Cohen's d. In Fig. 3, for each ML performance metric, we present the proportions of applications that fall into different effects, i.e., decreasing ML performance significantly (p-value < 0.05) with a large effect ($d \geq 0.8$), decreasing ML performance significantly with a medium effect ($0.5 \leq d < 0.8$), decreasing ML performance significantly with a small effect ($0 < d < 0.5$) , and increasing ML performance or decreasing ML performance insignificantly (p-value ≥ 0.05).

From Fig. 3, we observe that the values of the 11 ML performance metrics decrease significantly in an average of 53% of applications (ranging from 42% to 66% according to different metrics). This observation is in line with the intuition that lessening the impacts of protected attributes is likely to lessen ML performance. In particular, the values of the 11 metrics significantly decrease with a large effect in an average of 42% of applications (ranging from 34% to 53%). However, as described in Section 1, existing studies often evaluate bias mitigation methods in terms of only one or two ML performance metrics, ignoring some important metrics such as UnF-P and UnF-F1. As a result, researchers and practitioners may be unaware of the significantly decreased performance caused by bias mitigation methods in these unconsidered metrics and thus choose inappropriate methods, making the functional properties of ML software not up to expectations.

Next, we calculate the Spearman's rank correlation coefficient ρ between every pair of ML performance metrics. Specifically, we calculate the value differences of each ML performance metric before and after bias mitigation in the 938 applications. Then, for each two ML performance metrics, we calculate the overall correlation between the 938 value differences of the two metrics. In addition, we also calculate the correlation in the $8 \times 7 = 56$ task-model pairs, separately. Table 2 shows the correlation results. For each metric pair, we present the overall ρ, with * indicating a significant correlation (i.e., p-value < 0.05). Additionally, we list the number of task-model pairs where the

[3]Note that we apply the OP method in six tasks and the other 16 methods in eight tasks, as described in Section 3.5.

Table 2. **RQ1.1:** Correlation between ML performance metrics. In the table, metrics that have been considered by previous work are highlighted in shading; * indicates a significant correlation overall (p-value < 0.05); numbers in parentheses indicate that in how many task-model pairs the correlation shares the consistent pattern with the overall correlation. We observe that the effects of bias mitigation methods on the ML performance metrics newly considered in this paper do not have a consistent correlation with previously employed metrics across all the task-model pairs.

	F-R	F-F1	UnF-P	UnF-R	UnF-F1	Acc	Mac-P	Mac-R	Mac-F1	MCC
F-P	-0.573*(38/56)	-0.139*(6/56)	-0.328*(15/56)	0.719*(54/56)	0.777*(54/56)	0.589*(30/56)	0.732*(39/56)	0.056(0/56)	0.418*(35/56)	0.318*(26/56)
F-R	-	0.727*(51/56)	0.752*(42/56)	-0.888*(56/56)	-0.588*(36/56)	-0.052(27/56)	-0.154*(5/56)	0.446*(30/56)	0.169*(12/56)	0.281*(18/56)
F-F1	-	-	0.608*(47/56)	-0.520*(40/56)	-0.158*(16/56)	0.412*(27/56)	0.264*(22/56)	0.570*(32/56)	0.598*(32/56)	0.672*(34/56)
UnF-P	-	-	-	-0.571*(31/56)	-0.242*(9/56)	0.247*(28/56)	0.221*(28/56)	0.615*(38/56)	0.398*(25/56)	0.519*(36/56)
UnF-R	-	-	-	-	0.831*(53/56)	0.228*(15/56)	0.306*(21/56)	-0.102*(27/56)	0.162*(14/56)	0.056(10/56)
UnF-F1	-	-	-	-	-	0.531*(29/56)	0.523*(33/56)	0.266*(18/56)	0.553*(32/56)	0.453*(19/56)
Acc	-	-	-	-	-	-	0.881*(56/56)	0.267*(19/56)	0.666*(37/56)	0.604*(37/56)
Mac-P	-	-	-	-	-	-	-	0.194*(19/56)	0.558*(31/56)	0.522*(31/56)
Mac-R	-	-	-	-	-	-	-	-	0.798*(49/56)	0.853*(55/56)
Mac-F1	-	-	-	-	-	-	-	-	-	0.959*(56/56)

correlation shares the consistent pattern with the overall correlation in parentheses. For example, the correlation result of F-P and MCC is 0.318*(26/56), indicating that the value differences of F-R and MCC have a significantly positive correlation overall, and the significantly positive correlation holds in 25 out of 56 task-model pairs. Overall, among the 55 metric pairs in Table 2, 41 show a significantly positive correlation, 11 a significantly negative correlation, 3 no significant correlation. Furthermore, we find that 52 metric pairs do not present a consistent correlation pattern across all the task-model pairs. Specifically, only the (Mac-F1, MCC), (Acc, Mac-P), and (F-R, UnF-R) pairs present a consistent correlation pattern. However, not all the metrics are considered in previous software fairness work [23, 24, 30, 31, 50, 77]. Since the metrics newly considered in this work are not consistently correlated with any previously employed metric, we may not use the latter as their proxy. This finding suggests that we take comprehensive ML performance metrics into account during the evaluation of bias mitigation methods, especially considering that different ML performance metrics measure the functional properties of ML software from different aspects and thus may provide different guidelines for real-world application scenarios.

Finding 1: The values of all the 11 ML performance metrics (including those not considered in previous work) decrease significantly in a notable proportion of applications (42%∼66% according to different metrics) after applying existing bias mitigation methods. In particular, these methods significantly decrease accuracy in 66% of scenarios. Moreover, the effects of bias mitigation methods on the ML performance metrics newly considered in this work do not consistently correlated with previously employed metrics, and therefore the latter cannot be used as a proxy.

4.1.2 RQ1.2: ML Performance Comparison among Methods. To compare different bias mitigation methods in terms of ML performance degradation, we first calculate the proportions of scenarios, i.e., *(task-model pair, ML performance metric)* combinations, that fall into different effects after applying each bias mitigation method. Fig. 4 show the results. We present the bias mitigation methods in descending order by the proportion of scenarios with significantly decreased ML performance. Among the bias mitigation methods that we study, RW performs the best in retaining ML performance, while LFR performs the worst. Specifically, RW significantly decreases ML performance in the fewest (27%) scenarios, with a large effect in 10%. The good performance of RW is because it adjusts only the weights of examples in training data and does not modify their features

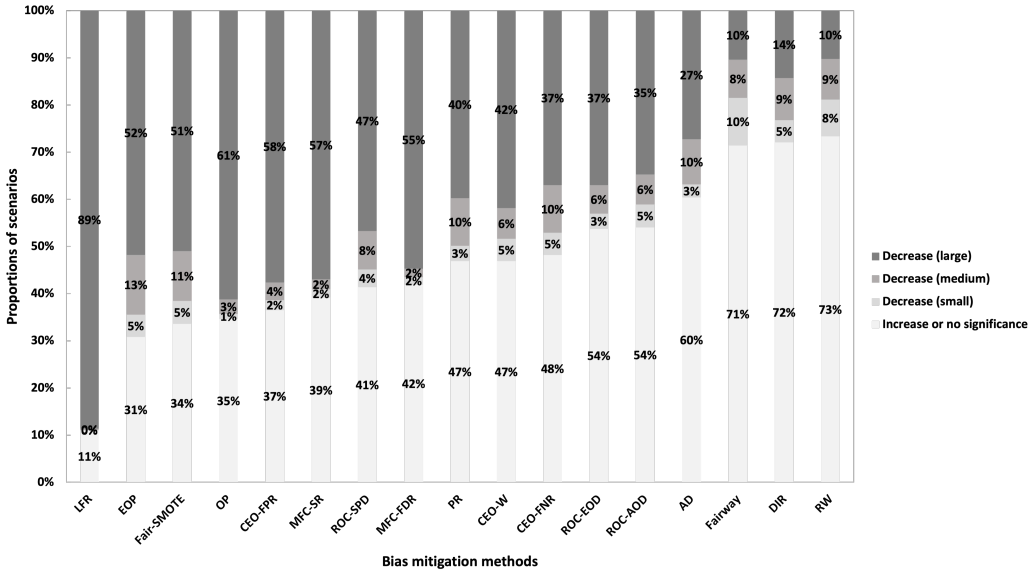


Fig. 4. **RQ1.2:** Effect distribution of different bias mitigation methods in ML performance. We find that 6 out of 17 bias mitigation methods significantly decrease ML performance in less than 50% of scenarios.

Table 3. **RQ1.2:** Mean rank of each bias mitigation method in terms of different ML performance metrics. For each metric, we highlight the top-ranked bias mitigation method (i.e., the method with the smallest rank value) in shading. For any ML performance metric, none of the 17 bias mitigation methods can consistently achieve better results than other methods in all the task-model pairs.

Metric	OP	LFR	RW	DIR	PR	AD	MFC -FDR	MFC -SR	ROC -SPD	ROC -AOD	ROC -EOD	CEO -FNR	CEO -FPR	CEO -W	EOP	Fair -way	Fair -SMOTE
F-P	12	16	6	7	5	6	12	11	7	6	7	9	8	7	9	6	8
F-R	8	9	9	7	9	10	7	7	10	8	8	7	9	9	9	7	10
F-F1	9	11	7	5	9	8	10	12	10	8	7	6	9	9	9	4	9
UnF-P	11	16	8	6	10	8	6	6	9	7	7	7	10	9	9	5	9
UnF-R	11	9	7	9	7	7	10	9	7	8	8	10	8	7	9	8	8
UnF-F1	10	14	6	7	8	6	11	9	8	7	8	9	10	8	9	5	8
Acc	10	14	4	3	7	5	14	13	11	10	11	6	8	6	9	2	9
Mac-P	11	16	5	4	5	5	13	12	10	9	10	7	7	6	10	3	10
Mac-R	11	16	7	6	11	8	8	9	6	3	2	10	12	11	9	6	8
Mac-F1	11	16	6	5	10	6	11	12	8	5	5	9	11	9	9	3	8
MCC	11	16	6	5	10	7	10	11	8	4	3	9	11	9	10	3	8

or labels, avoiding introducing additional noise. In contrast, LFR learns data representations that obfuscate information about protected attributes, and decreases ML performance significantly with a large effect in 89% of scenarios, 79% more than RW. Given the big difference in ML performance of different methods, researchers and practitioners should select their bias mitigation method carefully with ML performance considered. Our comprehensive results provide implications for researchers and practitioners to choose appropriate bias mitigation methods according to different ML performance requirements.

Methods that mitigate bias with ML performance taking into account (e.g., DIR, Fairway, and AD) tend to better retain ML performance. Many of existing methods mitigate bias without considering ML performance, which causes the significant decrease in ML performance. We take LFR, EOP, and Fair-SMOTE as examples, which significantly decrease ML performance in the most scenarios as

shown Fig. 4. They all do not consider ML performance during bias mitigation. Specifically, LFR obfuscates information about protected attributes to mitigate bias; EOP changes output labels to optimize equalized odds; Fair-SMOTE generates new data points to balance samples in different groups to reduce data bias. In contrast, DIR improves fairness while preserving rank-ordering within groups; AD maximizes accuracy and reduces evidence of protected attributes simultaneously; Fairway employs multi-objective optimization to maximize model performance while making it fair. As a result, they minimally damage ML performance (as shown in Fig. 4).

Furthermore, for each ML performance metric, we follow previous work [23] to compute the average rank of each bias mitigation method in the 56 task-model pairs. The smaller the rank value, the less the corresponding method reduces the ML performance metric value. Table 3 shows the results and highlights the top-ranked method for each metric. We observe that the performance drop can be significantly different across different performance metrics. For example, ROC-EOD ranks 2nd for Mac-R among the 17 methods, but it ranks 11th for Acc. This is because different ML performance metrics do not necessarily have a positive correlation and many of them are even negatively correlated (as shown in Table 2). In this case, a bias mitigation method cannot retain its ML performance regarding every metric. Considering the diverse ML performance requirements in real-world applications, this finding suggests researchers take comprehensive metrics for bias mitigation research so as to capture any decrease in performance caused by bias mitigation methods, which further evidences the motivation of our comprehensive study.

Finding 2: Among the 17 bias mitigation methods that we study, RW performs the best in retaining ML performance, while LFR performs the worst. Methods that mitigate bias with ML performance taking into account (e.g., Fairway, DIR, and AD) tend to better retain ML performance. The performance drop can be significantly different across different performance metrics.

4.2 RQ2: Influence on Fairness

This section analyzes the evaluation results for four fairness metrics by answering two specific questions:

RQ2.1 (Effects on fairness metric values): *How do the values of fairness metrics change after bias mitigation?* First, we investigate whether the values of fairness metrics are significantly changed after we apply bias mitigation methods, and whether the changes in different fairness metrics are significantly correlated.

RQ2.2 (Fairness comparison among methods): *How do different bias mitigation methods affect ML software fairness?* Second, we compare existing bias mitigation methods in terms of different fairness metrics. The results provide implications for the choice of bias mitigation methods in application scenarios where fairness is critical.

4.2.1 RQ2.1: Effects on Fairness Metric Values. We follow the procedures in Section 4.1.1 to analyze the value changes of the four fairness metrics caused by bias mitigation methods, and present the results in Fig. 5. We observe that after applying existing bias mitigation methods, the values of the four fairness metrics (i.e., SPD, EOD, AOD, and ERD) significantly decrease (i.e., fairness is improved) in 59%, 52%, 50%, and 24% of applications, with an average of 46%. Considering that the main goal of bias mitigation methods is to improve software fairness (i.e, decrease the values of fairness metrics), our results warn the research community about the limitations of existing bias mitigation methods, especially in reducing ERD.

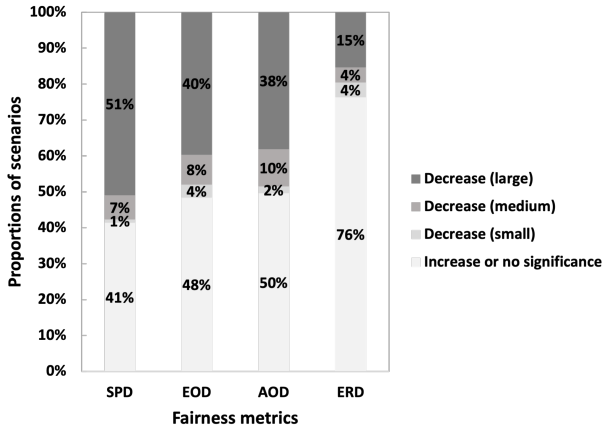


Fig. 5. **RQ2.1:** Effects of bias mitigation methods on different fairness metrics. After applying existing bias mitigation methods, the values of the four fairness metrics significantly decrease (i.e., fairness is improved) in an average of 46% of applications.

Furthermore, similar to Section 4.1.1, we calculate the correlation between each two fairness metrics in terms of their value differences before and after bias mitigation. Table 4 shows the results. We observe that the correlation coefficient between AOD and EOD is 0.906 at a significant level, and the significantly positive correlation holds in all the 56 task-model pairs. This suggests that bias mitigation methods that perform well in AOD may also perform well in EOD, and that the evaluation results of bias mitigation methods using AOD may also provide guidance for method selection in application scenarios that pursue a low EOD. Therefore, it is not surprising to find that some previous work [50] employ only one of them for evaluation. In addition, SPD and AOD have a significantly positive correlation in 55 out of the 56 task-model pairs, and the overall correlation coefficient between them is 0.865.

In other fairness metric pairs, we find that the correlations vary in different task-model pairs, which is consistent with the finding in previous work [23] that analyzes the metric correlation in only two bias mitigation tasks. For instance, we find that SPD and EOD have a significantly positive correlation in 34 out of the 56 task-model pairs.

Additionally, we observe that ERD has a negative correlation with other fairness metrics overall. This indicates that bias mitigation methods that decrease the ERD value often increase bias in terms of other three fairness metrics. Moreover, ERD changes do not have a consistent correlation with changes of any other metric across the task-model pairs. This may be attributed to the basis of these fairness metrics. Specifically, SPD, EOD, and AOD are calculated based on the positives (i.e., all positives for SPD and AOD, and true positives for EOD), while ERD relies on false negatives and false positives. This indicates that other metrics may not act as a proxy of ERD. However, some existing work [30, 31, 50, 78] does not take ERD into account. To perform a comprehensive evaluation, we suggest that researchers follow Biswas and Rajan [23, 24] to consider this metric in future work.

Table 4. **RQ2.1:** Correlation between fairness metrics. In the table, * indicates a significant correlation (p-value < 0.05) overall, and numbers in parentheses indicate the number of task-model pairs the correlation shares the same pattern with the overall correlation. We find that AOD and EOD have a consistently positive correlation across all the task-model pairs, and that ERD does not have a consistent correlation with any other metric.

	AOD	EOD	ERD
SPD	0.865*(55/56)	0.688*(34/56)	-0.154*(10/56)
AOD	-	0.906*(56/56)	-0.218*(8/56)
EOD	-	-	-0.263*(16/56)

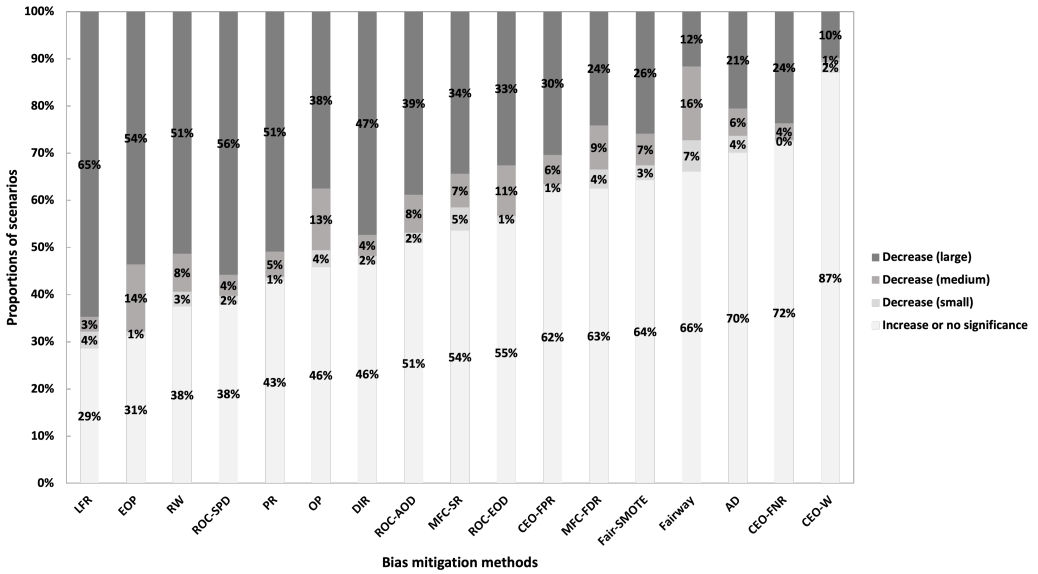


Fig. 6. **RQ2.2:** Effect distribution of different bias mitigation methods in fairness. We observe that 7 out of 17 bias mitigation methods significantly reduce bias in more than 50% of scenarios.

Finding 3: Existing bias mitigation methods improve fairness significantly (in terms of SPD, AOD, EOD, and ERD) in an average of 46% of the studied applications. In particular, existing methods significantly improve fairness measured by ERD in 24% of scenarios. Additionally, fairness improvement measured by different metrics are not necessarily correlated. In particular, ERD changes do not have a consistent correlation with the changes of any other fairness metric.

4.2.2 RQ2.2: Fairness Comparison among Methods. Similar to Section 4.1.2, we calculate the proportions of scenarios, i.e., *(task-model pair, fairness metric)* combinations, that fall into different effects after applying each bias mitigation method. Fig. 6 shows the results. We present bias mitigation methods in descending order by the proportion of scenarios with significantly improved fairness. We find that LFR significantly improves fairness in the most scenarios (71%) among the 17 bias mitigation methods. As described in Section 4.1.2, it also significantly decreases ML performance in the most scenarios, further providing empirical evidence for the existence of fairness-performance

Table 5. **RQ2.2:** Mean rank of each bias mitigation method in terms of different fairness metrics. For each metric, we highlight the top-ranked bias mitigation method (i.e., the method with the smallest rank value) in shading. It is difficult for bias mitigation methods to achieve fairness with respect to all the metrics.

Metric	OP	LFR	RW	DIR	PR	AD	MFC -FDR	MFC -SR	ROC -SPD	ROC -AOD	ROC -EOD	CEO -FNR	CEO -FPR	CEO -W	EOP	Fair -way	Fair -SMOTE
SPD	9	2	5	8	6	10	10	7	4	10	11	11	11	14	6	10	10
AOD	9	2	5	6	8	13	8	7	7	8	9	10	12	15	4	9	10
EOD	9	3	7	7	8	13	7	4	8	6	7	11	12	14	5	10	10
ERD	8	15	7	8	13	8	13	11	5	6	8	8	8	5	5	7	8

trade-off. In contrast, CEO-W significantly improves fairness in the fewest (23%) scenarios, 48% fewer than LFR. Moreover, although fairness improvement is the main goal of bias mitigation methods, only 7 out of the 17 bias mitigation methods (i.e., LFR, EOP, RW, ROC-SPD, PR, OP, and DIR) can significantly improve fairness in more than 50% of scenarios. Therefore, a community effort is required to bring software fairness improvement to a level where it becomes more effective and usable in practice.

Methods designed for optimizing specific metrics (e.g., CEO-W, CEO-FNR, MFC-FDR, and CEO-FPR) tend to have poor overall fairness. For example, CEO-FNR aims at minimizing the false negative rate difference across population groups, and thus the ML model is optimized for this goal, causing damage to other goals. In contrast, the LFR method, which reduces the information of protected attributes rather than optimizing for a specific fairness goal, achieves the best fairness over all the metrics.

Next, we calculate the mean rank of each bias mitigation method in the 56 task-model pairs for each fairness metric. Table 5 shows the results. Different fairness metrics yield significantly different ranking results of the bias mitigation effectiveness. For example, LFR is the top-ranked method for SPD, but it ranks 15th among the 17 methods for ERD. It is because that SPD often has a negative correlation with ERD (as shown in Table 4). This further evidences that it is difficult for bias mitigation methods to achieve fairness with respect to all the metrics [22, 23, 39]. It suggests that besides the fairness-performance trade-off, researchers also need to take into account the trade-off between different fairness metrics when designing bias mitigation methods, especially considering the diverse fairness requirements in real-world applications.

Finding 4: LFR significantly improves fairness in the most scenarios (71%) among the 17 bias mitigation methods that we study. Moreover, only 7 of the 17 methods significantly improve fairness in more than half of the scenarios. In particular, methods designed for optimizing specific metrics (e.g., CEO-W, CEO-FNR, MFC-FDR, and CEO-FPR) tend to have poor overall fairness. Different fairness metrics yield significantly different ranking results of the bias mitigation effectiveness. For example, LFR is the top-ranked method for SPD, but it ranks 15th among the 17 methods for ERD.

4.3 RQ3: Fairness-performance Trade-off

In this section, we present the measurement results of 20 types of fairness-performance trade-offs, i.e., combinations of four fairness metrics (SPD, AOD, EOD, and ERD) and five ML performance metrics (Acc, Mac-P, Mac-R, Mac-F1, and MCC), achieved by different bias mitigation methods using Fairea [50].

As described in Section 3.4.3, the first step of Fairea is to construct the trade-off baseline using a series of pseudo models generated via model behavior mutation. For each *(task-model pair, fairness-performance metric pair)* combination, we construct the baseline separately. As a result, we construct

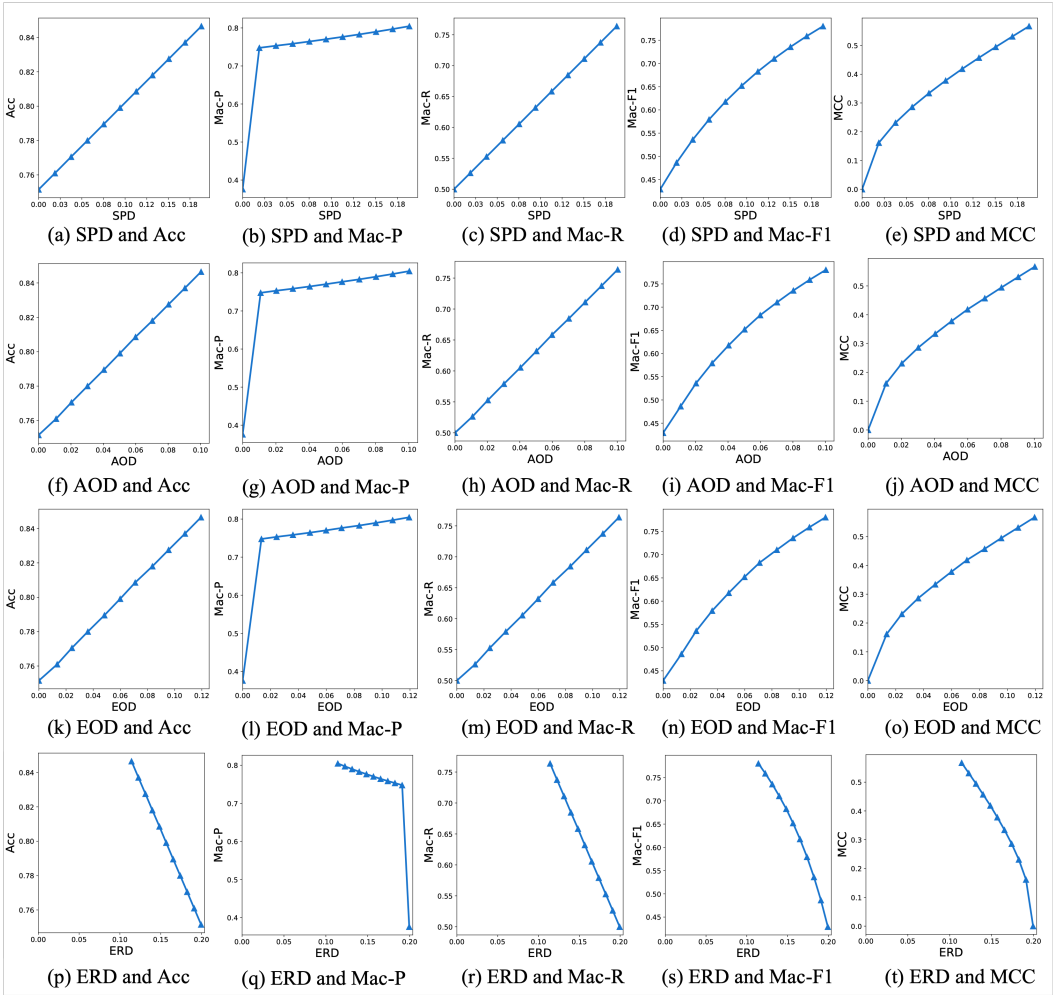


Fig. 7. **RQ3:** Fairness-performance trade-off baselines for LR model in Adult-Sex task. ERD shows a different trade-off pattern from other fairness metrics in the baselines constructed by Fairea.

a total of $8 \times 7 \times 4 \times 5 = 1,120$ baselines. Based on the 1,120 baselines, we observe that the pseudo models show a fairness-performance trade-off for SPD, AOD, and EOD, i.e., the higher the value of SPD, AOD, or EOD, the higher the value of each ML performance metric. However, we fail to construct such trade-off baselines for ERD. We take the baselines constructed for the LR model in the Adult-Sex task (shown in Fig. 7) as an example. Based on the generated pseudo models, ERD shows a different trade-off pattern from SPD, AOD, and EOD. The different trade-off patterns shown by ERD and other fairness metrics can be explained by their different definitions (i.e., calculation methods). Based on their calculation methods presented in Section 3.4.2, we can find that when all samples are predicted as the same label (i.e., the model achieves the worst ML performance), SPD, AOD, and EOD all equal to 0 (their minimum), but ERD equals to the difference of favorable rates between the privileged and unprivileged groups (not its minimum). This means that ERD does not meet the hypothesis behind Fairea. The baseline in Fairea is constructed based on the

hypothesis that the fairness of an ML model can be improved by sacrificing its performance with the increased model behavior mutation degrees. However, from Fig. 7(p)~(t), we observe that the increased mutation degrees not only result in the sacrifice of ML performance, but also cause the increased ERD value (i.e., decreased fairness in terms of ERD).

Since Fairea fails to construct the trade-off baseline for ERD, the effectiveness region division method and the trade-off quantification method provided by Fairea are also not applicable to ERD. Therefore, we consider 15 types of fairness-performance trade-offs, i.e., combinations of three fairness metrics (SPD, AOD, and EOD) and five ML performance metrics (Acc, Mac-P, Mac-R, Mac-F1, and MCC), and the corresponding $8 \times 7 \times 3 \times 5 = 840$ baselines, in the rest of the paper.

Based on the measurement results of the 15 trade-offs, we explore RQ3 by answering two specific questions:

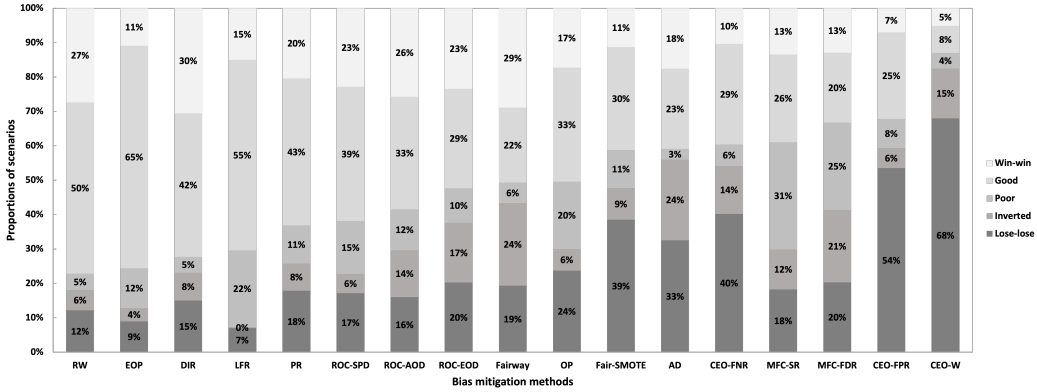
RQ3.1 (Effectiveness region distribution): *What effectiveness regions do existing bias mitigation methods fall into according to Fairea?* This research question evaluates the overall effectiveness of existing bias mitigation methods by analyzing how the mitigation cases achieved by them are matched into the five effectiveness regions in Fig. 2(b).

RQ3.2 (Quantitative assessment of trade-off): *What fairness-performance trade-off do existing bias mitigation methods achieved according to the quantitative assessment of Fairea?* This research question evaluates existing bias mitigation methods in terms of quantitative assessment of fairness-performance trade-off. Specifically, we calculate the area shown in Fig. 2(a) for each bias mitigation method and then compare these methods quantitatively.

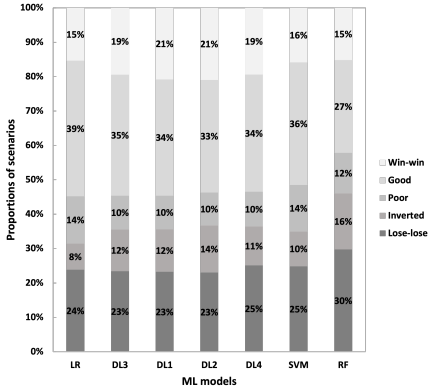
4.3.1 RQ3.1: Effectiveness Region Distribution. We use the 840 baselines to evaluate the effectiveness of 17 bias mitigation methods in 56 task-model pairs. We apply each bias mitigation method to each task-model pair 50 times and treat each individual run as a mitigation case. Therefore, we have $8 \times 7 \times 50 = 2,800$ mitigation cases for each bias mitigation method.[4] In Fig. 8(a), we present the overall results of different bias mitigation methods for all the task-model pairs and fairness-performance metric pairs.

From Fig. 8(a), we find that RW achieves a win-win or good trade-off in the most scenarios (77%), while CEO-W does so in the fewest scenarios (13%). From this perspective, RW makes the best trade-off in general, achieving a win-win trade-off in 27% (ranging from 23% to 34% according to different fairness-performance metric pairs) and a good trade-off in 50% of cases (ranging from 38% to 63%). A possible reason behind the superiority of RW is that it modifies the weights of different training samples so that it tackles the training data imbalance problem, which has been recognized as a root cause of bias in ML software [30, 58]. Meanwhile, it does not modify features or labels of training data, which avoids introducing additional noise and thus retains ML performance. Nevertheless, we still observe that 12% of mitigation cases fall into a lose-lose trade-off after applying RW, and the situation is even worse for other bias mitigation methods. For example, CEO-W achieves a lose-lose trade-off in 68% of cases. Overall, the proportion of cases that fall into the lose-lose region obtained by all the studied methods is 25%. The notable proportion of the lose-lose trade-off is also observed in previous work [50], and our results increase the confidence of this finding with more bias mitigation methods and fairness-performance metric pairs. One possible reason behind the high percentage of the lose-lose trade-off is that bias mitigation methods are often designed to optimize one fairness metric, which may affect other fairness metrics. For instance, ROC-AOD achieves a lose-lose trade-off between AOD and Acc in 17.3% of mitigation cases, while achieves a lose-lose trade-off between SPD and Acc in 32.0% of mitigation cases; ROC-SPD achieves a lose-lose trade-off between SPD and Acc in 1.3% of mitigation cases, while achieves a lose-lose trade-off between AOD and Acc in 22.6% of mitigation cases.

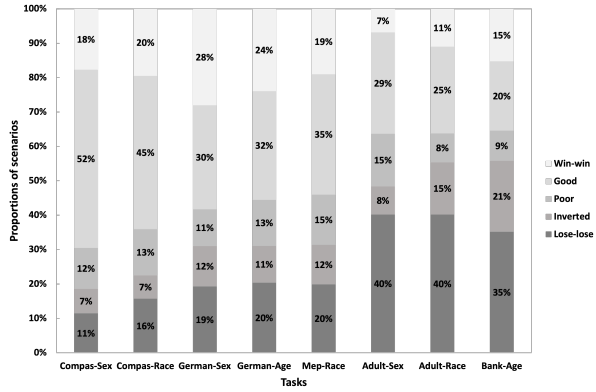
[4]For the OP method, we have $6 \times 7 \times 50 = 2,100$ mitigation cases, as we apply it in only six tasks.



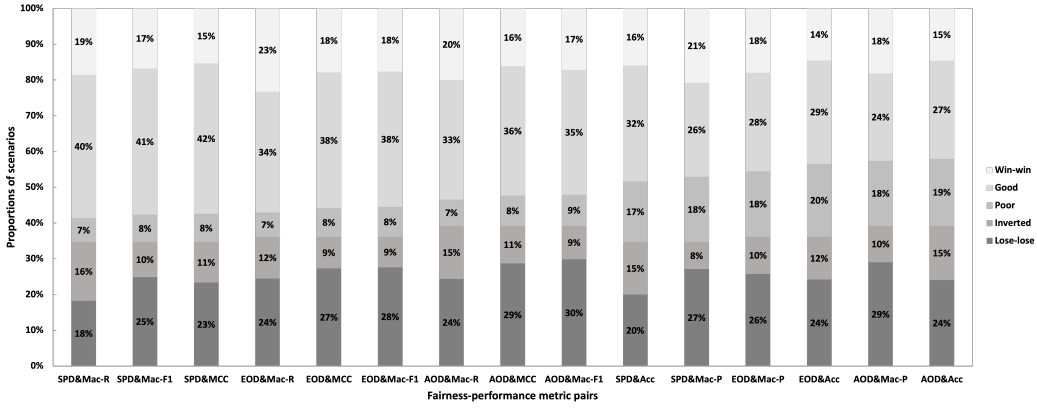
(a) Region distribution of different bias mitigation methods



(b) Region distribution of different models



(c) Region distribution of different bias mitigation tasks



(d) Region distribution of different fairness-performance metric pairs

Fig. 8. **RQ3.1:** Proportion of mitigation cases that fall into each mitigation region organized by (a) bias mitigation methods, (b) models, (c) bias mitigation tasks, and (d) fairness-performance metric pairs. We observe that a notable proportion (25%) of mitigation cases fall into a lose-lose trade-off region, and that the effectiveness of bias mitigation methods depends on models, tasks, the selection of protected attributes, and the set of metrics used to assess fairness and ML performance.

Furthermore, we organize the results by different models, bias mitigation tasks, and fairness-performance metric pairs, to analyze whether the region distribution is influenced by these factors. We find that the effectiveness of bias mitigation methods depends on models, tasks, the selection of protected attributes, and the set of metrics used to assess fairness and ML performance. It is because the different characteristics of different models, tasks, protected attributes, and metrics pose different degrees of difficulty for bias mitigation. The results are presented in Figs. 8(b), (c), and (d), respectively, and the analysis is as follows.

Comparison among models: From Fig. 8(b), we observe that the region distribution is model-dependent. Although LR, SVM, RF, DL1, DL2, DL3, and DL4 share similar proportions of mitigation cases that fall into the lose-lose, poor, and win-win trade-off regions, they have obvious differences in the proportions of the inverted and good trade-offs. Specifically, LR, SVM, DL1, DL2, DL3, and DL4 achieve a relatively higher percentage of the good trade-off (a lower percentage of the inverted trade-off) than RF. For example, the difference between LR and RF in the proportion of the good trade-off is 12% (i.e., 39% v.s. 27%). Therefore, we suggest that researchers consider different models when evaluating the effectiveness of bias mitigation methods.

Comparison among tasks: From Fig. 8(c), we find that the region distribution is task-dependent. Existing methods achieve worse fairness-performance trade-off on imbalanced datasets than on balanced datasets. For example, the Compas dataset is the most balanced among all the datasets, and thus the classification on it is easier compared to other tasks. Therefore, as shown in Fig. 8(c), we can observe that existing bias mitigation methods can retain ML performance well while mitigating bias (i.e., having a good fairness-performance trade-off). In contrast, existing methods achieve the worst trade-off in the most imbalanced dataset (i.e., the Bank dataset), where the major label accounts for 82.8%. Furthermore, we observe that even for the same dataset, the selection of different protected attributes also affects the region distribution. For example, the proportion of the poor trade-off in the Adult-Sex is 15%, nearly twice the corresponding proportion in Adult-Race task (8%). This finding suggests that (1) researchers evaluate bias mitigation methods in diverse tasks to improve the generalizability of the results, (2) practitioners need to be careful when choosing bias mitigation methods for their own tasks based on existing evaluation results on other tasks, and (3) practitioners may need to choose different bias mitigation methods for different protected attributes of the same dataset.

Comparison among fairness-performance metric pairs: From Fig. 8(d), we find that different fairness-performance metric pairs result in different region distributions. In particular, the mitigation cases that obtain a win-win or good trade-off between AOD and Acc account for 42%, the lowest among all the fairness-performance metric pairs. In contrast, the corresponding proportion between SPD and Mac-R is 12% higher (i.e., 59%). This finding further demonstrates the necessity of this study that measures the trade-off in terms of various fairness-performance metric pairs to obtain more general and comprehensive results.

Finding 5: According to the trade-off classification framework provided by Fairea, RW has the best fairness-performance trade-off (with a win-win or good trade-off in 77% of the scenarios). Furthermore, on average, existing methods damage both fairness and ML performance (i.e., a lose-lose trade-off) in 25% of the studied scenarios. The effectiveness of bias mitigation methods depends on models, tasks, the selection of protected attributes, and the set of metrics used to assess fairness and ML performance. In particular, existing methods achieve worse fairness-performance trade-off on imbalanced datasets than on balanced datasets.

Table 6. **RQ3.2:** Numbers of scenarios where each method achieves the best quantitative result, organized by different bias mitigation tasks. For each task, we highlight the bias mitigation method that achieves the best trade-off in the most scenarios in shading. Overall, none of the existing bias mitigation methods can achieve a better trade-off than other methods in all the scenarios, as even the best method that we find outperforms other methods in 30% of scenarios.

Task	OP	LFR	RW	DIR	PR	AD	MFC -FDR	MFC -SR	ROC -SPD	ROC -AOD	ROC -EOD	CEO -FNR	CEO -FPR	CEO -W	EOP	Fair -way	Fair -SMOTE
Adult-Sex	0%	0%	0%	11%	20%	15%	0%	0%	6%	20%	4%	3%	0%	0%	13%	15%	14%
Adult-Race	0%	0%	7%	46%	0%	23%	5%	2%	17%	15%	6%	0%	0%	0%	4%	9%	7%
Compas-Sex	77%	0%	4%	50%	14%	0%	0%	0%	3%	0%	0%	0%	0%	0%	0%	20%	0%
Compas-Race	48%	8%	13%	34%	54%	63%	0%	0%	3%	8%	0%	0%	0%	0%	0%	66%	0%
German-Sex	0%	34%	0%	32%	17%	0%	41%	63%	36%	51%	50%	0%	5%	5%	0%	61%	7%
German-Age	0%	34%	28%	34%	0%	0%	0%	67%	28%	48%	45%	0%	11%	0%	0%	25%	8%
Bank-Age	-	3%	43%	0%	7%	0%	2%	7%	17%	5%	15%	57%	0%	0%	15%	22%	2%
Mep-Race	-	0%	9%	14%	32%	28%	16%	0%	24%	39%	36%	0%	0%	0%	0%	18%	8%
Overall	20%	10%	13%	27%	18%	16%	8%	17%	17%	23%	19%	7%	2%	0%	4%	30%	6%

4.3.2 RQ3.2: Quantitative Assessment of Trade-off. We finally use Fairea to quantify the fairness-performance trade-off of different bias mitigation methods. Fairea quantifies only the cases that fall into the good trade-off region, as the other regions are either dominating the original model (the win-win region), dominated by the Fairea baseline (the poor trade-off region), or do not improve fairness (the inverted and lose-lose trade-off regions) [50].

For each mitigation scenario, i.e., *(task-model pair, fairness-performance metric pair)* combination, we calculate the mean ML performance and fairness results of the 50 runs of each bias mitigation method to indicate its average effectiveness. Then we quantify the trade-off effectiveness of each method with the help of the trade-off baselines. The trade-off quantification enables a direct and convenient comparison among different bias mitigation methods. We consider win-win > good > poor > inverted > lose-lose trade-off (">" means "better than"), and among good trade-offs, the larger the quantitative result, the better the trade-off achieved by the corresponding bias mitigation method. According to this rule, for each bias mitigation task, we calculate the proportion of scenarios where each bias mitigation method achieves the best quantitative result. Table 6 shows the result for each task and the overall result in all the tasks. We highlight the highest proportion for each row (i.e., each bias mitigation task). Note that the sum of the numbers in a row is larger than 100%, since there may be multiple methods to achieve the best quantitative result in a scenario.

At a glance of Table 6, we find that the distributions of the best results vary in different bias mitigation tasks, further demonstrating the aforementioned finding in Section 4.3.1 that the effectiveness of bias mitigation methods is task-dependent. For instance, OP achieves the best quantitative result in 77% of scenarios in the Compas-Sex task, but the corresponding proportion is 0% in the Adult-Sex, Adult-Race, German-Sex, and German-Age tasks.

Furthermore, we compare the overall quantitative results achieved by different bias mitigation methods. From this perspective, the top five methods are Fairway (achieving the best quantitative result in 30% of all the scenarios), DIR (27%), ROC-AOD (23%), OP (20%), and ROC-EOD (19%). Overall, even the best method (i.e., Fairway) that we observe outperforms other methods in 30% of the scenarios. This can be explained by our previous analysis for ML performance and fairness, which demonstrates that no existing methods can consistently have better ML performance or fairness than other methods. This finding suggests practitioners need to choose the bias mitigation method best suited to their intended application scenario(s).

Researchers can also design more generally effective bias mitigation methods. The design of Fairway provides implications for it. Specifically, different from other methods, Fairway combines two different types of bias mitigation methods, i.e., pre-processing and in-processing methods. Similarly, in other software engineering tasks, researchers also find that combining diverse models

is able to tackle more "corner cases" instances, thereby achieving good results [59]. In the future, researchers can propose effective methods that combine bias mitigation strategies across the phases before, in, and after training to achieve better fairness-performance trade-off. In addition, researchers can also design a framework that adaptively provides different appropriate bias mitigation strategies for different scenarios.

Finding 6: According to the quantitative assessment provided by Fairea, the best method that we observe (i.e., Fairway) outperforms other methods in 30% of the scenarios. There is no bias mitigation method that can achieve the best trade-off in all the scenarios that we study. This indicates that practitioners need to carefully choose the bias mitigation method that best suits their intended application scenario(s).

5 THREATS TO VALIDITY

Threats to construct validity concern how adequate a concept definition is and how well the indicators represent the concept. The primary threat lies in the definition of ML performance and fairness. To mitigate the threat, we use 11 ML performance metrics and 4 fairness metrics. The 11 ML performance metrics measure not only the performance of individual classes, but also the overall performance; the 4 fairness metrics are the most widely adopted in previous fairness work.
Threats to internal validity primarily lie in the implementation of the code used in this study. To mitigate the threats, we conduct the experiments based on the widely-adopted IBM AIF360 framework and the released code of Fairway, Fair-SMOTE, and Fairea. We also make all scripts and data publicly available to allow for reproduction and replication.
Threats to external validity concern the generalizability of our experimental results. To alleviate the threats, we adopt eight bias mitigation tasks, which cover social, financial, and medical domains, and are well adopted in the fairness literature.

Moreover, we employ 17 representative bias mitigation methods proposed in the ML and SE communities. For each method, we use both traditional ML algorithms and DNNs for implementation, reducing the influence of model selection on the generalization of results.

In terms of evaluation measures, we use 11 ML performance metrics, 4 fairness metrics, and 20 types of fairness-performance trade-offs to obtain comprehensive measurement results. Fairea poses a potential threat to the use of fairness-performance trade-off measurements. As discussed in Section 4.3, Fairea is constructed based on the hypothesis that the fairness of an ML model can be improved by sacrificing its performance with the increased model behavior mutation degrees. In other words, Fairea cannot be applied to metrics (e.g., ERD) that do not meet the hypothesis, and thus we do not use these metrics in Section 4.3.

Nevertheless, with increasing attention on software fairness, researchers are proposing more and more bias mitigation methods, fairness metrics, and datasets. In the future, one could replicate this study with more methods, metrics, measures, and datasets.

It is valuable to evaluate existing bias mitigation methods on real deployed software systems, not just libraries as we have done here. Unfortunately it is not possible to use real software in this paper because of the lack of public availability of real software with fairness issues (i.e., lack of open source code and data). This is because decision-making software systems that demand fairness are often human-related and social-critical. For such systems, training data or code is typically unreleased due to privacy and legal concerns. This can also explain the observations of recent surveys [58, 63] that state-of-the-art work on software fairness also conducts their experiments on widely-adopted datasets and ML models like us, rather than on real software. To combat this issue, we use scikit–learn, which is widely used in industry, including big companies such as J.P.

Morgan, Amazon, and Microsoft [2, 18]. We also choose the three ML models that are the most widely adopted in the real world [66]. For logistic regression and random forest, an official report about bias in algorithmic decision-making from the UK government [13] reveals that when it comes to use cases that demand fairness such as credit scoring decisions, most banks are using simple logistic regression models and random forest models because they have good explainability. For SVM, it has also been widely adopted in critical decision-making applications, such as criminal justice [46] and credit risk prediction [20].

In addition, in practice, software engineers may need to mitigate bias regarding multiple protected attributes. However, most of existing bias mitigation methods do not support dealing with multiple protected attributes at the same time. Therefore, we follow previous work [24, 31, 50, 77] to consider each protected attribute individually for each bias mitigation task, which may pose a threat to the validity of this study. In the future, when more methods that support this functionality are proposed, one could evaluate them with multiple protected attributes considered at the same time.
Threats to conclusion validity primarily concern the use of statistical methods. To mitigate the threats, we use the non-parametric Mann Whitney U-test, Cohen's d, and Spearman's rank correlation, all of which are widely adopted in SE research [21, 25, 70]. They suit our purpose well as they do not assume normality. In addition, to improve the reliability of our statistical analysis, we apply each bias mitigation method on different tasks 50 times.

6 RELATED WORK

Fairness has attracted increasing interest from both the ML research community [58] and the SE research community [50] and also from practitioners as well as researchers. For example, Microsoft publishes the ethical principles of AI [12], stating that ML software must be fair in real-life applications, and creates a research group named FATE [5] to promote software fairness. In addition, researchers, Brun and Meliou [26] set out a vision for SE research approaches to tackle fairness problems, and a recent survey on ML testing [78] classifies fairness as a non-functional software property, surveying SE approaches to tackling it.

The rising attention on fairness inspires the emergence of a series of fairness testing techniques. Themis [45] generates test suites to measure causal discrimination in software. Aeqitas [71] exploits the inherent robustness property of ML models for directing fairness test generation. SG [19] combines symbolic execution and local explainability to generate test inputs for detecting individual discrimination. ADF [81] uses gradient computation and clustering to generate individual discriminatory instances for DNNs. Chen et al. [35] provide a comprehensive survey of existing research on fairness testing.

In addition to testing software fairness, researchers also attempt to improve software fairness. Zhang and Harman [77] explore the factors that affect software fairness, and find that enlarging feature set is a possible way to improve fairness. Chakraborty et al. [31] remove ambiguous data points in training data and then apply multi-objective optimization to train fair ML models. To better improve software fairness, the follow-up work of Chakraborty et al. [30] not only removes ambiguous data points, but also balances the internal distribution of training data. Moreover, IBM launches a software toolkit called AI Fairness 360 (abbreviated as IBM AIF360) [9], which integrates popular fairness improvement methods, including Adversarial Debiasing [76], Reweighting [51], Reject Option Classification [53], Learning Fair Representation [75], etc.

Furthermore, there are some studies that empirically evaluate the effectiveness of different bias mitigation methods. For example, Biswas and Rajan [23, 24] evaluate seven bias mitigation methods on real-world ML models from a crowd-sourced platform and explore the impact of popular pre-processing procedures on ML performance and fairness. Chakraborty et al. [30, 31] compare the bias mitigation methods proposed by them with several methods proposed in the ML community.

Hort et al. [50] propose a model behavior mutation method to quantitatively benchmark and evaluate the fairness-performance trade-off of different bias mitigation methods. However, these evaluations are conducted in terms of limited measurements and methods. In this paper, we present a comprehensive evaluation of existing bias mitigation methods.

7 CONCLUSIONS AND FUTURE WORK

This paper presents a large-scale empirical study evaluating 17 representative bias mitigation methods with 11 ML performance metrics, 4 fairness metrics, and 20 fairness-performance trade-off measurements for 8 widely-adopted benchmark tasks. The results of our comprehensive study reveal a series of findings. In particular, we find that (1) the bias mitigation methods significantly decrease the values of all ML performance metrics in a notable proportion of scenarios (42%∼66% according to different metrics); (2) the bias mitigation methods achieve significant fairness improvement in 46% of scenarios (24%∼59% according to different fairness metrics); (3) the bias mitigation methods even lead to decreases in both fairness and ML performance in 25% of scenarios; (4) the effectiveness of the bias mitigation methods depends on tasks, models, the choice of protected attributes, and the set of metrics used to assess fairness and ML performance; (5) there is no bias mitigation method that can achieve the best trade-off in all the scenarios that we study. The best bias mitigation method that we find outperforms other methods in 30% of scenarios.

In the future, we plan to develop new bias mitigation methods to tackle the weaknesses of existing methods revealed by our study. Moreover, considering that there is no "silver bullet" bias mitigation method, we plan to propose a framework that selects the most suitable bias mitigation method for different intended scenarios. In addition, when more methods, evaluation metrics, and datasets are proposed in the future, we plan to replicate this study.

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] [n.d.]. The Adult Census Income dataset. https://archive.ics.uci.edu/ml/datasets/adult. Retrieved on September 20, 2021.
[2] [n.d.]. Applications of scikit-learn. https://numfocus.org/project/scikit-learn#:~:text=Implementations%20rely%20either%20on%20vectorized,to%20analyzing%20brain%20imaging%20data. Retrieved on November 24, 2022.
[3] [n.d.]. The Bank dataset. https://archive.ics.uci.edu/ml/datasets/Bank+Marketing. Retrieved on September 20, 2021.
[4] [n.d.]. The Compas dataset. https://github.com/propublica/compas-analysis. Retrieved on September 20, 2021.
[5] [n.d.]. FATE: Fairness, Accountability, Transparency, and Ethics in AI. https://www.microsoft.com/en-us/research/theme/fate/. Retrieved on September 20, 2021.
[6] [n.d.]. The German Credit dataset. https://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29. Retrieved on September 20, 2021.
[7] [n.d.]. The GitHub repository of Fair-SMOTE. https://github.com/joymallyac/Fair-SMOTE/tree/master/Fair-SMOTE. Retrieved on September 20, 2021.
[8] [n.d.]. The GitHub repository of Fairway. https://github.com/joymallyac/Fairway. Retrieved on September 20, 2021.
[9] [n.d.]. IBM AI Fairness 360. https://aif360.mybluemix.net. Retrieved on September 20, 2021.
[10] [n.d.]. Machine Bias. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing. Retrieved on September 20, 2021.
[11] [n.d.]. The Mep dataset. https://meps.ahrq.gov/mepsweb/data_stats/download_data_files_detail.jsp?cboPufNumber=HC-181. Retrieved on September 20, 2021.
[12] [n.d.]. Microsoft AI principles. https://www.microsoft.com/en-us/ai/responsible-ai?activetab=pivot1%3aprimaryr6. Retrieved on September 20, 2021.

[13] [n.d.]. Review into bias in algorithmic decision-making. https://www.gov.uk/government/publications/cdei-publishes-review-into-bias-in-algorithmic-decision-making/main-report-cdei-review-into-bias-in-algorithmic-decision-making. Retrieved on November 24, 2022.
[14] [n.d.]. Scikit-learn. https://scikit-learn.org. Retrieved on September 20, 2021.
[15] [n.d.]. Semantics derived automatically from language corpora contain human-like biases. https://www.science.org/doi/10.1126/science.aal4230. Retrieved on September 20, 2021.
[16] [n.d.]. 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 September 20, 2021.
[17] [n.d.]. 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 September 20, 2021.
[18] [n.d.]. Who is using scikit-learn? https://scikit-learn.org/stable/testimonials/testimonials.html#id8. Retrieved on November 24, 2022.
[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] Kayode Omotosho Alabi, Sulaiman Olaniyi Abdulsalam, Roseline Oluwaseun Ogundokun, and Micheal Olaolu Arowolo. 2020. Credit risk prediction in commercial bank using chi-square with SVM-RBF. In *International Conference on Information and Communication Technology and Applications*. 158–169.
[21] Andrea Arcuri and Lionel C. Briand. 2011. A practical guide for using statistical tests to assess randomized algorithms in software engineering. In *Proceedings of the 33rd International Conference on Software Engineering, ICSE 2011*. 1–10.
[22] 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.
[23] 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.
[24] 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.
[25] Lionel C. Briand, Khaled El Emam, and Sandro Morasca. 1996. On the application of measurement theory in software engineering. *Empirical Software Engineering* 1, 1 (1996), 61–88.
[26] 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.
[27] Toon Calders, Faisal Kamiran, and Mykola Pechenizkiy. 2009. Building classifiers with independency constraints. In *Proceedings of the 2009, IEEE International Conference on Data Mining*. 13–18.
[28] Toon Calders and Sicco Verwer. 2010. Three naive Bayes approaches for discrimination-free classification. *Data Mining and Knowledge Discovery* 21, 2 (2010), 277–292.
[29] L. Elisa Celis, Lingxiao Huang, Vijay Keswani, and Nisheeth K. Vishnoi. 2019. Classification with fairness constraints: a meta-algorithm wit provable guarantees. In *Proceedings of the Conference on Fairness, Accountability, and Transparency, FAT* 2019*. 319–328.
[30] 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.
[31] 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.
[32] 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.
[33] Zhenpeng Chen, Yanbin Cao, Xuan Lu, Qiaozhu Mei, and Xuanzhe Liu. 2019. SEntiMoji: an emoji-powered learning approach for sentiment analysis in software engineering. In *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESECFSE 2019*. 841–852.
[34] 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.
[35] 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).
[36] 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 2022. 1122–1134.
[37] Zhenpeng Chen, Jie M. Zhang, Federica Sarro, and Mark Harman. 2023. Replication package. https://github.com/chenzhenpeng18/TOSEM23-BiasMitigationStudy. Retrieved on January 14, 2023.
[38] 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.
[39] Alexandra Chouldechova. 2017. Fair prediction with disparate impact: a study of bias in recidivism prediction instruments. *Big data* 5, 2 (2017), 153–163.
[40] Jacob Cohen. 2013. *Statistical power analysis for the behavioral sciences*. Academic press.
[41] Flávio du Pin Calmon, Dennis Wei, Bhanukiran Vinzamuri, Karthikeyan Natesan Ramamurthy, and Kush R. Varshney. 2017. Optimized pre-processing for discrimination prevention. In *Proceedings of the Annual Conference on Neural Information Processing Systems 2017, NIPS 2017*. 3992–4001.
[42] Michael Feldman, Sorelle A. Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. 2015. Certifying and removing disparate impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 259–268.
[43] Michael Feldman, Sorelle A. Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. 2015. Certifying and removing disparate impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2015*. 259–268.
[44] Anthony Finkelstein, Mark Harman, Afshin Mansouri, Jian Ren, and Yuanyuan Zhang. 2008. Fairness analysis in requirements assignments. In *Proceedings of the 16th IEEE International Requirements Engineering Conference*. 115–124.
[45] 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.
[46] Mehdi Ghasemi, Daniel Anvari, Mahshid Atapour, J Stephen Wormith, Keira C Stockdale, and Raymond J Spiteri. 2021. The Application of Machine Learning to a General Risk–Need Assessment Instrument in the Prediction of Criminal Recidivism. *Criminal Justice and Behavior* 48, 4 (2021), 518–538.
[47] Moritz Hardt, Eric Price, and Nati Srebro. 2016. Equality of opportunity in supervised learning. In *Proceedings of the Annual Conference on Neural Information Processing Systems 2016, NIPS 2016*. 3315–3323.
[48] 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).
[49] 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*.
[50] 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.
[51] Faisal Kamiran and Toon Calders. 2011. Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems* 33, 1 (2011), 1–33.
[52] 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.
[53] 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.
[54] Faisal Kamiran, Sameen Mansha, Asim Karim, and Xiangliang Zhang. 2018. Exploiting reject option in classification for social discrimination control. *Information Science* 425 (2018), 18–33.
[55] 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.
[56] Andrea Mangani. 2004. Online advertising: Pay-per-view versus pay-per-click. *Journal of Revenue and Pricing Management* 2, 4 (2004), 295–302.
[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] Rebecca Moussa, Giovani Guizzo, and Federica Sarro. 2022. MEG: Multi-objective ensemble generation for software defect prediction. In *Proceedings of the ACM / IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2022*. 159–170.
[60] Jerome L Myers, Arnold D Well, and Robert F Lorch Jr. 2013. *Research design and statistical analysis*. Routledge.
[61] 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.
[62] 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.
[63] D. Pessach and E. Shmueli. 2022. A review on fairness in machine learning. *Comput. Surveys* 55, 3 (2022).
[64] Geoff Pleiss, Manish Raghavan, Felix Wu, Jon M. Kleinberg, and Kilian Q. Weinberger. 2017. On fairness and calibration. In *Proceedings of the Annual Conference on Neural Information Processing Systems 2017, NIPS 2017*. 5680–5689.
[65] 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.
[66] Iqbal H Sarker. 2021. Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science* 2, 3 (2021), 1–21.
[67] Shlomo S Sawilowsky. 2009. New effect size rules of thumb. *Journal of Modern Applied Statistical Methods* 8, 2 (2009), 26.
[68] Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *Comput. Surveys* 34, 1 (2002), 1–47.
[69] Till Speicher, Hoda Heidari, Nina Grgic-Hlaca, Krishna P. Gummadi, Adish Singla, Adrian Weller, and Muhammad Bilal Zafar. 2018. A unified approach to quantifying algorithmic unfairness: measuring individual &group unfairness via inequality indices. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*. 2239–2248.
[70] Richard Torkar, Carlo A. Furia, Robert Feldt, Francisco Gomes de Oliveira Neto, Lucas Gren, Per Lenberg, and Neil A. Ernst. 2022. A Method to Assess and Argue for Practical Significance in Software Engineering. *IEEE Transactions on Software Engineering* 48, 6 (2022), 2053–2065.
[71] 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.
[72] 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.
[73] 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.
[74] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez-Rodriguez, and Krishna P. Gummadi. 2017. Fairness constraints: mechanisms for fair classification. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017*. 962–970.
[75] Richard S. Zemel, Yu Wu, Kevin Swersky, Toniann Pitassi, and Cynthia Dwork. 2013. Learning fair representations. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013*. 325–333.
[76] 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.
[77] 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.
[78] Jie M. Zhang, Mark Harman, Lei Ma, and Yang Liu. 2019. Machine learning testing: survey, landscapes and horizons. *IEEE Transactions on Software Engineering* (2019).
[79] 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.
[80] 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, ESEC/FSE 2022*.
[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] 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 IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022*. 1519–1531.