Do the Machine Learning Models on a Crowd Sourced Platform Exhibit Bias? An Empirical Study on Model Fairness

Sumon Biswas
Dept. of Computer Science, Iowa State University
Ames, IA, USA
sumon@iastate.edu

Hridesh Rajan
Dept. of Computer Science, Iowa State University
Ames, IA, USA
hridesh@iastate.edu

ABSTRACT

Machine learning models are increasingly being used in important decision-making software such as approving bank loans, recommending criminal sentencing, hiring employees, and so on. It is important to ensure the fairness of these models so that no discrimination is made based on *protected attribute* (e.g., race, sex, age) while decision making. Algorithms have been developed to measure unfairness and mitigate them to a certain extent. In this paper, we have focused on the empirical evaluation of fairness and mitigations on real-world machine learning models. We have created a benchmark of 40 top-rated models from Kaggle used for 5 different tasks, and then using a comprehensive set of fairness metrics, evaluated their fairness. Then, we have applied 7 mitigation techniques on these models and analyzed the fairness, mitigation results, and impacts on performance. We have found that some model optimization techniques result in inducing unfairness in the models. On the other hand, although there are some fairness control mechanisms in machine learning libraries, they are not documented. The mitigation algorithm also exhibit common patterns such as mitigation in the post-processing is often costly (in terms of performance) and mitigation in the pre-processing stage is preferred in most cases. We have also presented different trade-off choices of fairness mitigation decisions. Our study suggests future research directions to reduce the gap between theoretical fairness aware algorithms and the software engineering methods to leverage them in practice.

CCS CONCEPTS

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

KEYWORDS

fairness, machine learning, models

ACM Reference Format:
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 European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '20), November 8–13, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3368089.3409704

ESEC/FSE '20, November 8–13, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7043-1/20/11.
https://doi.org/10.1145/3368089.3409704

1 INTRODUCTION

Since machine learning (ML) models are increasingly being used in making important decisions that affect human lives, it is important to ensure that the prediction is not biased toward any protected attribute such as race, sex, age, marital status, etc. ML fairness has been studied for about past 10 years [16], and several fairness metrics and mitigation techniques [8, 11, 15, 20, 34, 36, 50, 52, 52] have been proposed. Many testing strategies have been developed [3, 17, 49] to detect unfairness in software systems. Recently, a few tools have been proposed [2, 4, 44, 48] to enhance fairness of ML classifiers. However, we are not aware how much fairness issues exist in ML models from practice. Do the models exhibit bias? If yes, what are the different bias types and what are the model constructs related to the bias? Also, is there a pattern of fairness measures when different mitigation algorithms are applied? In this paper, we have conducted an empirical study on ML models to understand these characteristics.

Harrison *et al.* studied how ML model fairness is perceived by 502 Mechanical Turk workers [21]. Recently, Holstein *et al.* conducted an empirical study on ML fairness by surveying and interviewing industry practitioners [22]. They outlined the challenges faced by the developers and the support they need to build fair ML systems. They also discussed that it is important to understand the fairness of existing ML models and improve software engineering to achieve fairness. In this paper, we have analyzed the fairness of 40 ML models collected from a crowd sourced platform, Kaggle, and answered the following research questions.

RQ1: (Unfairness) What are the unfairness measures of the ML models in the wild, and which of them are more or less prone to bias?

RQ2: (Bias mitigation) What are the root causes of the bias in ML models, and what kind of techniques can successfully mitigate those bias?

RQ3: (Impact) What are the impacts of applying different bias mitigating techniques on ML models?

First, we have created a benchmark of ML models collected from Kaggle. We have manually verified the models and selected appropriate ones for the analysis. Second, we have designed an experimental setup to measure, achieve, and report fairness of the ML models. Then we have analyzed the result to answer the research questions. The key findings are: model optimization goals are configured towards overall performance improvement, causing unfairness. A few model constructs are directly related to fairness of the model. However, ML libraries do not explicitly mention fairness in documentation. Models with effective pre-processing mitigation algorithm are more reliable and pre-processing mitigations always retain performance. We have also reported different patterns of exhibiting bias and mitigating them. Finally, we have reported the trade-off concerns evident for those models.

The paper is organized as follows: §2 describes the background and necessary terminology used in this paper. In §3, we have described the methodology of creating the benchmark and setting up experiment, and discussed the fairness metrics and mitigation techniques. §4 describes the fairness comparison of the models, §5 describes the mitigation techniques, and §9 describes the impacts of mitigation. We have discussed the threats to validity in §7, described the related work in §8, and concluded in §9.

2 BACKGROUND

The basic idea of ML fairness is that the model should not discriminate between different individuals or groups from the protected attribute class [16, 17]. *Protected attribute* (e.g., race, sex, age, religion) is an input feature, which should not affect the decision making of the models solely. Chen *et al.* listed 12 protected attributes for fairness analysis [10]. One trivial idea is to remove the protected attribute from the dataset and use that as training data. Pedreshi *et al.* showed that due to the redundant encoding of training data, it is possible that protected attribute is propagated to other correlated attributes [39]. Therefore, we need fairness aware algorithms to avoid bias in ML models. In this paper, we have considered both group fairness and individual fairness. *Group fairness* measures whether the model prediction discriminates between different groups in the protected attribute class (e.g., sex: *male/female*) [14]. *Individual fairness* measures whether similar prediction is made for similar individuals those are only different in protected attribute [14]. Based on different definitions of fairness, many group and individual fairness metrics have been proposed. Additionally, many fairness mitigation techniques have been developed to remove unfairness or bias from the model prediction. The fairness metrics and mitigation techniques have been described in the next section.

3 METHODOLOGY

In this section, first, we have described the methodology to create the benchmark of ML models for fairness analysis. Then we have described our experiment design and setup. Finally, we have discussed the fairness metrics we evaluated and mitigation algorithms we applied on each model.

3.1 Benchmark Collection

We have collected ML models from Kaggle kernels [25]. Kaggle is one of the most popular data science (DS) platform owned by Google. Data scientists, researchers, and developers can host or take part in DS competition, share dataset, task, and solution. Many Kaggle solutions resulted in impactful ML algorithms and research such as neural networks used by Geoffrey Hinton and George Dahl [12], improving the search for the Higgs Boson at CERN [23], state-of-the-art HIV research [9], etc. There are 376 competitions and 28,622 datasets in Kaggle to date. The users can submit solutions for the competitions and dataset-specific tasks. To create a benchmark to analyze the fairness of ML models, we have collected 40 kernels from the Kaggle. Each kernel provides solution (code and description) for a specific data science task. In this study, we have analyzed ML models that operate on 1) datasets utilized by prior studies on fairness, and 2) datasets with protected attribute (e.g., sex, race). With this goal, we have collected the ML models with different filtering criteria for each category. The overall process of collecting the benchmark has been depicted in Figure 1.

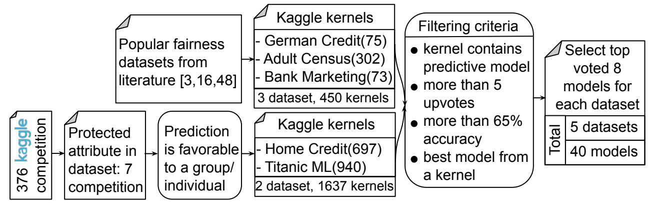

Figure 1: Benchmark model collection process

To identify the datasets used in prior fairness studies, we refer to the work on fairness testing by Galhotra *et al.* [17], where two datasets, German Credit and Adult Census have been used. Udeshi *et al.* experimented on models for the Adult Census dataset [49]. Aggarwal *et al.* used six datasets: German Credit, Adult Census, Bank Marketing, US Executions, Fraud Detection, and Raw Car Rentals) [3]. Among these datasets, German Credit, Adult Census and Bank Marketing dataset are available on Kaggle. From the solutions for these datasets, we have collected 440 kernels (65 for German Credit, 302 for Adult Census, and 73 for Bank Marketing). Furthermore, we have filtered the kernels based on three criteria to select the top-rated ones: 1) contain predictive models (some kernels only contain exploratory data analysis), 2) at least 5 upvotes, and 3) accuracy $\geq$ 65%. Often a kernel contains multiple models and tries to find the best performing one. In these cases, we have selected the best performing model from every kernel. Thus, we have selected the top 8 models based on upvotes for each of the 3 datasets and got 24 ML models.

Chen *et al.* [10] listed 12 protected attributes, e.g., age, sex, race, etc. for fairness analysis. We have found 7 competitions in Kaggle, that contain any of these attributes. From the selected ones, we have filtered out the competitions that involve prediction decisions not being favorable to individuals or a specific group. For example, although this competition [28] has customers *age* and *sex* in the dataset, the classification task is to recommend an appropriate product to the customers, which we can not classify as fair or unfair. Thus, we have got two appropriate competitions with several kernels. To select ML models from these competitions, we have utilized the same filtering criteria used before and selected 8 models for each dataset based on the upvotes. Finally, we have created a benchmark containing 40 top-rated Kaggle models that operate on 5 datasets. The characteristics of the datasets and tasks in the benchmark are shown in Table 1.

3.2 Experiment Design

After creating the benchmark, we have experimented on the models, evaluated performance and fairness metrics, and applied different bias mitigation techniques to observe the impacts. Our experiment design process is shown in Figure 2. The experiments on the benchmark have been peer reviewed and published as an artifact [7].

In our benchmark, we have models from five dataset categories. To be able to compare the fairness of different models in each dataset category, we have used the same data preprocessing strategy. We

Table 1: The datasets used in the fairness experimentation. # F: Feature count. PA: Protected attribute.

Dataset	Size	# F	PA	Description
German Credit [29]	1,000	21	age, sex	This dataset contains personal information about individuals and predicts credit risk (good or bad credit). The *age* protected attribute is categorized into young (< 25) and old (≥ 25) based on [16].
Adult Census [26]	32,561	12	race, sex	This dataset comprises of individual information from the 1994 U.S. census. The target feature of this dataset is to predict whether an individual earns ≥ $50,000 or not in a year.
Bank Marketing [27]	41,188	20	age	This dataset contains the direct marketing campaigns data of a Portuguese bank. The goal is to predict whether a client will subscribe for a term deposit or not.
Home Credit [30]	3,075,11	240	sex	This dataset contains data related to loan applications for individuals who do not get loan from the traditional banks. The target feature is to predict whether an individual who can repay the loan, get the application accepted or not.
Titanic ML [31]	891	10	sex	This dataset contains data about the passengers of Titanic. The target feature is to predict whether the passenger survived the sinking of Titanic or not. The target of the test set is not published. So, we have taken the training data and further split it into train and test.

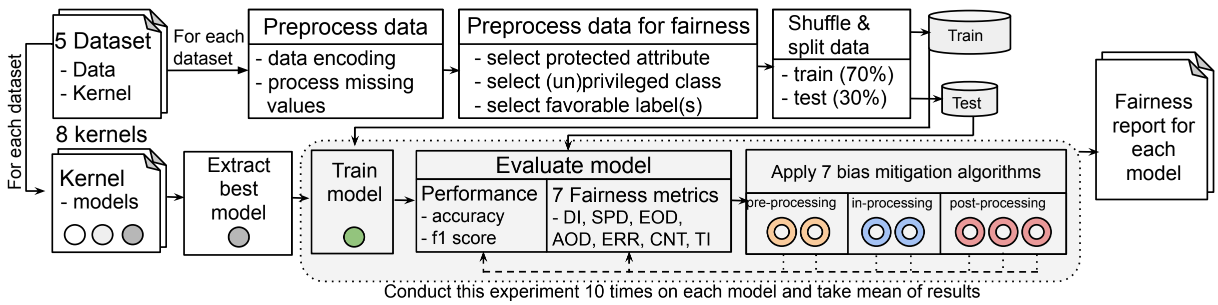

Figure 2: Experimentation to compute performance, fairness and mitigation impacts of machine learning models.

have processed the missing or invalid values, transformed continuous features to categorical (e.g., age<25: young, age≥25: old), and converted non-numerical features to numerical (e.g., *female*: 0, *male*: 1). We have done some further preprocessing to the dataset to be used for fairness analysis: specify the protected attributes, privileged and unprivileged group, and what are the favorable label or outcome of the prediction. For example, in the Home Credit dataset, *sex* is the protected attribute, where *male* is the privileged group, *female* is the unprivileged group, and the prediction label is credit risk of the person i.e., good (favorable label) or bad. For all the datasets, we have used shuffling and same train-test splitting (70%-30%) before feeding the data to the models.

For each dataset category, we have eight Kaggle kernels. The kernels contain solution code written in Python for solving classification problems. In general, the kernels follow these stages: data exploration, preprocessing, feature selection, modeling, training, evaluation, and prediction. From the kernels, we have manually extracted the code for modeling, training, and evaluation. For example, this kernel [33] loads the German Credit dataset, performs exploratory analysis and selects a subset of the features for training, preprocesses data, and finally implements XGBoost classifier for predicting the credit risk of individuals. We have manually sliced the code for modeling, training, and evaluation. Often the kernels try multiple models, evaluate results, and find the best model. From a single kernel, we have only sliced the best performing model found by the kernel. Some kernels do not specify the best model. In this case, we have selected the model with the best accuracy. For example, this kernel [32] works on Adult Census dataset and implements four models (Logistic Regression, Decision Tree, K-Nearest Neighbor and Gradient Boosting) for predicting income of individuals. We have selected the Gradient Boosting classifier model since it gives the best accuracy.

After extracting the best model, we train the model and evaluate performance (accuracy, F1 score). We have found that the model performance in our experiment is consistent with the prediction made in the kernel. Then, we have evaluated 7 different fairness metrics described in §3.3.2. Next, we have applied 7 different bias mitigation algorithms separately and evaluated the performance and fairness metrics. Thus, we collect the result of 9 metrics (2 performance metric, 7 fairness metric) before applying any mitigation algorithm and after applying each mitigation algorithm. For each model, we have done this experiment 10 times and taken the mean of the results as suggested by [16]. We have used the open-source Python library AIF 360 [4] developed by IBM for fairness metrics and bias mitigation algorithms. All experiments have been executed on a MAC OS 10.15.2, having 4.2 GHz Intel Core i7 processor with 32 GB RAM and Python 3.7.6.

3.3 Measures

We have computed the algorithmic fairness of each subject model in our benchmark. Let, $D = (X, Y, Z)$ be a dataset where X is the training data, Y is the binary classification label ($Y = 1$ if the label is favorable, otherwise $Y = 0$), Z is the protected attribute ($Z = 1$ for privileged group, otherwise $Z = 0$), and $\hat{Y}$ is the prediction label

(1 for favorable decision and 0 for unfavorable decision). If there are multiple groups for protected attributes, we have employed a binary grouping strategy (e.g., race attribute in Adult Census dataset has been changed to white/non-white).

3.3.1 Accuracy Measure. Before measuring the fairness of the model, we have computed the performance in terms of accuracy, and F1 score.
Accuracy: Accuracy is given by the ratio of truly classified items and total number of items.

$$\text{Accuracy} = (\#\ \text{True positive} + \#\ \text{True negative})/\#\ \text{Total}$$

F1 Score: This metric is given by the harmonic mean of precision and recall.

$$\text{F1} = 2 * (\text{Precision} * \text{Recall})/(\text{Precision} + \text{Recall})$$

3.3.2 Fairness Measure. Many quantitative fairness metrics have been proposed in the literature [6] based on different definitions of fairness. For example, AIF 360 toolkit has APIs for computing 71 fairness metrics [4]. In this paper, without being exhaustive, a representative list of metrics have been selected to evaluate the fairness of ML models. We have adopted the metrics recommendation of Friedler *et al.* [16] and further added the individual fairness metrics.

Metrics based on base rates:
Disparate Impact (DI): This metric is given by the ratio between the probability of unprivileged group gets favorable prediction and the probability of privileged group gets favorable prediction [15, 50].

$$\text{DI} = \text{P}[\hat{Y} = 1|Z = 0]/\text{P}[\hat{Y} = 1|Z = 1]$$

Statistical Parity Difference (SPD): This measure is similar to DI but instead of the ratio of probabilities, difference is calculated [8].

$$\text{SPD} = \text{P}[\hat{Y} = 1|Z = 0] - \text{P}[\hat{Y} = 1|Z = 1]$$

Metrics based on group conditioned rates:
Equal Opportunity Difference (EOD): This is given by the true-positive rate (TPR) difference between unprivileged and privileged groups.

$$\text{TPR}_u = \text{P}[\hat{Y} = 1|Y = 1, Z = 0]\ ;\ \text{TPR}_p = \text{P}[\hat{Y} = 1|Y = 1, Z = 1]$$
$$\text{EOD} = TPR_u - TPR_p$$

Average Odds Difference (AOD): This is given by the average of false-positive rate (FPR) difference and true-positive rate difference between unprivileged and privileged groups [20].

$$\text{FPR}_u = \text{P}[\hat{Y} = 1|Y = 0, Z = 0]\ ;\ \text{FPR}_p = \text{P}[\hat{Y} = 1|Y = 0, Z = 1]$$
$$\text{AOD} = \frac{1}{2}\{(FPR_u - FPR_p) + (TPR_u - TPR_p)\}$$

Error Rate Difference (ERD): Error rate is given by the addition of false-positive rate (FPR) and false-negative rate (FNR) [11].

$$\text{ERR} = \text{FPR} + \text{FNR}$$
$$\text{ERD} = \text{ERR}_u - \text{ERR}_p$$

Metrics based on individual fairness:
Consistency (CNT): This individual fairness metric measures how similar the predictions are when the instances are similar [51]. Here, *n_neighbors* is the number of neighbors for the KNN algorithm.

$$\text{CNT} = 1 - \frac{1}{n * n\_neighbors}\sum_{i=1}^{n}|\hat{y}_i - \sum_{j \in \mathcal{N}_{n\_neighbors(x_i)}} \hat{y}_j|$$

Theil Index (TI): This metric is also called the entropy index which measures both the group and individual fairness [45]. Theil index is given by the following equation where $b_i = \hat{y}_i - y_i + 1$.

$$\text{TI} = \frac{1}{n}\sum_{i=1}^{n}\frac{b_i}{\mu}\ln\frac{b_i}{\mu}$$

3.4 Bias Mitigation Techniques

In this section, we have discussed the bias mitigation techniques that have been applied to the models. These techniques can be broadly classified into preprocessing, in-processing, and postprocessing approaches.

Preprocessing Algorithms. Preprocessing algorithms do not change the model and only work on the dataset before training so that models can produce fairer predictions.
Reweighing [34]: In a biased dataset, different weights are assigned to reduce the effect of favoritism of a specific group. If a class of input has been favored, then a lower weight is assigned in comparison to the class not been favored.
Disparate Impact Remover [15]: This algorithm is based on the concept of the metric DI that measures the fraction of individuals achieves positive outcomes from an unprivileged group in comparison to the privileged group. To remove the bias, this technique modifies the value of protected attribute to remove distinguishing factors.

In-processing Algorithms. In-processing algorithms modify the ML model to mitigate the bias in the original model prediction.
Adversarial Debiasing [52]: This approach modifies the ML model by introducing backward feedback (negative gradient) for predicting the protected attribute. This is achieved by incorporating an adversarial model that learns the difference between protected and other attributes that can be utilized to mitigate the bias.
Prejudice Remover Regularizer [36]: If an ML model relies on the decision based on the protected attribute, we call that direct prejudice. In order to remove that, one could simply remove the protected attribute or regulate the effect in the ML model. This technique applies the latter approach, where a regularizer is implemented that computes the effect of the protected attribute.

Post-processing Algorithms. This genre of techniques modifies the prediction result instead of the ML models or the input data.
Equalized Odds (E) [20]: This approach changes the output labels to optimize the EOD metric. In this approach, a linear program is solved to obtain the probabilities of modifying prediction.
Calibrated Equalized Odds [41]: To achieve fairness, this technique also optimizes EOD metric by using the calibrated prediction score produced by the classifier.
Reject Option Classification [35]: This technique favors the instances in privileged group over unprivileged ones that lie in the decision boundary with high uncertainty.

4 UNFAIRNESS IN ML MODELS

In this section, we have explored the answer of RQ1 by analyzing different fairness measures exhibited by the ML models in our benchmark. Do the models have bias in their prediction? If so, which models are fairer and which are more biased? What is causing the models to be more prone to bias? What kind of fairness metric is sensitive to different models? To answer these questions, we have conducted experiment on the ML models and computed the fairness metrics. The result is presented in Table 2. The unfairness measures for all the 40 models are depicted in Figure 3. To be able to compare all the metrics in the same chart, disparate impact (DI), and consistency (CNT) have been plotted in the log scale. If the value of a fairness metric is 0, there is no bias in the model according to the corresponding metric. If the measure is less than or greater than 0, bias exists. The negative bias denotes that the prediction is biased towards privileged group and positive bias denotes that prediction is biased towards unprivileged group.

We have found that all the models exhibit unfairness and models specific to a dataset show similar bias patterns. From Figure 3, we can see that all the models exhibit bias with respect to most of the fairness metrics. For a model, metric values vary since the metrics follow different definitions of fairness. Therefore, we have compared bias of different models both cumulatively and using the specific metric individually. To compare total bias across all the metrics, we have taken the absolute value of the measures and computed the sum of bias for each model. In Figure 4, we can see the total bias exhibited by the models. Although the bias exhibited by models for each dataset follow similar pattern, certain models are fairer than others.

Finding 1: Model optimization goals seek overall performance improvement, which is causing unfairness.

Model GC1 exhibits the lowest bias among German Credit models. GC1 is a Random Forest (RFT) classifier model, which is built by using a grid search over a given range of hyperparameters. After the grid search, the best found classifier is:

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
        class_weight=None, criterion='gini', max_depth=3,
        max_features=4, max_leaf_nodes=None, max_samples=None,
        min_impurity_decrease=0.0, min_impurity_split=None,
        min_samples_leaf=1, min_samples_split=2,
        min_weight_fraction_leaf=0.0, n_estimators=25, n_jobs=None,
        oob_score=False, random_state=2, warm_start=False)
```

We have found that GC6 is also a Random Forest classifier built through grid search. However, GC6 is less fair in terms of cumulative bias (Figure 4), and individual metrics (Figure 3) except error rate difference (ERD). We have investigated the reason of the fairness differences in these two models by running both of them by changing one hyperparameter at a time. We have found that the fairness difference is caused by the scoring mechanism used by the two models. GC1 uses `scoring='recall'`, whereas GC6 uses `scoring='precision'`, as shown in the following code snippet.

```
# Model GC1
param_grid = {"max_depth": [3,5, 7, 10,None], "n_estimators"
        :[3,5,10,25,50,150], "max_features": [4,7,15,20]}
GC1 = RandomForestClassifier(random_state=2)
grid_search = GridSearchCV(GC1, param_grid=param_grid, cv=5,
        scoring='recall', verbose=4)
# Model GC6
params = {'n_estimators':[25,50,100,150,200,500],'max_depth'
        :[0.5,1,5,10],'random_state':[1,10,20,42], 'n_jobs':[1,2]}
GC6 = RandomForestClassifier()
grid_search_cv = GridSearchCV(GC6, params,scoring='precision')
```

Further investigation shows, in German Credit dataset, the data rows are personal information about individuals and task is to predict their credit risk. The data items are not balanced when *sex* of the individuals is concerned. The dataset contains 69% data instances of *male* and 31% *female* individuals. When the model is optimized towards recall (GC1) rather than precision (GC6), the total number of true-positives decreases and false-negative increases. Since the number of instances for privileged group (*male*) is more than the unprivileged group (*female*), decrease in the total number of true-positives also increases the probability of unprivileged group to be classified as favorable. Therefore, the fairness of GC1 is more than GC2, although the accuracy is less. Unlike other group fairness metrics, error rate difference (ERD) accounts for false-positive and false-negative rate difference between privileged and unprivileged group. As described before, optimizing the model for recall increases the total number of false-negatives. We have found that the percentage of *male* categorized as favorable is less than the percentage of *female* categorized as favorable. Therefore, an increase in the overall false-negative also increased the error rate of unprivileged group, which in turn caused GC1 to be more biased than GC2 in terms of ERD.

From the above discussion, we have observed that the model optimization hyperparameter only considers the overall rates of the performance. However, if we split the data instances based on protected attribute groups, then we see the change of rates vary for different groups, which induces bias. The libraries for model construction also do not provide any option to specify model optimization goals specific to protected attributes and make fairer prediction.

Here, we have seen that GC1 has less bias than GC6 by compromising little accuracy. Do all the models achieve fairness by compromising with performance? We have found that models can achieve fairness along with high performance. To compare model performance with the amount of bias, we have plotted the accuracy and F1 score of the models with the cumulative bias in Figure 4. We can see that GC6 is the most efficient model in terms of performance and has less bias than 5 out of 7 other models in German Credit data. AC6 has more accuracy and F1 score than any other models in Adult Census, and exhibits less bias than AC1, AC2, AC4, AC5, and AC7. Therefore, models can have better performance and fairness at the same time.

Finding 2: Libraries for model creation do not explicitly mention fairness concerns in model constructs.

From Figure 3, we can see that HC1 and HC2 show difference in most of the fairness metrics, while operating on the same dataset i.e., Home Credit. HC2 is fairer than HC1 with respect to all the metrics except DI. From Table 2, we can see that HC1 has positive bias, whereas HC2 exhibit negative bias. This indicates that HC1 is biased towards unprivileged group and HC2 is biased towards privileged group. We have found that HC1 and HC2 both are using

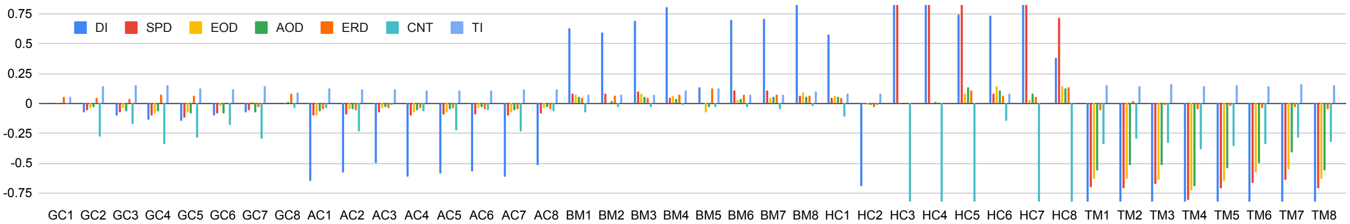

Figure 3: Unfairness exhibited by the ML models with respect to different metrics

Light Gradient Boost (LGB) model for prediction. The code for building the two models are:

```
# Model HC1
HC1 = lgb.LGBMClassifier(n_estimators=10000, objective='binary', class_weight='balanced', learning_rate=0.05, reg_alpha=0.1, reg_lambda=0.1, subsample=0.8, n_jobs=-1, random_state=50)
HC1.fit(X_train, y_train, eval_metric = 'auc', categorical_feature = cat_indices, verbose = 200)
# Model HC2
HC2 = LGBMClassifier(n_estimators=4000, learning_rate=0.03, num_leaves=30, colsample_bytree=.8, subsample=.9, max_depth=7, reg_alpha=.1, reg_lambda=.1, min_split_gain=.01, min_child_weight=2, silent=-1, verbose=-1)
HC2.fit(X_train, y_train, eval_metric= 'auc', verbose= 100)
```

We have executed both the models with varied hyperparameter combinations and found that `class_weight='balanced'` is causing HC1 not to be biased towards privileged group. By specifying `class_weight`, we can set more weight to the data items belonging to an infrequent class. Higher class weight implies that the data items are getting more emphasis in prediction. When the class weight is set to `balanced`, the model automatically accounts for class imbalance and adjust the weight of data items inversely proportional to the frequency of the class [24, 42]. In this case, HC1 mitigates the *male-female* imbalance in its prediction. Therefore, it does not exhibit bias towards the privileged group (*male*). On the other hand, HC2 has less bias but it is biased towards privileged group. Although we want models to be fair with respect to all groups and individuals, trade-off might be needed and in some cases, bias toward unprivileged may be a desirable trait.

We have observed that `class_weight` hyperparameter in LGBMClassifier allows developers to control group fairness directly. However, the library documentation of LGB classifier suggests that this parameter is used for improving performance of the models [42, 46]. Though the library documentation mentions about probability calibration of classes to boost the prediction performance using this parameter, however, there is no suggestion regarding the effect on the bias introduced due to the wrong choice of this parameter.

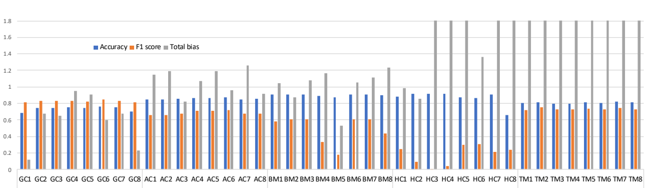

Figure 4: Cumulative bias and performance of the models

From the discussions, we can conclude that library developers still do not provide explicit ways to control fairness of the models. Although some parameters directly control the fairness of the models, libraries do not explicitly mention that.

Finding 3: Standardizing features before training models can help to remove disparity between groups in the protected class.

From Figure 3 and Figure 4, we observe that except BM5, other models in Bank Marketing exhibit similar unfairness. BM5 is a Support Vector Classifier (SVC) tuned using a grid search over given range of parameters. In the modeling pipeline, before training the best found SVC, the features are transformed using `StandardScalar`. Below is the model construction code for BM5 with the best found hyperparameters:

```
tuned_parameters = [{'kernel': ['rbf'], 'gamma': [0.1], 'C': [1]}]
SVC = GridSearchCV(SVC(), tuned_parameters, cv=5, scoring='precision')
# Best found SVC after grid search
# SVC(C=1, break_ties=False, cache_size=200, class_weight=None, coef0=0.0, decision_function_shape='ovr', degree=3, gamma=0.1, kernel='rbf', max_iter=-1, probability=True, random_state=None, shrinking=True, tol=0.001)
model = make_pipeline(StandardScaler(), SVC)
mdl = model.fit(X_train, y_train)
```

We have found that the usage of `StandardScalar` in the model pipeline is causing the model BM5 to be fairer. Especially DI of BM5 is 0.14 whereas, the mean of other seven BM models is very high (0.74). `StandardScalar` transforms the data features independently so that the mean value becomes 0 and the standard deviation becomes 1. Essentially, if a feature has variance in orders of magnitude than another feature, the model might learn from the dominating feature more, which might not be desirable [43]. In this case, Bank Marketing dataset has 55 features among which 41 has mean close to 0 ([0, 0.35]). However, *age* is the protected attribute having a mean value 0.97 (*older*: 1, *younger*: 0), since the number of older is significantly more than younger. Therefore, *age* is the dominating feature in these BM models. BM5 mitigates that effect by using standard scaling to all features. Therefore, balancing the importance of protected feature with other features can help to reduce bias in the models. This example also shows the importance of understanding the underlying properties of protected features and their effectiveness on prediction.

Finding 4: Dropping a feature from the dataset can change the model fairness effectively.

Both the models AC5 and AC6 are using XGB classifier for prediction but AC6 is fairer than AC5. Among the metrics, in terms of consistency (CNT), AC5 shows bias 3.61 times more than AC6. We have investigated the model construction and found that AC5 and AC6 differ in three constructs: features used in the model, number of trees used in the random forest, and learning rate of the classifier. We have observed that the number of trees and learning rate did not change the bias of the models. In AC5, the model excluded one feature from the training data. Bank Marketing dataset contains personal information about individuals and predicts whether the person has an annual income more than 50K dollars or not. In AC5, the model developer dropped one feature that contains number of years of education, since there is other categorical feature which represents education of the person (e.g., bachelors, doctorate, etc.). AC6 is using all the features in the dataset. CNT measures the individual fairness of the models i.e., how two similar individuals (not necessarily from different groups of protected attribute class) are classified to different outcomes. Therefore, dropping the number of years of education is causing the model to classify similar individuals to different outcome, which in turn generating individual unfairness.

Finding 5: Different metrics are needed to understand bias in different models.

From Figure 3, we can see that the models show different patterns of bias in terms of different fairness metrics. For example, compared to any Bank Marketing models, BM5 has disparity impact (DI) less than half but the error rate difference (ERD) more than twice. If the model developer only accounts for DI, then the model would appear fairer than what it actually is. As another example, GC6 is fairer than 90% of all the models in terms of total bias but if we only consider consistency (CNT), GC6 is fairer than only 50% of all the models. However, previous studies show that achieving fairness with respect to all the metrics is difficult and for some pair of metrics, mathematically impossible [5, 11, 37]. Also, the definition of fairness can vary depending on the application context and the stakeholders. Therefore, it is important to report on comprehensive set of fairness measures and evaluate the trade-off between the metrics to build fairer. We have plotted the correlation between different metrics from two datasets in Figure 5. A few metric pairs have a similar correlation in both the datasets such as (SPD, EOD), (SPD, AOD). This is understandable from the definitions of these metrics because they are calculated using same or correlated group conditioned rates (true-positives and false-positives). Although there are many metric pairs which are positively or negatively correlated, there is no pattern in correlation values between the two datasets. For instance, CNT and TI are highly negatively correlated in German Credit models but positively correlated in Titanic ML models. Therefore, we need a comprehensive set of metrics to evaluate fairness.

Finding 6: Except DI, EOD, and AOD, all the fairness measures remain consistent over multiple training and prediction.

To measure the stability of the fairness and performance metrics, we have computed the standard deviation of each metric over 10 runs similar to [16]. In each run, the dataset is shuffled before the train-test split, and model is trained on a new randomized training set. We have seen that the models are stable for the performance metrics and most of the fairness metrics. In particular, the average of the standard deviations of accuracy, F1 score, DI, SPD, EOD, AOD, ERD, CNT and TI over all the models are 0.01, 0.01, 0.12, 0.03, 0.04, 0.04, 0.03, 0.01, 0.01, respectively. Except for DI, EOD and AOD, the average standard deviation is very low (less than 0.03). For these three metrics, we have plotted the standard deviations in Figure 6. We can see that the trend of standard deviations is similar to the models of a specific dataset. In our benchmark, the largest dataset is Home Credit, which has the lowest standard deviation and the smallest dataset is Titanic ML, which has the most. Since in larger dataset, even after shuffling the training data remains more consistent, the deviation is less. On the other hand, the Titanic ML dataset is the smallest in size, having 891 data instances. The class distribution of data instances do not remain consistent when a random training set is chosen. Therefore, while dealing with smaller datasets, it is important to choose a training set that represents the original data and evaluate fairness multiple times.

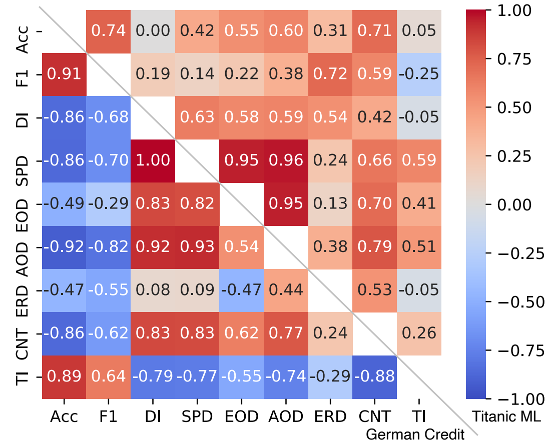

Figure 5: Corelation between the metrics. Bottom diagonal is for German Credit models, top diagonal is for Titanic ML models.

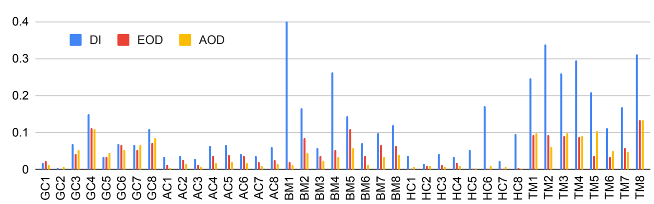

Figure 6: Standard deviation of the metrics: DI, EOD and AOD over multiple experiments. Other metrics have very low standard deviation.

DI has more standard deviation than other metrics. DI is computed using the ratio of two probabilities, P_u/P_p, where P_u is the probability of unprivileged group getting favorable label, and P_p is the probability of privileged group getting favorable label. Even the probability difference is very low, the value of DI can be very high. Therefore, DI fluctuates more frequently than other metrics.

Table 2: Unfairness measures of the models before and after the mitigations

	Model	Before mitigation									After mitigation									
		Acc	F1	DI	SPD	EOD	AOD	ERD	CNT	TI	Acc	F1	DI	SPD	EOD	AOD	ERD	CNT	TI	Rank
German Credit (Sex)*	GC1-RFT	.687	.814	.002	.002	0	.004	.052	-.002	.058	.683	.811	.002	.002	0	.004	-.032	-.002	.058	RAOD/PCE
	GC2-XGB	.743	.828	-.076	-.058	-.039	-.036	.047	-.282	.142	.709	.829	0	0	0	0	.067	0	.057	AORD/PCE
	GC3-XGB	.742	.827	-.105	-.079	-.043	-.065	.036	-.173	.149	.729	.831	-.045	-.040	-.006	-.043	.037	-.095	.100	AR/DPOCE
	GC4-SVC	.753	.832	-.138	-.104	-.081	-.068	.070	-.338	.153	.716	.834	0	0	0	0	.090	0	.057	AORD/PEC
	GC5-EVC	.743	.826	-.148	-.116	-.075	-.089	.067	-.286	.127	.687	.814	0	0	0	0	.112	0	.058	AORD/PEC
	GC6-RFT	.761	.845	-.103	-.083	-.023	-.085	.005	-.183	.121	.759	.844	-.071	-.058	-.023	-.085	-.027	-.183	.121	RD/APCEO
	GC7-XGB	.751	.831	-.073	-.056	.009	-.072	-.033	-.293	.144	.709	.829	0	0	0	0	.047	0	.057	ADR/POCE
	GC8-KNN	.698	.815	.003	.002	0	.011	.081	-.041	.090	.702	.825	0	0	0	0	.086	0	.057	AR/DPCOE
Adult Census (Race)*	AC1-LRG	.845	.657	-.654	-.104	-.100	-.069	-.050	-.045	.127	.261	.399	.023	.023	.017	.021	.120	-.019	.040	ORCDAP/E
	AC2-RFT	.846	.657	-.582	-.098	-.047	-.046	-.060	-.236	.119	.787	.249	-.354	-.014	.007	.003	-.086	-.005	.232	AROC/DPE
	AC3-GBC	.858	.677	-.496	-.079	-.041	-.031	-.045	-.010	.120	.858	.675	-.131	-.024	-.041	-.031	-.004	-.010	.120	ROAC/DPE
	AC4-CBC	.869	.712	-.616	-.102	-.077	-.056	-.044	-.069	.107	.805	.683	-.127	-.044	.080	.044	-.001	-.102	.082	ORAC/PDE
	AC5-XGB	.867	.708	-.588	-.097	-.073	-.051	-.043	-.224	.111	.865	.705	-.203	-.039	-.073	-.051	-.002	-.224	.111	ROAC/PDE
	AC6-XGB	.871	.717	-.570	-.096	-.044	-.036	-.047	-.062	.106	.808	.691	-.132	-.046	.072	.044	.009	-.094	.078	ORAC/PDE
	AC7-RFT	.852	.678	-.615	-.104	-.078	-.059	-.051	-.235	.117	.638	.329	-.289	-.024	-.005	-.009	-.039	-.009	.187	AORCD/PE
	AC8-DCT	.853	.675	-.519	-.086	-.040	-.035	-.050	-.068	.121	.852	.673	-.153	-.029	-.040	-.035	-.010	-.068	.121	ROAC/DPE
Bank Marketing (Age)	BM1-XGB	.906	.582	.627	.087	.074	.053	.051	-.078	.074	.905	.581	.274	.032	.074	.053	.017	-.078	.074	ROCPD/EA
	BM2-LGB	.908	.606	.593	.083	.004	.022	.069	-.034	.072	.772	.498	.076	.026	-.037	-.037	-.031	-.040	.066	ORDC/PAE
	BM3-GBC	.908	.604	.688	.100	.083	.056	.051	-.032	.072	.852	.529	.066	.013	-.059	-.052	006	-.089	078	CODR/APE
	BM4-XGB	.887	.330	.810	.048	.067	.042	.074	-.010	.111	.887	.328	.442	.022	.067	.042	.001	-.010	.111	RCA/OPDE
	BM5-SVC	.875	.175	.139	.003	-.077	-.031	.126	-.032	.126	.873	.002	.139	0	-.001	0	.110	0	.136	ERCDO/AP
	BM6-GBC	.908	.612	.698	.105	.030	.038	.076	-.033	.071	.795	.521	.110	.034	-.072	-.053	-.019	-.039	.065	OCRD/PAE
	BM7-XGB	.910	.611	.713	.107	.051	.052	.072	-.047	.070	.829	.485	.022	.004	-.037	-.044	-.007	-.122	.085	CODRA/PE
	BM8-RFT	.899	.435	.834	.066	.091	.058	.064	-.023	.097	.795	.462	.289	.042	-.048	-.027	.005	-.052	.073	ORACDP/E
Home Credit (Sex)	HC1-LGB	.883	.249	.574	.046	.065	.052	.051	-.110	.083	.238	.132	-.025	-.002	-.003	-.002	-.020	-.006	.030	APECR/OD
	HC2-LGB	.920	.094	-.698	-.006	-.016	-.010	-.032	-.012	.081	.919	.002	.076	0	0	0	-.033	0	.084	PECROA/D
	HC3-GNB	.913	.010	.974	.999	.007	.005	.006	-2.449	0	.732	.194	.181	.857	.047	.019	.031	-2.285	0	OA/DECPR
	HC4-XGB	.919	.046	.868	.994	.003	.013	.007	-2.482	0	.918	.012	-.103	.998	0	-.003	-.002	-2.468	0	CEDRP/OA
	HC5-CBC	.870	.302	.744	.865	.085	.140	.106	-2.524	0	.552	.075	-.134	.999	-.025	-.017	-.021	-2.772	.001	ACEPR/DO
	HC6-CBC	.869	.305	.735	.085	.144	.107	.068	-.147	.080	.583	.074	.021	0	0	0	.007	0	.056	ACPER/DO
	HC7-XGB	.911	.211	.953	.953	.033	.084	.054	-2.533	0	.907	.090	.408	.966	.009	-.052	-.019	-2.453	0	ECPR/DOA
	HC8-RFT	.661	.239	.383	.719	.147	.129	.133	-2.449	.001	.645	.226	.337	.681	.133	.098	.112	-2.426	.001	CPRD/AEO
Titanic ML (Sex)	TM1-XGB	.807	.720	-2.247	-.705	-.631	-.559	-.056	-.341	.153	.649	.580	-.082	-.039	.027	.177	.115	-.272	.189	OAERDP/C
	TM2-RFT	.816	.753	-2.013	-.709	-.635	-.515	.022	-.293	.142	.644	.566	-.106	-.045	.059	.166	.023	-.269	.223	OAERDP/C
	TM3-EBG	.799	.725	-2.125	-.674	-.637	-.514	-.017	-.333	.165	.647	.572	-.108	-.045	.031	.148	.050	-.317	.223	OAERD/PC
	TM4-LRG	.800	.732	-2.439	-.808	-.729	-.694	-.051	-.381	.144	.658	.577	-.075	-.034	.072	.160	.038	-.327	.207	OAEPRD/C
	TM5-GBC	.816	.740	-2.268	-.708	-.647	-.542	-.022	-.357	.151	.651	.572	-.087	-.033	.097	.174	.029	-.332	.205	OAERD/CP
	TM6-XGB	.804	.730	-1.948	-.665	-.583	-.499	-.042	-.345	.146	.625	.568	-.079	-.038	.075	.157	.092	-.367	.190	OAERD/CP
	TM7-RFT	.825	.747	-2.232	-.639	-.555	-.411	-.029	-.285	.161	.653	.577	-.099	-.043	.100	.188	.003	-.261	.219	OAERDP/C
	TM8-RFT	.814	.732	-2.306	-.716	-.633	-.563	-.051	-.321	.149	.649	.596	-.082	-.042	.011	.166	.157	-.327	.172	OAERD/PC

*Experiment has been conducted for multiple protected attributes. RFT: Random Forest, XGB: XGBoost, SVC: Support Vector Classifier, EVC: Ensemble Voting Classifier, KNN: K-Nearest Neighbors, LRG: Logistic Regression, GBC: Gradient Boosting Classifier, CBC: Cat Boost Classifier, DCT: Decision Tree, LGB: Light Gradient Boost, GNB: Gaussian Naive Bayes, EBG: Ensemble Bagging. Mitigation techniques applied to the models are as follows. Result is shown for the best mitigation. Rank of mitigation uses acronym below (mitigations before '/' have been able to mitigate bias, rest have not.)

Reweighing (R)	DI Remover (D)	Adversarial Debiasing (A)	Prejudice Remover (P)	Equalized Odds(E)	Calibrated Equalized Odds (C)	Reject Option Classification (O)

Finding 7: A fair model with respect to one protected attribute is not necessarily fair with respect to another protected attribute.

To understand the behavior of the same models on different protected attributes, we have analyzed the fairness of German Credit and Adult Census models on two protected attributes. In Figure 7, we have plotted the fairness measures of German Credit models on *sex* and *age* and Adult Census models on *sex* and *race*. We have found that the models can show different fairness when different protected attribute is considered. The total bias exhibited by German Credit dataset are: for *sex* attribute 4.82 and for *age* attribute 7.72. For Adult Census, the total bias are: for *sex* attribute 15.15 and for *race* attribute 8.56. However, most of the models exhibit similar trend of difference in the fairness when considering two different attributes.

GC1 and GC6 show cumulative bias 0.12 and 0.60 when *sex* is considered. Surprisingly, GC1 and GC6 shows cumulative bias 0.85 and 0.88 when *age* is considered. GC1 is much fairer model than GC6 in the first case but in the second case, the fairness is almost similar. We have discussed the behavior of these two models in Finding 1 and explained how GC1 is fairer when *sex* is the protected attribute. However, the fair prediction does not persist for the *age* because there is no imbalance in German Credit with respect to *age* groups. Therefore, GC1 and GC6 show similar fairness when *age* is considered.

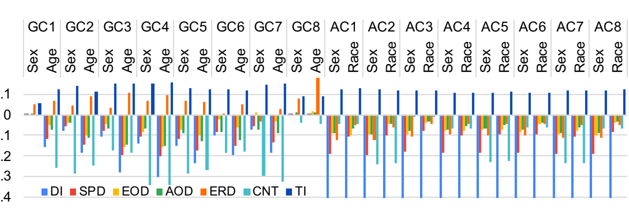

Figure 7: Fairness of ML models with respect to different protected attributes

5 MITIGATION

In this section, we have investigated the fairness results of the models after applying bias mitigation techniques. We have employed 7 different bias mitigation algorithms separately on 40 models and compared the fairness results with the original fairness exhibited by the models. For each model, we have selected the most successful mitigation algorithm and plotted the fairness values after mitigation in Figure 8. We have observed that similar to Figure 3, the fairness patterns are similar for the models in a dataset. DI, SPD, and CNT are the most difficult metrics to mitigate.

To understand the root causes of unfairness, we have focused on the models which exhibit more or less bias and then investigated the effects of different mitigation algorithms. Here, among the mitigation algorithms, the preprocessing techniques operate on the training data and retrain the original model to remove bias. On the other hand, post-processing techniques do not change the training data or original model but change the prediction made by the model. The in-processing techniques do not alter the dataset or prediction result but employ completely new modeling technique.

Finding 8: Models with effective preprocessing mitigation technique is preferable than others.

We have found that Reweighing algorithm has effectively debiased many models: GC1, GC6, AC3, AC5, AC8, BM1 and BM4. These models produce fairer results when the dataset is pre-processed using Reweighing. In other words, these models do not propagate bias themselves. In other cases where pre-processing techniques are not effective, we had to change the model or alter the prediction, which implies that bias is induced or propagated by the models. Another advantage is that in these models, after mitigations the models have retained the accuracy and F1 score. Other mitigation techniques often hampered the performance of the model. For a few other models (GC3, GC8, AC1, AC2, AC4, AC6, BM2, BM5, BM8), Reweighing has been the second most successful mitigation algorithm. Among these models, in AC1, AC2, BM2, and BM5, the most successful algorithm to mitigate bias loss accuracy or F1 score at least 22%. In all of these cases, Reweighing has retained both accuracy and F1 score.

Finding 9: Models with more bias are debiased effectively by post-processing techniques, whereas originally fairer models are debiased effectively by preprocessing or in-processing techniques.

From Table 2, we can see that 21 out of 40 models are debiased by one of the three post-processing algorithms i.e., Equalized odds (EO), Calibrated equalized odds (CEO), and Reject option classifier (ROC). These algorithms have been able to mitigate bias (not necessarily the most successful) in 90% of the models. Especially, ROC and CEO are the dominant post-processing techniques. ROC takes the model prediction, and gives the favorable outcome to the unprivileged group and unfavorable outcome to privileged group with a certain confidence around the decision boundary [35]. CEO takes the probability distribution score generated by the classifier and find the probability of changing outcome label and maximize equalized odds [41]. EO also changes the outcome label with certain probability obtained by solving a linear program [20]. We have found that these post-processing methods have been able to mitigate bias more effectively when the original model produces more biased results. From Figure 4, we can see that the most biased 5 models are TM4, TM8, TM5, TM1, HC7, where the post-processing has been the most successful algorithms. On the contrary, in case of the 5 least biased model (GC1, GC8, BM5, GC6, GC3), rather than mitigating, all three post-processing techniques increased bias.

In Table 2, we have shown the rank of mitigation algorithms to debias each model. In Table 3, we have shown the mean of the ranks of each mitigation algorithms, where rank of most successful algorithm is 1 and least is 7. We can see that for most biased models, Reject option classification and Equalized odds have been more successful than all others. For the least biased models, both preprocessing algorithms and Adversarial Debiasing have been effective, and the post-processing algorithms have been ineffective.

Table 3: Mean rank of each bias mitigation algorithm for 10 least biased models (LBM), 10 most biased models (MBM), and overall.

Stage	Algorithms	LBM	MBM	All
Preprocessing	Reweighing (R)	2.1	4.5	3.03
	Disparate Impact Remover (D)	3.7	4.8	4.58
In-processing	Adversarial Debiasing (A)	3	2.9	3
	Prejudice Remover Regularizer (P)	4.5	5.3	4.98
Post-processing	Equalized Odds (E)	5.8	2.8	5.18
	Calibrated Equalized Odds (C)	4.8	5.1	4.33
	Reject Option Classification (O)	4.1	2.6	2.93

6 IMPACT

While mitigating bias, there is a chance that the performance of the model is diminished. The most successful algorithm in debiasing a model does not always give good performance. So, often the developers have to trade-off between fairness and performance. In this section, we have investigated the answer to RQ3. What are the impacts when the bias mitigation algorithms are applied to the models? We have analyzed the accuracy and F1 score of the models after applying the mitigation algorithms. First, for each model, we have analyzed the impacts of the most effective mitigation algorithms in removing bias. In Figure 9, we have plotted the change in accuracy, F1 score, and total bias when the most successful mitigating algorithms are applied. We can see that while mitigating bias, many models are losing their performance. From Table 2, pre-processing algorithms, especially Reweighing has been the most effective in model GC1, GC6, AC3, AC5, AC8, BM1, and BM3. From Figure 9, these models always retain their performance after mitigation.

Finding 10: When mitigating bias effectively, in-processing mitigation algorithms show uncertain behavior in their performance.

Among in-processing algorithms, Adversarial debiasing has been the most effective in 11 models (GC2, GC3, GC4, GC5, AC2, AC7, HC1, HC5, HC6), and Prejudice remover has been the most effective in 1 model (HC2). We have found that for German Credit models Adversarial debiasing has been effective without losing performance.

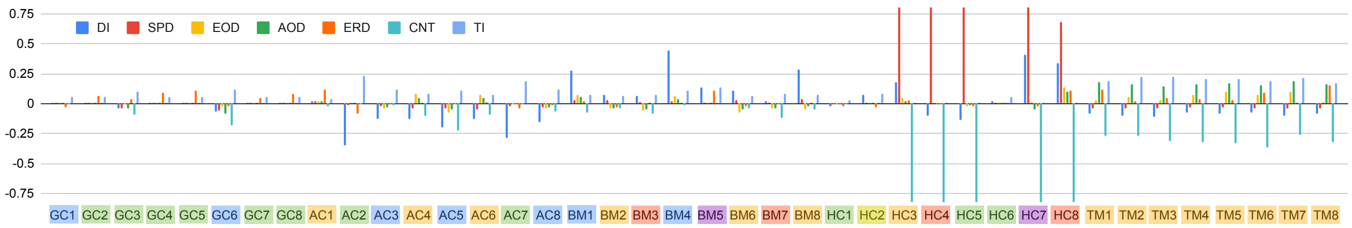

Figure 8: The fairness exhibited by the models after applying the bias mitigation techniques. The color coding in Table 2 is used to denote the most successful mitigation algorithm for each model.

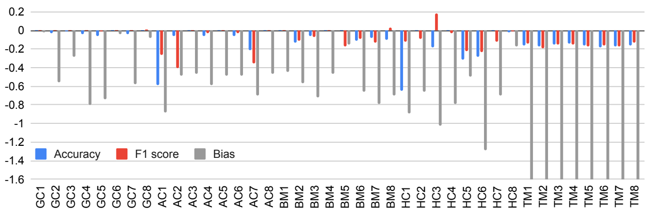

Figure 9: Change of performance and bias after applying bias mitigation technique (negative value indicates reduction)

But in other cases, AC1, AC7, HC1, and HC7, the accuracy has decreased at least 21.4%. In HC2, Prejudice remover also loses F1 score while mitigating the bias. Since, in-processing techniques employ new model and ignore the prediction of the original model, in all situations (dataset and task), it is not giving better performance. In our evaluation, adversarial debiasing is giving good performance with German Credit dataset but not on Adult Census or Home Credit dataset. Therefore, in-processing techniques are not appropriate when we can not change the original modeling. Also, since these techniques are uncertain in retaining performance, the developers should be careful about the accuracy of prediction after the intervention.

Finding 11: Although post-processing algorithms are the most dominating in debiasing, they are always diminishing the model accuracy and F1 score.

From Table 2, we can see that in 21 out of 40 models, one of the post-processing algorithms are being the most successful. But in all of the cases they are losing performance. The average accuracy reduction in these models is 7.49% and average F1 decrease is 10.07%. For example, in AC1, the most bias mitigating algorithm is Reject option classification but the model is loosing 26.1% accuracy and 40% F1 score. In these cases, developers should move to the next best mitigation algorithm. In a few other cases such as HC8, the Reject Option classification mitigates bias with only 1.6% loss in accuracy and 1.3% loss in f1 score. In such situations, post-processing techniques can be applied to mitigate the bias.

Finding 12: Trade-off between performance and fairness exists, and post-processing algorithms have most competitive replacement.

Since some most mitigating algorithms are having performance reduction, for each model, we have compared the most successful algorithm with the next best mitigation algorithm in Figure 10. We have found that for 18 out of 40 models, the performance of the 2nd ranked algorithm is same or better than the 1st ranked algorithm. Among them, in AC4, AC6, BM5, HC5, and HC8, the 2nd ranked algorithm has bias very close (not more than 0.1) to the 1st ranked one. All of these, except HC5, the 1st ranked bias mitigation algorithm is a post-processing technique. We observe that competitive alternative mitigation technique is more common for post-processing mitigation algorithms. Therefore, if we increase the tolerable range of bias, then other mitigation techniques would be better alternative in terms of performance.

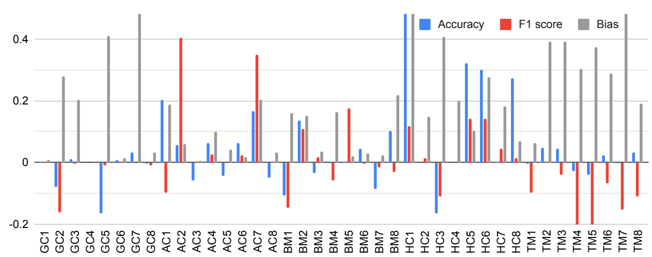

Figure 10: Change of performance and bias between the 1st and 2nd most successful mitigation algorithms (negative value indicates reduction)

7 THREATS TO VALIDITY

Benchmark Creation. To avoid experimenting on low-quality kernels, we have only considered the kernels with more than 5 votes. In addition, we have excluded the kernels where the model accuracy is very low (less than 65%). Finally, we have selected the top voted ones from the list. We have also verified that the collected kernels are runnable. To ensure the models collected from Kaggle are appropriate for fairness study, we have first selected the fairness analysis datasets from previous works and searched models for those datasets. Finally, we have searched competitions that use dataset with protected attributes used in the literature.

Fairness and performance evaluation. Our collected models give the same performance, as mentioned in the corresponding Kaggle kernels. For evaluating fairness and applying mitigation algorithms we have used AIF 360 toolkit [4] developed by IBM. Bellamy *et al.* presented fairness results (4 metrics) for two models (Logistic regression and Random forest) on Adult Census dataset with protected attribute *race* [4]. We have done experiment with the same setup and validated our result [4]. Similar to [16], for each metric, we have evaluated 10 times and taken the mean of the values. The stability comparison of the results is shown in §4.

Fairness comparison. As different metrics are computed based on different definitions of fairness, we have compared bias using a specific metric or cumulatively. Finally, in this paper, we have focused on comparing fairness of different models. Therefore, for each dataset, we followed the same method to pre-process training and testing data.

8 RELATED WORKS

SE for Fairness in ML. This line of work is the closest to our work. FairTest [48] proposes methodology to detect unwarranted feature associations and potential biases in a dataset using manually written tests. Themis [17] generates random tests automatically to detect causal fairness using black-box decision making process. Aequitas [49] is a fully automated directed test generation module to generate discriminatory inputs in ML models, which can be used to validate individual fairness. FairML [1] introduces an orthogonal transformation methodology to quantify the relative dependence of black-box models to its input features, with the goal of assessing fairness. A more recent work [3] proposes black-box fairness testing method to detect individual discrimination in ML models. They [3] propose a test case generation algorithm based on symbolic execution and local explainability. The above works have proposed novel techniques to detect and test fairness in ML systems. However, we have focused on empirical evaluation of fairness in ML models written by practitioners and reported our findings. Friedler *et al.* also worked on an empirical study but compared between fairness enhancing interventions and not models [16]. Harrison *et al.* conducted survey based empirical study to understand how fairness of different models is perceived by humans [21]. Holstein *et al.* also conducted survey on industry developers to find the challenges for developing fairness-aware tools and models [22]. However, no empirical study has been conducted to measure and compare fairness of ML models in practice, and analyze the impacts of mitigation algorithms on the models.

Fairness measure and algorithms. The machine learning community has focused on novel techniques to identify, measure and mitigate bias [8, 11, 13–15, 18, 20, 36, 38, 50]. This body of work concentrate on the theoretical aspects of bias in ML classifiers. Different fairness measures and mitigation algorithms have been discussed in §3.3 and §3.4. In this work, we have focused on the software engineering aspects of ML models used in practice.

ML model testing. DeepCheck [19] proposes lightweight white-box symbolic analysis to validate deep neural networks (DNN). DeepXplore [40] proposes a white-box framework to generate test input that can exploit the incorrect behavior of DNNs. DeepTest [47] uses domain-specific metamorphic relations to detect errors in DNN based software. These works have focused on the robustness property of ML systems, whereas we have studied fairness property that is fundamentally different from robustness [49].

9 CONCLUSION

ML fairness has received much attention recently. However, ML libraries do not provide enough support to address the issue in practice. In this paper, we have empirically evaluated the fairness of ML models and discussed our findings of software engineering aspects. First, we have created a benchmark of 40 ML models from 5 different problem domains. Then, we have used a comprehensive set of fairness metrics to measure fairness. After that, we have applied 7 mitigation techniques on the models and computed the fairness metric again. We have also evaluated the performance impact of the models after mitigation techniques are applied. We have found what kind of bias is more common and how they could be addressed. Our study also suggests that further SE research and library enhancements are needed to make fairness concerns more accessible to developers.

ACKNOWLEDGMENTS

This work was supported in part by US NSF under grants CNS-15-13263, and CCF-19-34884. All opinions are of the authors and do not reflect the view of sponsors.

REFERENCES

[1] Julius Adebayo and Lalana Kagal. 2016. Iterative orthogonal feature projection for diagnosing bias in black-box models. *arXiv preprint arXiv:1611.04967* (2016).
[2] Julius A Adebayo et al. 2016. *FairML: ToolBox for diagnosing bias in predictive modeling.* Ph.D. Dissertation. Massachusetts Institute of Technology.
[3] Aniya Aggarwal, Pranay Lohia, Seema Nagar, Kuntal Dey, and Diptikalyan Saha. 2019. Black box fairness testing of machine learning models. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering.* 625–635.
[4] Rachel KE Bellamy, Kuntal Dey, Michael Hind, Samuel C Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, Aleksandra Mojsilovic, et al. 2018. AI Fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias. *arXiv preprint arXiv:1810.01943* (2018).
[5] Richard Berk, Hoda Heidari, Shahin Jabbari, Michael Kearns, and Aaron Roth. 2018. Fairness in criminal justice risk assessments: The state of the art. *Sociological Methods & Research* (2018), 0049124118782533.
[6] Reuben Binns. 2017. Fairness in machine learning: Lessons from political philosophy. *arXiv preprint arXiv:1712.03586* (2017).
[7] Sumon Biswas and Hridesh Rajan. 2020. *ML-Fairness: Accepted Artifact for ESEC/FSE 2020 Paper on Fairness of Machine Learning Models.* https://doi.org/10.5281/zenodo.3912064
[8] Toon Calders and Sicco Verwer. 2010. Three naive Bayes approaches for discrimination-free classification. *Data Mining and Knowledge Discovery* 21, 2 (2010), 277–292.
[9] Jennifer Carpenter. 2011. May the best analyst win. American Association for the Advancement of Science.
[10] Jiahao Chen, Nathan Kallus, Xiaojie Mao, Geoffry Svacha, and Madeleine Udell. 2019. Fairness under unawareness: Assessing disparity when protected class is unobserved. In *Proceedings of the Conference on Fairness, Accountability, and Transparency.* 339–348.
[11] Alexandra Chouldechova. 2017. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big data* 5, 2 (2017), 153–163.
[12] George E Dahl, Navdeep Jaitly, and Ruslan Salakhutdinov. 2014. Multi-task neural networks for QSAR predictions. *arXiv preprint arXiv:1406.1231* (2014).
[13] Lucas Dixon, John Li, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2018. Measuring and mitigating unintended bias in text classification. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society.* 67–73.
[14] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. 2012. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference.* 214–226.
[15] Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. 2015. Certifying and removing disparate impact. In *proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining.* 259–268.
[16] Sorelle A Friedler, Carlos Scheidegger, Suresh Venkatasubramanian, Sonam Choudhary, Evan P Hamilton, and Derek Roth. 2019. A comparative study of fairness-enhancing interventions in machine learning. In *Proceedings of the Conference on Fairness, Accountability, and Transparency.* 329–338.
[17] 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.* 498–510.
[18] Gabriel Goh, Andrew Cotter, Maya Gupta, and Michael P Friedlander. 2016. Satisfying real-world goals with dataset constraints. In *Advances in Neural Information*

Processing Systems. 2415–2423.
[19] Divya Gopinath, Kaiyuan Wang, Mengshi Zhang, Corina S Pasareanu, and Sarfraz Khurshid. 2018. Symbolic execution for deep neural networks. *arXiv preprint arXiv:1807.10439* (2018).
[20] Moritz Hardt, Eric Price, and Nati Srebro. 2016. Equality of opportunity in supervised learning. In *Advances in neural information processing systems*. 3315–3323.
[21] Galen Harrison, Julia Hanson, Christine Jacinto, Julio Ramirez, and Blase Ur. 2020. An empirical study on the perceived fairness of realistic, imperfect machine learning models. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. 392–402.
[22] Kenneth Holstein, Jennifer Wortman Vaughan, Hal Daumé III, Miro Dudik, and Hanna Wallach. 2019. Improving fairness in machine learning systems: What do industry practitioners need?. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–16.
[23] Kathryn Jepsen. 2014. The machine learning community takes on the Higgs. https://www.symmetrymagazine.org/article/july-2014/the-machine-learning-community-takes-on-the-higgs.
[24] Prateek Joshi. 2017. *Artificial intelligence with python*. Packt Publishing Ltd.
[25] Kaggle. 2010. The world's largest data science community with powerful tools and resources to help you achieve your data science goals. www.kaggle.com.
[26] Kaggle. 2017. Adult Census Dataset. https://www.kaggle.com/uciml/adult-census-income.
[27] Kaggle. 2017. Bank Marketing Dataset. https://www.kaggle.com/c/bank-marketing-uci.
[28] Kaggle. 2017. Competition: Santander Product Recommendation. https://www.kaggle.com/c/santander-product-recommendation/overview.
[29] Kaggle. 2017. German Credit Dataset. https://www.kaggle.com/uciml/german-credit.
[30] Kaggle. 2017. Home Credit Dataset. https://www.kaggle.com/c/home-credit-default-risk.
[31] Kaggle. 2017. Titanic ML Dataset. https://www.kaggle.com/c/titanic/data.
[32] Kaggle. 2019. Adult Census Kernel: Multiple ML Techniques and Analysis. https://www.kaggle.com/bananuhbeatdown/multiple-ml-techniques-and-analysis-of-dataset.
[33] Kaggle. 2019. Kernel: German Credit Risk Analysis. https://www.kaggle.com/pahulpreet/german-credit-risk-analysis-beginner-s-guide.
[34] Faisal Kamiran and Toon Calders. 2012. Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems* 33, 1 (2012), 1–33.
[35] Faisal Kamiran, Asim Karim, and Xiangliang Zhang. 2012. Decision theory for discrimination-aware classification. In *2012 IEEE 12th International Conference on Data Mining*. IEEE, 924–929.
[36] Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma. 2012. Fairness-aware classifier with prejudice remover regularizer. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 35–50.
[37] Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan. 2016. Inherent trade-offs in the determination of risk scores. *arXiv preprint arXiv:1609.05807* (2016).
[38] Judea Pearl et al. 2009. Causal inference in statistics: An overview. *Statistics surveys* 3 (2009), 96–146.
[39] Dino Pedreshi, Salvatore Ruggieri, and Franco Turini. 2008. Discrimination-aware data mining. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 560–568.
[40] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2017. Deepxplore: Automated whitebox testing of deep learning systems. In *proceedings of the 26th Symposium on Operating Systems Principles*. 1–18.
[41] Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q Weinberger. 2017. On fairness and calibration. In *Advances in Neural Information Processing Systems*. 5680–5689.
[42] Scikit Learn. 2019. LightGBM API Documentation. https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMClassifier.html.
[43] Scikit Learn. 2019. SVC API Documentation. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html.
[44] Kacper Sokol, Raul Santos-Rodriguez, and Peter Flach. 2019. FAT Forensics: A Python Toolbox for Algorithmic Fairness, Accountability and Transparency. *arXiv preprint arXiv:1909.05167* (2019).
[45] 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*. 2239–2248.
[46] Stack Overflow. 2016. How does the class_weight parameter in scikit-learn work? https://stackoverflow.com/questions/30972029/how-does-the-class-weight-parameter-in-scikit-learn-work.
[47] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. 2018. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th international conference on software engineering*. 303–314.
[48] Florian Tramer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, Jean-Pierre Hubaux, Mathias Humbert, Ari Juels, and Huang Lin. 2017. FairTest: Discovering unwarranted associations in data-driven applications. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 401–416.
[49] 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*. 98–108.
[50] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P Gummadi. 2015. Fairness constraints: Mechanisms for fair classification. *arXiv preprint arXiv:1507.05259* (2015).
[51] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. 2013. Learning fair representations. In *International Conference on Machine Learning*. 325–333.
[52] 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*. 335–340.