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  
suman@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 between different groups in a 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

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 [15], and several fairness metrics and mitigation techniques [7, 10, 14, 19, 33, 35, 49, 51] have been proposed. Many testing strategies have been developed [3, 16, 48] to detect unfairness in software systems. Recently, a few tools have been proposed [2, 4, 43, 47] to enhance fairness in 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 Mechanical 502 Turk workers [20]. Recently, Holstein et al. conducted an empirical study on ML fairness by interviewing and surveying industry practitioners. They outlined the challenges faced by the developers and the support they need to build fair ML systems [21]. 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 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. 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 than other models 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 in §9, we conclude.

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 [15, 16]. 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 [9]. One trivial idea is to remove the protected attribute from the data and train data on that. Pedreshi et al. showed that due to redundant encoding of training data, it is possible that protected attribute is propagated to other correlated attributes [38]. 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 or female) [13]. Individual fairness measures whether a similar prediction is made for similar individuals (only different in protected attribute) [13]. Based on different definitions of fairness, many 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 are described in the next section.

3 METHODOLOGY

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

3.1 Benchmark Collection

We have collected ML models from Kaggle kernels [24]. 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 competitions, 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 [11], improving the search for the Higgs Boson at CERN [22], state-of-the-art HIV research [8], etc. There are 376 competitions and 28,622 datasets in Kaggle to date. Users 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 a solution code and description for a specific ML task. In this study, we have analyzed ML models using 1) datasets utilized by prior studies that are specific to fairness and 2) datasets with the protected attribute (age, sex, and race). Based on this goal, we have collected the ML models with different filtering criteria for each category. The overall process of collecting the benchmarks has been depicted in Figure 1.

To identify the datasets related to fairness used by prior works, we refer to the work on fairness testing by Galhotra et al. [16], where two datasets, German Credit and Adult Census have been utilized. Also, Udeshi et al. [48] experimented on models for the Adult Census dataset. Agarwal et al. [3] also used six datasets (German Credit Data, Adult census income, Bank marketing, US Executions, Fraud Detection, Raw Car Rentals). However, apart from the German Credit and Adult dataset, only the Bank marketing dataset is available on the Kaggle. Based on 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 kernels associated with the models having 1) prediction models (some kernels only contain exploratory data analysis), 2) 5 upvotes, and 3) accuracy ≥65%. Often a kernel contains multiple models and tries to find the best performing one. In these cases, we have selected only the best performing model from one kernel. Finally, we have selected the top 8 models for each dataset based on upvotes and have selected 24 ML models in this category.

Chen et al. [9] listed 12 protected attributes, e.g., age, sex, race, etc. Based on these criteria, we have selected 7 competitions, that contain the fields, from 376 competition in the Kaggle. From the selected competitions, we have filtered out the competitions that involve prediction decisions not being favorable to individuals or a specific group. For example, although this competition [27] 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. Finally, we got two competitions with several kernels. On selecting the ML models from the competitions, we have utilized the same filtering criteria used before and have selected 8 kernels for each dataset based on the top upvotes. Finally, we have created a benchmark containing 40 top-rated Kaggle models that operate on 5 datasets. The models and corresponding datasets are shown in Table 1. The characteristics of the datasets in the benchmark are as follows.

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.

In our benchmark, we have models from five dataset categories. To be able to compare the fairness of different models in the same dataset category, we have used the same data preprocessing strategy. We 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 do 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 and female is unprivileged group and the prediction label is credit risk of the person i.e., good (favorable label) or bad. For all the datasets, we use shuffling and the same train (70%) and test (30%) split before feeding the data to our 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 the following 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 [31] loads the German Credit dataset, performs exploratory analysis and selects a subset of the features for training, preprocesses data and implements XGBoost Classification model 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 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 section 3.3.2. Next, we have applied 7 different bias mitigation algorithms separately and evaluated the performance and fairness metrics. Thus, we collect 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. We have used the open source Python library AI Fairness 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 \((Y = 1 \text{ if the label is favorable, otherwise } Y = 0)\), \( Z \) is the protected attribute \((Z = 1 \text{ for privileged group, otherwise } Z = 0)\) 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 compute the performance in terms of accuracy, and F1 score.
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 post-processing 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 [33]: 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 has been assigned in comparison to the class not been favored.

Disparate Impact Remover [14]: This algorithm is based on the concept of the metric DI that measures the fraction of individuals achieves positive outcome 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 [51]: 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 [35]: 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) [19]: This approach also 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 [40]: To achieve fairness, this technique also optimizes EOD metric by using the calibrated prediction score produced by the classifier.

Reject Option Classification [34]: 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.

Figure 3: The unfairness exhibited by the ML models

We have found that all the models exhibit unfairness, and models specific to a dataset show similar bias pattern. 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 cumulatively or using the same metric. To compare total bias across all the metrics, we have taken the absolute value of the measures and computed 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, a few 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 parameters. After a 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, verbose=0, warm_start=False)
```

We have found that GC6 is also a Random Forest classifier built through grid search. However, GC6 exhibit more bias 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 with changing one hyperparameter at a time. We have found that the difference is caused by the scoring mechanism used by the two models. GC1 uses scoring='recall', whereas GC6 uses scorings='precision', shown in following code snippet.

```
# 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% male data 31% female. 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 (men) is more than the unprivileged group (women), 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, 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 found 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 GC6 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 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 exhibit 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. 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 Light Gradient Boost model (LGB) for prediction. The code for building the two models are:

```
# Model HC1
HC1 = 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=0.8, subsample=0.9,
                             max_depth=7, reg_alpha=1, reg_lambda=1,
                             min_child_weight=0.1, min_split_gain=0.1,
                             random_state=50)
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 weight of data items inversely proportional to the frequency of the class [23, 41]. 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 LGBMClassifier suggests that this parameter is used for improving performance of the models [41, 45]. 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.

From the above two findings, we can see that the library developers still do not provide explicit ways to control fairness of the models. Although some parameters directly control fairness, 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 parameters:

```
#SVC (C=1, break_ties=False, cache_size=200, class_weight=None, kernel='rbf',
     degree=3, gamma=0.01, decision_function_shape='ovo',
     max_iter=100, probability=True, random_state=None, shrinking=True, tol=0.001, verbose=False)
model = make_pipeline(StandardScaler(), SVC)
md1 = model.fit(X_train, y_train)
```

We have found that usage of `StandardScalar` in the model pipeline is causing the model to be fairer. Especially DI of BM5 is 0.14 whereas, the mean of other seven BM models is 0.74. `StandardScalar` transforms the data features independently so that the mean value is 0 and standard deviation is 1. Essentially, if a feature has variance in orders of magnitude than another feature, the model might learn from the dominating feature more, which is not desirable [42]. In this case, Bank Marketing dataset has 55 features among which 41 has mean close to 0 ([0, 0.35]). Here, age is the protected attribute having a mean of 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 where BM5 mitigates that effect by using standard scaling to all features. Therefore, balancing the protected feature importance 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 effect on prediction.

**Finding 4:** Dropping a feature in the dataset can change the model fairness.

AC6 and AC5 both are using XGB classifier for prediction but AC6 is fairer than AC5. Among all the metrics, in terms of consistency metric (CNT), AC5 has 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 observed that 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 one has income more than 50K dollar or not. In AC5, the model developer dropped one feature that contain number of years of education, since there are other categorical features related to education (e.g., bachelors, doctorate, etc.). But 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 and generating individual unfairness.
Table 2: Model unfairness measures and mitigations

<table>
<thead>
<tr>
<th>Model</th>
<th>Before mitigation</th>
<th>After mitigation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Acc</td>
<td>FI</td>
</tr>
<tr>
<td>GCI-RFT</td>
<td>0.874</td>
<td>0.002</td>
</tr>
<tr>
<td>GCI-XGB</td>
<td>0.736</td>
<td>0.006</td>
</tr>
<tr>
<td>GCI-XGB</td>
<td>0.712</td>
<td>0.005</td>
</tr>
<tr>
<td>GCI-SVC</td>
<td>0.753</td>
<td>0.003</td>
</tr>
<tr>
<td>GCI-EPIC</td>
<td>0.743</td>
<td>0.004</td>
</tr>
<tr>
<td>GCI-EPIC</td>
<td>0.745</td>
<td>0.003</td>
</tr>
<tr>
<td>GCI-EPIC</td>
<td>0.751</td>
<td>0.003</td>
</tr>
<tr>
<td>GCI-EPIC</td>
<td>0.751</td>
<td>0.003</td>
</tr>
<tr>
<td>GCI-EPIC</td>
<td>0.751</td>
<td>0.003</td>
</tr>
</tbody>
</table>

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 measures. For example, BMS has disparity impact (DI) less than half but the error rate difference (ERD) more than twice compared to any other models in that dataset. 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 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 pairs of metrics, mathematically impossible [5, 10, 36]. Therefore, it is important to report on comprehensive set of fairness measures and evaluate the trade-off between the metrics. We have plotted the correlation between different metrics from two datasets in Figure 5. A few metric pairs have similar correlation in both the datasets like (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, 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 comprehensive set of metrics to evaluate fairness.

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

To measure the stability of the fairness and performance measure, we have computed the standard deviation of each metric over 10 runs similar to [15]. In each run, the dataset is shuffled before the train-test split, and model is trained on the new set of training data. We have seen that the models are stable for the performance metrics and most of the fairness metrics. In particular, the average
we have plotted the fairness measures of German Credit models, top diagonal is for Titanic ML models. The in-processing techniques do not alter the dataset, but employ the effects of different mitigation algorithms. Here, among the in-processing techniques, 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, but employ a completely new model.

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 produce fairer results when the dataset is pre-processed.
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 pre-processing 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 [34]. CEO takes the probability distribution score generated by the classifier and find the probability of changing outcome label and maximize equalized odds [40]. EO also changes the outcome label with certain probability obtained by solving a linear program [19]. We have found that these 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 when applied on these models except BM5. In Table 2, we have shown the rank of mitigation algorithms to debias each model. In Table 3, we have shown the mean rank of the six bias mitigation algorithms. 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.

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

Among in-processing algorithms, Adversarial debiasing has been the most effective in 11 (GC2, GC3, GC4, GC5, AC2, AC7, HC1, HC5, HC6) models 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. 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 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 good performance. In this case, along with debiasing, adversarial debiasing is giving good performance with German Credit dataset but not on Adult Census or Home Credit dataset.

Table 3: Mean rank of each bias mitigation algorithm. LBM: 10 least biased models, MBM: 10 most biased models.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Algorithms</th>
<th>LBM</th>
<th>MBM</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preprocessing</td>
<td>Reweighing (R)</td>
<td>2.1</td>
<td>4.5</td>
<td>3.03</td>
</tr>
<tr>
<td></td>
<td>Disparate Impact Remover (D)</td>
<td>3.7</td>
<td>4.8</td>
<td>4.58</td>
</tr>
<tr>
<td>In-processing</td>
<td>Adversarial Debiasing (A)</td>
<td>3</td>
<td>2.9</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Prejudice Remover Regularizer (P)</td>
<td>4.5</td>
<td>5.3</td>
<td>4.98</td>
</tr>
<tr>
<td>Post-processing</td>
<td>Equalized Odds (E)</td>
<td>5.8</td>
<td>2.8</td>
<td>5.18</td>
</tr>
<tr>
<td></td>
<td>Calibrated Equalized Odds (C)</td>
<td>4.8</td>
<td>5.1</td>
<td>4.33</td>
</tr>
<tr>
<td></td>
<td>Reject Option Classification (O)</td>
<td>4.1</td>
<td>2.6</td>
<td>2.93</td>
</tr>
</tbody>
</table>
From Table 2, we can see that in 21 out of 40 models, any of the three 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.

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 issues, 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. 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 mitigating algorithm (rank 2 - rank1)

8 RELATED WORKS

SE for Fairness in ML. This line of work is the closest to our work. FairTest [47] proposes methodology to detect unwarranted feature associations and potential biases in a dataset using manually written tests. Themis [16] generates random tests automatically to detect causal fairness using black-box decision making process. Aequitas [48] 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 balck-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.

Finding 11: Although post-processing algorithms are the most dominating in debiasing, they are always diminishing the model accuracy and f1 score.

Friedler et al. also worked on an empirical study but compared between fairness enhancing interventions and not models [15]. Harrison et al. conducted survey based empirical study to understand how fairness of different models is perceived by humans [20]. Holstein et al. also conducted survey on industry developers to find the challenges and need to develop fairness-aware tools and models [21]. However, no empirical study has been conducted to measure and compare fairness of ML models in practice.

Friedler et al. also worked on an empirical study but compared between fairness enhancing interventions and not models [15]. Harrison et al. conducted survey based empirical study to understand how fairness of different models is perceived by humans [20]. Holstein et al. also conducted survey on industry developers to find the challenges and need to develop fairness-aware tools and models [21]. However, no empirical study has been conducted to measure and compare fairness of ML models in practice.

Fairness measure and algorithms. The machine learning community has focused on novel techniques to identify, measure and mitigate bias [7, 10, 12–14, 17, 19, 35, 37, 49]. 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 [18] proposes lightweight white-box symbolic analysis to validate deep neural networks (DNN). DeepXplore [39] proposes a white-box framework to generate test input that can exploit incorrect behavior of DNNs. DeepTest [46] 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 [48].
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 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 performance impacts of the models when mitigation techniques is applied. We have found what kind of bias is more common and how they could be addressed. Our study also suggests further SE research in ML fairness, and library enhancement 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


