Fair Preprocessing: Towards Understanding Compositional Fairness of Data Transformers in Machine Learning Pipeline

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

In recent years, many incidents have been reported where machine learning models exhibited discrimination among people based on race, sex, age, etc. Research has been conducted to measure and mitigate unfairness in machine learning models. For a machine learning task, it is a common practice to build a pipeline that includes an ordered set of data preprocessing stages followed by a classifier. However, most of the research on fairness has considered a single classifier based prediction task. What are the fairness impacts of the preprocessing stages in machine learning pipeline? Furthermore, studies showed that often the root cause of unfairness is ingrained in the data itself, rather than the model. But no research has been conducted to measure the unfairness caused by a specific transformation made in the data preprocessing stage. In this paper, we introduced the causal method of fairness to reason about the fairness impact of data preprocessing stages in ML pipeline. We leveraged existing metrics to define the fairness measures of the stages. Then we conducted a detailed fairness evaluation of the preprocessing stages in 37 pipelines collected from three different sources. Our results show that certain data transformers are causing the model to exhibit unfairness. We identified a number of fairness patterns in several categories of data transformers. Finally, we showed how the local fairness of a preprocessing stage composes in the global fairness of the pipeline. We used the fairness composition to choose appropriate downstream transformer that mitigates unfairness in the machine learning pipeline.

CCS CONCEPTS

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

KEYWORDS

fairness, machine learning, preprocessing, pipeline, models

ACM Reference Format:
Sumon Biswas and Hridesh Rajan. 2021. Fair Preprocessing: Towards Understanding Compositional Fairness of Data Transformers in Machine Learning Pipeline. In *Proceedings of the 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '21), August 23–28, 2021, Athens, Greece.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3468264.3468536

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
ESEC/FSE '21, August 23–28, 2021, Athens, Greece
© 2021 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-8562-6/21/08.
https://doi.org/10.1145/3468264.3468536

1 INTRODUCTION

Fairness of machine learning (ML) predictions is becoming more important with the rapid increase of ML software usage in important decision making [5, 22, 30, 49], and the black-box nature of ML algorithms [3, 27]. There is a rich body of work on measuring fairness of ML models [15, 19, 23, 25, 32, 64, 71, 73] and mitigate the bias [15, 19, 29, 32, 41, 54, 71, 74]. Recent work [10, 13, 17, 26, 33, 35] has shown that more software engineering effort is required towards detecting bias in complex environment and support developers in building fairer models.

The majority of work on ML fairness has focused on classification task with single classifier [3, 12, 25, 27]. However, real-world machine learning software operate in a complex environment [12, 21]. In an ML task, the prediction is made after going through a series of stages such as data cleaning, feature engineering, etc., which build the machine learning pipeline [4, 70]. Studying only the fairness of the classifiers (e.g., *Decision Tree*, *Logistic Regression*) fails to capture the fairness impact made by other stages in ML pipeline. In this paper, we conducted a detailed analysis on how the data preprocessing stages affect fairness in ML pipelines.

Prior research observed that bias can be encoded in the data itself and missing the opportunity to detect bias in earlier stage of ML pipeline can make it difficult to achieve fairness algorithmically [22, 31, 35, 43]. Additionally, bias mitigation algorithms operating in the preprocessing stage were shown to be successful [25, 40]. Therefore, it is evident that the preprocessing stages of ML pipeline can introduce bias. However, no study has been conducted to measure the fairness of the preprocessing stages and show how it impacts the overall fairness of the pipeline. In this paper, we used the causal method of fairness to reason about the fairness impact of preprocessing stages in ML pipeline. Then, we leveraged existing fairness metrics to measure fairness of the preprocessing stages. Using the measures, we conducted a thorough analysis on a benchmark of 37 real-world ML pipelines collected from three different sources, which operate on five datasets. These ML pipelines allowed us to evaluate fairness of a wide selection of preprocessing stages from different categories such as data standardization, feature selection, encoding, over/under-sampling, imputation, etc. For comparative analysis, we also collected data transformers e.g., `StandardScaler`, `MinMaxScaler`, `PCA`, `l1-normalizer`, `QuantileTransformer`, etc., from the pipelines as well as corresponding ML libraries, and evaluated fairness. Finally, we investigated how fairness of these preprocessing techniques (*local fairness*) composes with other preprocessing stages, and the whole pipeline (*global fairness*). Specifically, we answered the following three research questions.

RQ1 (fairness of preprocessing stages): What are the fairness measures of each preprocessing stage in ML pipeline? **RQ2** (fair

transformers): What are the fair (and biased) data transformers among the commonly used ones? **RQ3** (fairness composition): How fairness of data preprocessing stages composes in ML pipeline?

- How local fairness compose into global fairness?
- Does choosing a downstream transformer depend on the fairness of an upstream transformer?

To the best of our knowledge, we are the first to evaluate the fairness of preprocessing stages in ML pipeline. Our results show that by measuring the fairness impact of the stages, the developers would be able to build fairer predictions effectively. Furthermore, the libraries can provide fairness monitoring into the data transformers, similar to the performance monitoring for the classifiers. Our evaluation on real-world ML pipelines also suggests opportunities to build automated tool to detect unfairness in the preprocessing stages, and instrument those stages to mitigate bias. We have made the following contributions in this paper:

(1) We created a fairness benchmark of ML pipelines with several stages. The benchmark, code and results are shared in our replication package[1] in GitHub repository, that can be leveraged in further research on building fair ML pipeline.
(2) We introduced the notion of causality in ML pipeline and leveraged existing metrics to measure the fairness of preprocessing stages in ML pipeline.
(3) Unfairness patterns have been identified for a number of stages.
(4) We identified alternative data transformers which can mitigate bias in the pipeline.
(5) Finally, we showed the composition of stage-specific fairness into overall fairness, which is used to choose appropriate downstream transformer that mitigates bias.

The paper is organized as follows: §2 describes the motivating examples, §3 describes the existing metrics and our approach. In §4, we described the benchmark and experiments. §5 explores the results, §6 provides a comparative study among transformers, and §7 evaluates the fairness composition. Finally, §9 describes the threats to validity, §10 discusses related work, and §11 concludes.

2 MOTIVATION

In this section, we present two ML pipelines which show that the preprocessing stage affects the fairness of the model and it is important to study the bias induced by certain data transformers.

2.1 Motivating Example 1

Yang *et al.* [70] studied the following ML pipeline which was originally outlined by Propublica for recidivism prediction on *Compas* dataset [5]. The goal is predict future crimes based on the data of defendants. The fairness values, in terms of statistical parity difference (SPD: -0.102) and equal opportunity difference (EOD: -0.027), suggest that the prediction is biased towards [2] *Caucasian* defendants when *race* is considered as sensitive attribute. The pipeline consists of several preprocessing stages before applying `LogisticRegression` classifier. Data preprocessing includes cleaning, encoding categorical features, and missing value imputation. Recent research [70] showed that the data transformation in this pipeline is not symmetric across gender groups i.e., male defendants are filtered more than the female. Do these data transformations introduce unfairness in the prediction? If yes, what are the unfairness measures of these transformers? Is it possible to leverage existing metrics to measure the unfairness of each component? If we can understand the effect of each data transformer, it would be possible to choose data preprocessing technique wisely to avoid introducing bias as well as mitigate the inherent bias in data or classifier.

```
df = pd.read_csv(f_path)
df = df[(df.days_b_screening_arrest <= 30)
    & (df.days_b_screening_arrest >= -30)
    & (df.is_recid != -1) & (df.c_charge_degree != 'O')
    & (df.score_text != 'N/A')]
df = df.replace('Medium', 'Low')
labels = LabelEncoder().fit_transform(df.score_text)
impute1_onehot = Pipeline([
    ('imputer1', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))])
impute2_bin = Pipeline([
    ('imputer2', SimpleImputer(strategy='mean')),
    ('discretizer', KBinsDiscretizer(n_bins=4, encode='ordinal',
       strategy='uniform'))])
featurizer = ColumnTransformer(transformers=[
    ('impute1_onehot', impute1_onehot, ['is_recid']),
    ('impute2_bin', impute2_bin, ['age'])])
pipeline = Pipeline([('features', featurizer),
    ('classifier', LogisticRegression())])
```

2.2 Motivating Example 2

The following ML pipeline is collected from the benchmark used by Biswas and Rajan [10] for studying fairness of ML models. This pipeline operates on *German Credit* dataset. Here, the goal is to predict the credit risk (good/bad) of individuals based on their personal data such as age, sex, income, etc. In this pipeline, before training the classifier, data has been processed using two transformers: `PCA` for principal component analysis, and `SelectKBest` for selecting high-scoring features. The fairness value (SPD: 0.005) shows that prediction is slightly biased towards *female* candidates. However, if the transformers are not applied, then prediction becomes biased towards *male* (SPD: -0.117). By applying one transformer at a time, we observed that `PCA` alone is not causing the change of fairness. In this case, `SelectBest` is causing bias towards *female*, which in turn mitigating the overall fairness of the pipeline. Therefore, in addition to study the fairness of transformers in isolation, it is important to understand how fairness of components composes in the pipeline.

```
features = []
features.append(('pca', PCA(n_components=2)))
features.append(('select_best', SelectKBest(6)))
feature_union = FeatureUnion(features)
estimators = []
estimators.append(('feature_union', feature_union))
estimators.append(('RF', RandomForestClassifier()))
model = Pipeline(estimators)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

Our key idea is to leverage causal reasoning and observe fairness impact of a stage on prediction. To do that we create alternative pipeline by removing a stage. For example, from the above pipeline, we remove the `SelectKBest` and compare the predictions with original pipeline. We observe that `SelectKBest` is causing 1.1% of the female and 3.6% of the male participants to change predictions from favorable (good credit) to unfavorable (bad credit). Since the stage is causing more unfavorable decisions to male, the stage is biased towards female. Thus, we used existing fairness criteria to measure fairness impact of a stage and propose novel metrics.

[1] https://github.com/sumonbis/FairPreprocessing
[2] Bias *towards* a group connotes that the prediction favours that group.

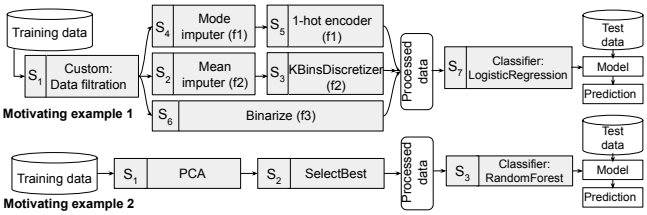

Figure 1: ML pipelines for the motivating examples, having a sequence of preprocessing stages followed by a classifier.

3 METHODOLOGY

In this section, first, we describe the background of ML pipeline, focussing on the data preprocessing stages. Second, we formulate the method and metrics to measure fairness of a certain preprocessing stage with respect to the pipeline it is used within.

3.1 ML Pipeline

Amershi *et al.* proposed a nine-stage machine learning pipeline with data-oriented (collection, cleaning, and labeling) and model-oriented (model requirements, feature engineering, training, evaluation, deployment, and monitoring) stages [4]. Other research [1, 7] also described data preprocessing as an integral part of the ML pipeline. The pipelines in the motivating examples are depicted in Figure 1, which follows the representation provided by Yang *et al.* [70]. In this paper, we adapted the canonical definition of pipeline from Scikit-Learn pipeline specification [14, 62], which is aligned with the ML models studied in the literature for fair classification tasks [3, 8, 10, 26, 27, 70]. We are interested in investigating the fairness of the data preprocessing stages in the pipeline, which is depicted with grey boxes in Figure 1.

To summarize, a canonical *ML pipeline* is an ordered set of m stages with a set of *preprocessing stages* $(S_1, S_2, \ldots S_{m-1})$ and a final *classifier* (S_m). Each preprocessing stage, S_k operates on the data already processed by preceding stages $S_1, \ldots S_{k-1}$. A data preprocessing stage S_k can be a data transformer or a set of custom operations. A *data transformer* is a well-known algorithm or method to perform a specific operation such as variable encoding, feature selection, feature extraction, dimensionality reduction, etc. on the data [14]. For example, in the second motivating example, two transformers (`PCA` and `SelectKBest`) have been used. *Custom transformation* includes data/task-specific contextual operations on the dataset. For example, in §2.1 (line 2-3), the data instances that do not contain a value in the range [-30, 30] for the feature `days_b_screening_arrest`, have been filtered. This means the pipeline ignored the data of the defendants with more than 30 days between their screening and arrest. This formulation of ML pipeline allowed us to evaluate fairness of the preprocessing stages in real-world ML tasks.

3.2 Existing Fairness Metrics

We have leveraged existing fairness metrics to measure the fairness of the whole pipeline. Many fairness metrics have been proposed in the literature for measuring fairness of classification tasks [8, 9, 22]. In general, the fairness metrics compute group-specific classification rates (e.g., true positives, false positives), and calculates the difference between groups to measure the fairness. In this paper, we adopted the representative group fairness metrics used by [10, 26]. Specifically, we leveraged the following metrics: statistical parity difference (SPD) [25, 40, 71], equal opportunity difference (EOD) [32], average odds difference (AOD) [32], and error rate difference (ERD) [19]. Given a dataset D with n instances, let, actual classification label be Y, predicted classification label be $\hat{Y}$, and sensitive attribute be $\mathcal{A}$. Here, $Y = 1$ if the label is favorable to the individuals, otherwise $Y = 0$. For example, classification task on *German Credit* dataset predicts the credit risk (good/bad credit) of individuals. In this case, $Y = 1$ if the prediction is *good credit*, otherwise $Y = 0$. Suppose, for privileged group (e.g., *White*), $\mathcal{A} = 1$ and for unprivileged group (e.g., *non-White*), $\mathcal{A} = 0$. SPD is computed by observing the probability of giving favorable label to each group and taking the difference. EOD measures the true-positive rate difference between groups. AOD calculates both true positive rate and false positive rate difference and then takes the average. ERD calculates the sum of false positive rate difference and false negative rate difference between groups. The definitions of these metrics are as follows:

$$
\begin{aligned}
\text{SPD} &= \text{P}[\hat{Y} = 1 | \mathcal{A} = 0] - \text{P}[\hat{Y} = 1 | \mathcal{A} = 1] \\
\text{EOD} &= \text{P}[\hat{Y} = 1 | Y = 1, \mathcal{A} = 0] - \text{P}[\hat{Y} = 1 | Y = 1, \mathcal{A} = 1] \\
\text{AOD} &= (1/2)\{(\text{P}[\hat{Y} = 1 | Y = 1, \mathcal{A} = 0] - \text{P}[\hat{Y} = 1 | Y = 1, \mathcal{A} = 1]) \\
&\quad + (\text{P}[\hat{Y} = 1 | Y = 0, \mathcal{A} = 0] - \text{P}[\hat{Y} = 1 | Y = 0, \mathcal{A} = 1])\} \\
\text{ERD} &= (\text{P}[\hat{Y} = 1 | Y = 0, \mathcal{A} = 0] - \text{P}[\hat{Y} = 1 | Y = 0, \mathcal{A} = 1])\} \\
&\quad + (\text{P}[\hat{Y} = 0 | Y = 1, \mathcal{A} = 0] - \text{P}[\hat{Y} = 0 | Y = 1, \mathcal{A} = 1])\}
\end{aligned} \tag{1}
$$

Disparate impact (DI) and statistical parity difference (SPD) both measure the same rate i.e., probability of classifying data instance as favorable, but DI computes the ratio of privileged and unprivileged groups' rate, whereas SPD computes the difference. Therefore, from DI and SPD, we only used SPD in our evaluation.

3.3 Fairness of Preprocessing Stages

Suppose, $\mathcal{P}$ is a pipeline with m stages and our goal is to evaluate the fairness of the stage S_k, where $1 \leq k < m$. In other words, we want to measure the fairness impact of S_k on the prediction made by $\mathcal{P}$. To achieve that we applied the causal reasoning for evaluating fairness. The causality theorem was proposed by Pearl [52, 53] and further studied extensively to reason about fairness in many scenarios [27, 46, 56, 58, 75]. Causality notion of fairness captures that everything else being equal, the prediction would not be changed in the counterfactual world where only an intervention happens on a variable [27, 46, 56]. For example, Galhotra et al. proposed causal discrimination score for fairness testing [27]. The authors created test inputs by altering original protected attribute values of each data instance, and observed whether prediction is changed for those test inputs. If the intervention causes the prediction to be changed, we call the software causally unfair with respect to that intervention. In our case, if a preprocessing stage S_k be the intervention, to measure the fairness of S_k, we have to capture the prediction disparity caused by the intervention S_k. This causal reasoning of fairness is a stronger notion since it provides causality in software by observing changes in the outcome made by a specific stage in the pipeline [27, 53].

3.3.1 Causal method to measure fairness of preprocessing stage. From pipeline $\mathcal{P}$, we construct another pipeline $\mathcal{P}^*$ by only excluding the stage S_k from $\mathcal{P}$. After applying the stage S_k in $\mathcal{P}^*$, to what extent the prediction of $\mathcal{P}^*$ changes, and whether the change is favorable to any group? Broadly, this can be measured by observing the prediction difference between $\mathcal{P}$ and $\mathcal{P}^*$ and computing the fairness of these changes using the fairness metrics from (1).

Suppose, the predictions made by the two pipelines are $\hat{Y}(\mathcal{P})$ and $\hat{Y}(\mathcal{P}^*)$. Let, I be the impact set for S_k, which denotes the prediction parity between $\hat{Y}(\mathcal{P})$ and $\hat{Y}(\mathcal{P}^*)$ such that for i^{th} data instance, if $\hat{Y}_i(\mathcal{P}) = \hat{Y}_i(\mathcal{P}^*)$, then $I_i = 0$, otherwise 1. By causality, the fairness of preprocessing stage (denoted by SF) is calculated based on $[\hat{Y}(\mathcal{P}), \hat{Y}(\mathcal{P}^*)]_{I=1}$ with respect to a fairness metric M, which is shown in (2a). We noticed that a few preprocessing stages, specifically the encoders can not be removed without replacing with an alternative stage. For such situations, we have defined the fairness of S_k with reference to another stage S'_k, denoted by $\mathsf{SF}(\mathsf{S_k}|\mathsf{S'_k})$ in (2b).

Zelaya also used the similar method for quantifying the effect of a preprocessing stage with a goal of computing *volatility* of a stage [72]. *Volatility* quantifies how much impact a preprocessing stage has on the outcome by computing the probability of prediction changes. However, it does not capture the fairness of the stage, since a stage can cause high change in the prediction by maintaining the predictions fair. Next in §3.3.2, we have extended our causality based formulation of (2a) for each fairness metric in (1) to capture the fairness impact of each preprocessing stage. Similar to [27], the benefit of this formulation is, the measures do not require an oracle, since the prediction equivalence of pipelines $\mathcal{P}$ and $\mathcal{P}^*$ serves the goal of evaluating fairness of the stage. Note that the rest of the definitions in §3.3.2 are independent of (2a) and (2b).

$$I = \begin{cases} 0 & \text{if } \hat{Y}_i(\mathcal{P}) = \hat{Y}_i(\mathcal{P}^*) \\ 1 & \text{otherwise} \end{cases}, \text{for all } i \in \{1 \dots n\}$$

$$\mathsf{SF}(S_k) = M[\hat{Y}(P), \hat{Y}(P^*)]_{I=1} \text{ where } \mathcal{P}^* = \mathcal{P} \setminus S_k \tag{2a}$$

$$\mathsf{SF}(S_k|S'_k) = M[\hat{Y}(P), \hat{Y}(P^*)]_{I=1} \text{ where } \mathcal{P}^* = (\mathcal{P} \setminus S_k) \cup S'_k \tag{2b}$$

3.3.2 Fairness metircs for preprocessing stage. We have leveraged the definition of metrics SPD, EOD, AOD, and ERD from (1) to capture the stage-specific fairness of S_k. Essentially, the new metrics will identify the disparities between $\hat{Y}_i(\mathcal{P})$ and $\hat{Y}_i(\mathcal{P}^*)$ and use corresponding fairness criteria to measure how much S_k favors a specific group with respect to other group(s).

Suppose, among n data instances, n_u are from the unprivileged group and n_p from the privileged group. $\mathsf{SFC_{SPD}}$ computes how many of the data instances have been changed from unfavorable to favorable after applying the stage S_k. To do that we count changes in both directions (unfavorable to favorable and favorable to unfavorable), and take the difference. The sign of $\mathsf{SFC_{SPD}}$ preserves the direction of changes. Finally, the metric $\mathsf{SF_{SPD}}$ is computed by taking the difference of rates ($\mathsf{SFR_{SPD}}$) between unprivileged and privileged groups. Note that the metric captures fairness by measuring the difference of favorable change rates between groups. Simply counting the mismatches between $\hat{Y}_i(\mathcal{P})$ and $\hat{Y}_i(\mathcal{P}^*)$ could provide degree of changes in $\mathsf{SFC_{SPD}}$ but would not capture fairness. Furthermore, computing favorable changes to both groups separately and evaluating the disparity between them captures fairness according to the original definition of SPD.

$$\mathsf{SFC}_{i\mathsf{SPD}} = \begin{cases} 1 & \text{if } \hat{Y}_i(\mathcal{P}) = 1 \text{ and } \hat{Y}_i(\mathcal{P}^*) = 0 \\ -1 & \text{if } \hat{Y}_i(\mathcal{P}) = 0 \text{ and } \hat{Y}_i(\mathcal{P}^*) = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\mathsf{SFC_{SPD}} = \sum_{i=1}^{n} \mathsf{SFC}_{i\mathsf{SPD}}$$

$$\mathsf{SFR_{SPD}}(u) = \mathsf{SFC_{SFD}}(u)/n_u, \mathsf{SFR_{SPD}}(p) = \mathsf{SFC_{SPD}}(p)/n_p$$

$$\mathsf{SF_{SPD}} = \mathsf{SFR_{SPD}}(u) - \mathsf{SFR_{SPD}}(p) \tag{3}$$

Similarly, $\mathsf{SF_{EOD}}$ is defined using the following equation. In this case, only the true-positive changes are considered as suggested by the definition of EOD from (1).

$$\mathsf{SFC}_{i\mathsf{EOD}} = \begin{cases} 1 & \text{if } Y_i = \hat{Y}_i(\mathcal{P}) = 1 \text{ and } \hat{Y}_i(\mathcal{P}^*) = 0 \\ -1 & \text{if } \hat{Y}_i(\mathcal{P}) = 0 \text{ and } Y_i = \hat{Y}_i(\mathcal{P}^*) = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\mathsf{SFC_{EOD}} = \sum_{i=1}^{n} \mathsf{SFC}_{i\mathsf{EOD}}$$

$$\mathsf{SFR_{EOD}}(u) = \mathsf{SFC_{EOD}}(u)/n_{u,Y=1}, \mathsf{SFR_{EOD}}(p) = \mathsf{SFC_{EOD}}(p)/n_{p,Y=1}$$

$$\mathsf{SF_{EOD}} = \mathsf{SFR_{EOD}}(u) - \mathsf{SFR_{EOD}}(p) \tag{4}$$

Since AOD computes the average of true positive (TP) rate and false positive (FP) rate, first the change set for TP and FP predictions is computed. Then averaging the probability of changes for TP and FP, the change rates are computed for both groups. Finally, $\mathsf{SF_{AOD}}$ is calculated by taking the difference of rates between privileged and unprivileged groups.

$$\mathsf{SFC}_{i\mathsf{TP}} = \begin{cases} 1 & \text{if } Y_i = \hat{Y}_i(\mathcal{P}) = 1 \text{ and } \hat{Y}_i(\mathcal{P}^*) = 0 \\ -1 & \text{if } \hat{Y}_i(\mathcal{P}) = 0 \text{ and } Y_i = \hat{Y}_i(\mathcal{P}^*) = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\mathsf{SFC}_{i\mathsf{FP}} = \begin{cases} 1 & \text{if } \hat{Y}_i(\mathcal{P}) = 1 \text{ and } Y_i = \hat{Y}_i(\mathcal{P}^*) = 0 \\ -1 & \text{if } Y_i = \hat{Y}_i(\mathcal{P}) = 0 \text{ and } \hat{Y}_i(\mathcal{P}^*) = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\mathsf{SFC_{TP}} = \sum_{i=1}^{n} \mathsf{SFC}_{i\mathsf{TP}}, \mathsf{SFC_{FP}} = \sum_{i=1}^{n} \mathsf{SFC}_{i\mathsf{FP}}$$

$$\mathsf{SFR_{AOD}}(u) = (1/2)\{\mathsf{SFC_{TP}}(u)/n_{u,Y=1} + \mathsf{SFC_{FP}}(u)/n_{u,Y=0}\}$$

$$\mathsf{SFR_{AOD}}(p) = (1/2)\{\mathsf{SFC_{TP}}(p)/n_{p,Y=1} + \mathsf{SFC_{FP}}(p)/n_{p,Y=0}\}$$

$$\mathsf{SF_{AOD}} = \mathsf{SFR_{AOD}}(u) - \mathsf{SFR_{AOD}}(p) \tag{5}$$

Finally, $\mathsf{SF_{ERD}}$ is computed using the change of count in both false positives (FP) and false negatives (FN) as mentioned in the definition of ERD in (1).

$$\mathsf{SFC}_{i\mathsf{FN}} = \begin{cases} 1 & \text{if } \hat{Y}_i(\mathcal{P}) = 0 \text{ and } Y_i = \hat{Y}_i(\mathcal{P}^*) = 1 \\ -1 & \text{if } Y_i = \hat{Y}_i(\mathcal{P}) = 1 \text{ and } \hat{Y}_i(\mathcal{P}^*) = 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\mathsf{SFC_{FN}} = \sum_{i=1}^{n} \mathsf{SFC}_{i\mathsf{FN}}$$

$$\mathsf{SFR_{ERR}}(u) = \mathsf{SFC_{FP}}(u)/n_{u,Y=0} + \mathsf{SFC_{FN}}(u)/n_{u,Y=1}$$

$$\mathsf{SFR_{ERR}}(p) = \mathsf{SFC_{FP}}(p)/n_{p,Y=0} + \mathsf{SFC_{FN}}(p)/n_{p,Y=1}$$

$$\mathsf{SF_{ERR}} = \mathsf{SFR_{ERR}}(u) - \mathsf{SFR_{ERR}}(p) \tag{6}$$

Thus far, we have four fairness metrics ($\mathsf{SF_{SPD}}$, $\mathsf{SF_{EOD}}$, $\mathsf{SF_{AOD}}$, and $\mathsf{SF_{ERD}}$) to measure the fairness of the stage. In general, the rates computed by each metric (SFR) follow the same range of the original metrics [-1, 1]. Therefore, the above metrics have a range [-2, 2]. Positive values indicate bias towards unprivileged group, negative values indicate bias towards privileged group, and values very close to 0 indicate fair preprocessing stage.

4 EVALUATION

In this section, we describe the benchmark dataset and pipelines that we used for evaluation. Then we present the experiment design and results for answering the research questions.

4.1 Benchmark

We collected ML pipelines used in prior studies for fairness evaluation. First, Biswas and Rajan collected a benchmark of 40 ML models collected from Kaggle that operate on 5 different datasets e.g., *German Credit* [34], *Adult Census* [44], *Bank Marketing* [48], *Home Credit* [38] and *Titanic* [39]. However, the authors did not study the fairness at the component level, rather the ultimate fairness of the classifiers e.g., `RandomForest`, `DecisionTree`, etc. We revisited these Kaggle kernels and collected the preprocessing stages used in the pipelines. We noticed that *Home Credit* dataset [38] in this benchmark is not unified like the other datasets, distributed over multiples CSV files, and the models under this dataset do not operate on the same data files. Hence, these models (8 out of 40) are not suitable for comparing fairness of data preprocessing stages.

Second, we collected the pipelines provided by Yang et al. [70]. The authors released 3 pipelines on two different datasets - *Adult Census* and *Compas*. Third, Zelaya [72] studied the volatility of the preprocessing stages using two pipelines on a fairness dataset i.e., *German Credit*. We included these pipelines in our benchmark. Thus, we created a benchmark of 37 ML pipelines that operate on five datasets. The pipelines with the stages in each dataset category and their performances are shown in Table 1. Below we present a brief description of the datasets and associated tasks.

Table 1: The preprocessing stages and performance measures (accuracy, f1 score) of the pipelines in the benchmark

German	Stages	Acc	F1	Adult	Stages	Acc	F1
GC1	PCA, SB	0.64	0.76	AC1	SS, LE	0.85	0.66
GC2	SMOTE, SS	0.74	0.81	AC2	MV	0.85	0.68
GC3	PCA	0.73	0.83	AC3	Custom(f)	0.87	0.66
GC4	LE, SS	0.73	0.82	AC4	PCA, SS	0.85	0.66
GC5	SS	0.74	0.83	AC5	LE	0.87	0.71
GC6	PCA, SS, LE	0.73	0.83	AC6	Custom(f), Custom(c)	0.85	0.65
GC7	PCA, SB	0.66	0.77	AC7	PCA, SS, Custom(f)	0.78	0.51
GC8	SS	0.72	0.81	AC8	SS, Custom(f), Stratify	0.85	0.67
GC9	SMOTE	0.67	0.77	AC9	SS	0.81	0.61
GC10	Usamp	0.6	0.81	ACP10	Impute	0.81	0.62
Bank	**Stages**	**Acc**	**F1**	**Titanic**	**Stages**	**Acc**	**F1**
BM1	Custom, LE, SS	0.9	0.56	TT1	MV, Custom(f), Encode	0.77	0.83
BM2	LE	0.91	0.61	TT2	MV, Custom(f)	0.78	0.72
BM3	LE, SS, Custom(f)	0.9	0.48	TT3	Custom(f), Impute	0.8	0.72
BM4	SS	0.89	0.33	TT4	Custom(f), Impute, RFECV	0.81	0.73
BM5	SS	0.88	0.23	TT5	Custom(f)	0.83	0.76
BM6	FS, Stratify, SS	0.91	0.58	TT6	Custom(f)	0.82	0.74
BM7	FS	0.91	0.6	TT7	SS, LE, Custom(f)	0.82	0.77
BM8	Stratify	0.9	0.56	TT8	Custom(f)	0.83	0.76
Compas	**Stages**					**Acc**	**F1**
CP1	Filter, Impute1, Encode, Impute2, Kbins, Binarize					0.97	0.97

SB: SelectBest, SS: StandardScaler, LE*: LabelEncoder, Usamp: Undersampling, Custom(f/c): Custom feature engineering or cleaning, FS: Feature selection, MV: Missing value processing, RFECV: Feature selection method

* Fairness is measured with respect to a reference stage.

German Credit. The dataset contains 1000 data instances and 20 features of individuals who take credit from a bank [34]. The target is to classify whether the person has a good/bad credit risk.

Adult Census. The dataset is extracted by Becker [44] from 1994 census of United States. It contains 32,561 data instances and 12 features including demographic data of individuals. The task is to predict whether the person earns over 50K in a year.

Bank Marketing. This dataset contains a bank's marketing campaign data of 41,188 individuals with 20 features [48]. The goal is to classify whether a client will subscribe to a term deposit.

Titanic. The dataset contains information about 891 passengers of Titanic [39]. The task is to predict the survival of the individuals on Titanic. The sensitive attribute of this dataset is *sex*.

Compas. The dataset contains data of 6,889 criminal defendants in Florida. Propublica used this dataset and showed that the recidivism prediction software used in US courts discriminates between White and non-White [5]. The task is to classify whether the defendants will re-offend where *race* is considered as the sensitive attribute.

4.2 Experiment design

Each pipeline in the benchmark consists of one or more preprocessing stages followed by the classifier. In this paper, our main goal is to evaluate the fairness of different preprocessing stages using the fairness metrics described in §3. The benchmark, code and results are released in the replication package [11].

The experiment design for evaluating the pipelines is shown in Figure 2. First, for each pipeline, we identified the preprocessing stages. For example, the pipeline in §2.1 contains six preprocessing stages. To evaluate the fairness of a stage S_k in a pipeline $\mathcal{P}$, we create an alternative pipeline $\mathcal{P}^*$ by removing the stage S_k. For stages that can not be removed, we replaced S_k with a reference stage S'_k. Among the preprocessing stages shown in Table 1, we found only the encoders can not be removed. We experimented with all the encoders in Scikit-Learn library [60], i.e., `OneHotEncoder`, `LabelEncoder`, `OrdinalEncoder`, and found that `OneHotEncoder` does not exhibit any bias. Therefore, we used `OneHotEncoder` as the reference stage for the encoders in our experiment.

Second, the original dataset is split into training (70%) and test set (30%). Then two copies of training data are used to train pipeline $\mathcal{P}$ and $\mathcal{P}^*$. After training the classifiers, two models predict the label for the same set of test data instances. Then, similar to the experimentation of [72], for each prediction label we compare the two predictions $\hat{Y}_i(\mathcal{P})$ and $\hat{Y}_i(\mathcal{P}^*)$ with the true prediction label Y_i. This comparison provides the necessary data to compute the four fairness metrics. Similar to [10, 26], for each stage in a pipeline, we run this experiment ten times, and then report the mean and standard deviation of the metrics, to avoid inconsistency of the randomness in the ML classifiers. Finally, we followed the ML best practices so that noise is not introduced evaluating the fairness of preprocessing stages. For example, while applying some data transformation, lack of data isolation might introduce noise in the evaluation, e.g., when applying `PCA` on dataset, it is important to train the `PCA` only using the training data. If we use the whole dataset to train the `PCA` and then transform data, then information from the test-set might leak, and transformation might not be different than expected. Third, since a stage operates on the data processed by the preceding stage(s), there are interdependencies between them. We always maintained the order of the stages while removing or replacing a stage (§5). Also, to observe fairness of data transformers without interdependencies, we applied them on vanilla pipelines (§6).

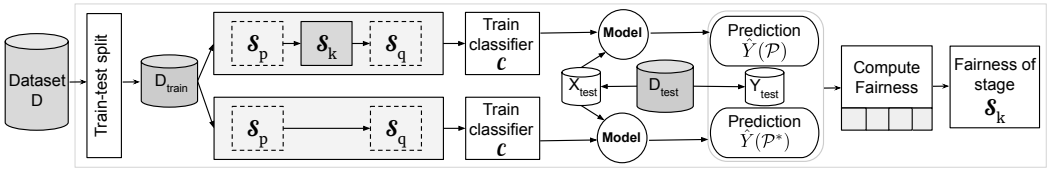

Figure 2: Experiment design to measure fairness of preprocessing stages in machine learning pipeline.

5 FAIRNESS OF PREPROCESSING STAGES

In this paper, we used a diverse set of metrics, to evaluate fairness of preprocessing stages. While developing an ML pipelines, if the developer has a comprehensive idea of the fairness of preprocessing stages, it would be convenient to build a fair pipeline. The evaluation has been done on 69 preprocessing stages in 37 ML pipelines from 5 dataset categories. For *Compas* dataset, we found one pipeline (§2.1). Five out of six stages in this pipeline exhibit no bias, which has been discussed later. For other 4 dataset categories, the evaluation has been shown in Figure 3. In this section, first, we discuss how we can interpret the metrics. Second, we answer the first research question and discuss the findings from our evaluation.

5.1 What do the metrics imply?

We investigated the fairness of the preprocessing stages using four metrics: SF_{SPD}, SF_{EOD}, SF_{AOD}, and SF_{ERD}. These metrics measure the fairness of the stages by using the existing fairness criteria, e.g., SF_{SPD} measures the fairness of a stage with respect to statistical parity difference (SPD) criteria. These fairness criteria evaluate algorithmic fairness of ML pipelines [15, 25, 71]. The unfairness characterized by these criteria is measured based on the prediction disparities, although the root cause can be the training data or the algorithm (e.g., data preprocessing, classifier) itself. Therefore, when an ML model is identified as unfair, it implies that in the given predictive scenario, the outcome is biased. Similarly, the metrics proposed in this paper measure algorithmic unfairness caused by a specific preprocessing stage with respect to its pipeline. For instance, in Figure 3, pipeline GC4 has two stages: `LabelEncoder` and `StandardScaler`. The fairness metrics suggest that `LabelEncoder` is biased towards unprivileged group (positive value), and `StandardScaler` is biased towards privileged group (negative value). The stages for which the measures are very close to zero, can be considered as fair preprocessing.

The metrics can provide different fairness signals for a certain stage. For example, in AC4, SF_{ERD} shows positive fairness, whereas the other metrics suggest negative fairness for both the stages `PCA` and `StandardScaler`. This disparity occurs because different metrics accounts for different fairness criteria. In this case, SF_{ERD}, depends on the false positive and false negative rate difference. No other metric is concerned about the false negative rate difference, and hence SF_{ERD} provides a different fairness signal than other metrics. In practice, appropriate fairness criteria can vary depending on the task, usage scenario, or involved stakeholders. Study suggests that developers need to be aware of different fairness indicators to build fairer pipelines [26]. Therefore, we defined and evaluated fairness of stages with respect to multiple metrics.

5.2 Fairness analysis of stages

The pipelines used both built-in algorithm imported from libraries i.e., *data transformers* [14], as well as *custom* preprocessing stages. The stages found in each pipeline are shown in Table 1, and the fairness measures of those stages are plotted in Figure 3. Although the unfairness exhibited by a stage is with respect to the pipeline, we found fairness patterns of some stages and investigated them further. In general, our findings show that the stages which change the underlying data distribution significantly, or modify minority data are responsible for increasing bias in the pipelines.

Finding 1: *Data filtering and missing value removal change the data distribution and hence introduce bias in ML pipeline.*

Most of the real-world datasets contain missing values (MV) for several reasons such as data creation errors, not-applicable (N/A) attributes, incomplete data collection, etc. In our benchmark, *Adult Census* and *Titanic* contain MV that required further processing in the pipeline. 7.4% rows in *Adult Census* and 20.2% rows in *Titanic* have at least one missing feature in the dataset. The pipelines either remove the rows with MV or apply certain imputation [61] technique that replaces the MV with mean, median or most frequently occurred values. Removal of rows with MV can significantly change the data distribution, which introduces bias in the pipeline. For example, both TT1 and TT2 removed data items with MV by applying `df.dropna()` method, which introduces bias in the prediction (Figure 3). Research has shown that MV are not uniformly distributed over all groups and data items from minority groups often contain more MV [22]. If those data items are entirely removed, the representation of minority groups in the dataset becomes scarce. On the other hand, TT3 applied mean-imputation and TT4 applied both median- and mode-imputation using `df.fillna()`, which exhibits fairness compared to data removal. While our findings suggest that removing data items with MV introduces bias, the most popular fairness tools AIF 360 [8], Aequitas [57], Themis-ML [6] ignore these data items and remove entire row/column. Our evaluation strategy confirms that the tools can integrate existing imputation methods [61] in the pipeline and allow users to choose appropriate ones. Additionally, more research is needed to understand and develop imputation techniques that are fairness aware.

Finding 2: *New feature generation or feature transformation can have large impact on fairness.*

We found that most of the feature engineering stages, especially the custom transformations exhibit bias in the pipeline. For example, the pipelines in *Titanic* dataset used custom feature engineering, since the dataset contains composite features which may provide additional information about the individuals. For instance, TT8 operates on the feature *name* to create a new feature *title* e.g., Mr,

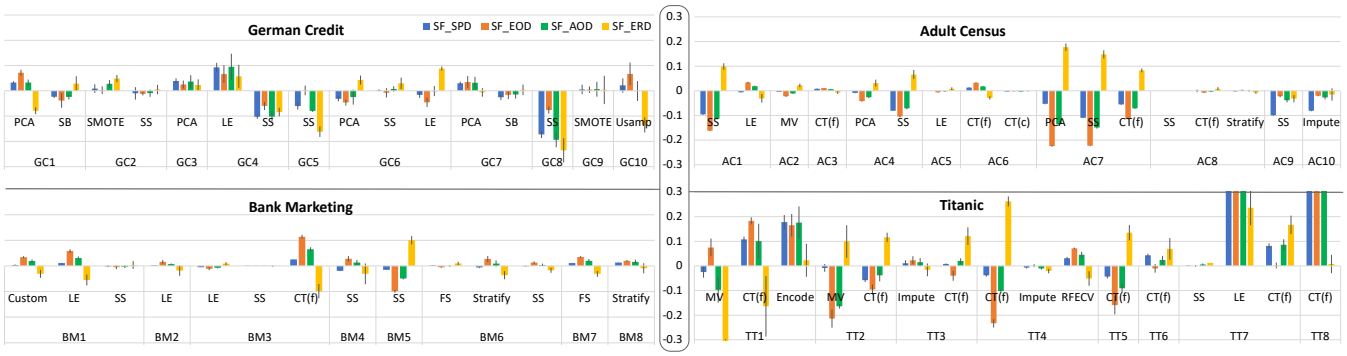

Figure 3: Fairness measures (Y-axis) of preprocessing stages in pipelines (X-axis). Grey lines above bars indicate standard error.

Mrs, Dr, etc. This transformation aims at better prediction of the survival of passengers by extracting the social status, but creates high bias between male and female.

In *Adult Census*, the feature *eduction* of individuals contain values such as *preschool*, *10th*, *1st-4th*, *prof-school*, etc., which have been replaced by broad categories such *Dropout*, *HighGrad*, *Masters* in the pipeline AC7. In addition, instead of using *age* as continuous value, the feature has been discretized into n number of bins. In both cases, the original data values have been modified, which has caused unfairness in the pipeline. Nevertheless, some pipelines (AC3, AC8) have custom feature transformations that are fair. Previous studies showed that certain features contribute more to the predictive quality of the model [28, 55]. Feature importance in prediction and corelation of features with the sensitive attribute also led to bias detection [16, 31] in ML models. However, does creating new features (by removing certain semantics) from a potentially biased feature increase the fairness, is an open question. Our method to quantify the fairness of such changes can guide further research in this direction.

Finding 3: *Encoding techniques should be chosen cautiosly based on the classifier.*

Two most used encoding techniques for converting categorical feature to numerical feature are `OneHotEncoder` and `LabelEncoder`. `OneHotEncoder` creates n new columns by replacing one column for each of the n categories. `LabelEncoder` does not increase the number of the columns, and gives each category an integer label between 0 and $(n-1)$. In our evaluation, we found that `LabelEncoder` introduces bias in *German Credit* and *Titanic* dataset but `OneHotEncoder` does not change fairness. Since `LabelEncoder` imposes a sequential order between the categories, it might create a linear relation with the target value, and hence have an impact on the classifier to change fairness. For example, pipelines TT7 creates a new feature called *Family* based on the surname of the person. This feature has a large number of unique categories (667 unique ones in 891 data instances). Therefore, the non-sparse representation in *LabelEncoder* adds additional weight to the feature, which is causing unfairness in TT7. Developers might avoid `OneHotEncoder` because it suffers from the curse of dimensionality and the ordinal relation of data is lost. In that case, developers should be aware of the fairness impact of the encoder. One solution might be using PCA for dimensionality reduction, which has been done in GC7.

Finding 4: *The variability of fairness of preprocessing stages depend on the dataset size and overall prediction rate of the pipelines.*

We have plotted the standard error of the metrics as error bars in Figure 3. Firstly, it shows that the metrics in *German Credit* and *Titanic* dataset are more unstable. The reason is that the size of these two datasets is less than the other three datasets. *German Credit* dataset has 1000 instances, and *Titanic* has 891 instances. *Adult Census* and *Bank Marketing* dataset have more than 30K instances. If the sample size is large, data distribution tends to be similar even after taking a random train-test split [26]. However, when the dataset size is smaller, the distribution is changed among different train-test splitting. Furthermore, we have found that $\mathrm{SF}_{\mathrm{ERD}}$ is more unstable than other metrics. $\mathrm{SF}_{\mathrm{ERD}}$ depends on the change of false positive and false negative rates. However, in most cases, the pipelines are optimized for accuracy and precision, since these are some best performing ones collected from Kaggle. Therefore, before deploying preprocessing stages, it would be desirable to test the stability of over multiples executions.

Finding 5: *The unfairness of a preprocessing stage can be dominated by dataset or the classifier used in the pipeline.*

For the *Compas* dataset, we evaluated the six stages shown in §2.1. All the stages exhibited data filtering show bias. The data filtering also showed bias close to zero (less than .005) with respect to all the metrics. Although Yang *et al.* [70] argued that this pipeline filters data in different proportions from *male* and *female* group, our evaluation confirms it does not cause unfairness. This pipeline has been used by Propublica [5] to show the bias in the prediction. Therefore, it is understandable that they did not employ any preprocessing that introduces bias in the pipelines. Other than that, almost all the preprocessing stages in *Bank Marketing* pipelines also exhibit very little unfairness, which suggest that the preprocessing on this dataset are fair in general.

A few stages show different behavior when they are used in composition with different classifiers. For example, `StandardScaler` has been applied on both GC6 and GC8. While GC6 employs a *RandomForest* classifier, GC8 uses *K-Neighbors* classifier. We have observed the opposite fairness measures for *StandardScaler* in these two pipelines. Therefore, fairness can be dominated by the underlying properties of data or the pipeline where it is applied. We have further investigated this phenomenon by applying transformers on different classifiers in the next section.

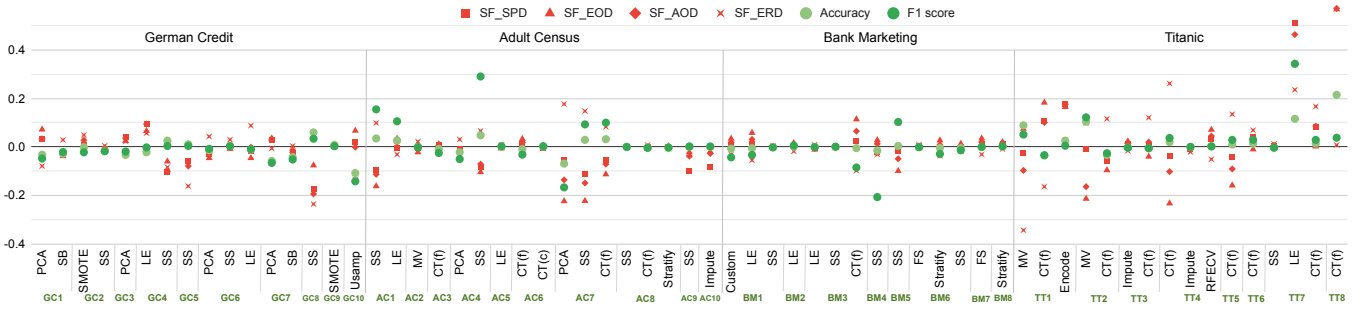

Figure 4: Performance changes (green) are plotted with fairness (red) of the preprocessing stages. A positive or negative performance change indicates performance increase or decrease respectively, after applying the stage.

5.3 Fairness-Performance Tradeoff

In this section, we investigated the fairness-performance tradeoff for the preprocessing stages. The original performances of each pipeline have been reported in Table 1. To investigate fairness of a stage, we created pipeline $\mathcal{P}^*$ by removing the stage from original pipeline $\mathcal{P}$ (§2). To understand fairness-performance tradeoff, we evaluated performance (both accuracy and f1 score) of $\mathcal{P}^*$ and $\mathcal{P}$ in the same experimental setup. Then we computed the performance difference to observe the impact of the stage on performance. For example, $Acc(\mathcal{P}) - Acc(\mathcal{P}^*)$ gives the accuracy increase (or decrease, if negative) after applying a stage. We plotted the performance impacts of the stages with their fairness measures in Figure 4.

First, many preprocessing stages have negligible performance impact. In Figure 4, 19 out 63 stages exhibits accuracy and f1 score change in the range [-0.005, 0.005], which indicates that the performance change is not more than 0.05%. We found that in all of these cases, except AC9 and AC10, the preprocessing stages are fair with a very small degree of bias. Second, tradeoff between performance and fairness is observed for the stages which improve performance. 17 stages improve accuracy or f1 score more than 0.05%, which further exhibits moderate to high degree of bias. Overall, the most biased stages - TT7(LE), TT8(CT), TT4 (CT), TT1 (MV), GC8 (SS), are improving performance of the pipelines. This stage-specific tradeoff is aligned with the overall performance-fairness tradeoff of ML models discussed in prior work [10, 17, 26]. Third, we found that some stages decrease the performance, either accuracy or f1 score. Surprisingly, most of these stages also exhibit high degree of bias. For instance, the most performance-decreasing stages - BM4 (SS), AC7 (PCA), GC10 (Undersampling), are showing high bias comparatively. The fairness and performance evaluation together would facilitate developers to identify such stages in the pipeline.

6 FAIR DATA TRANSFORMERS

In §5, we found that many data preprocessing stages are biased. Many bias mitigation techniques in data preprocessing have been shown successful in the literature [25, 40]. If we process data with appropriate transformer, then it might be possible to avoid bias and mitigate inherent bias in data or classifier. Even if a data transformer is biased towards a specific group, it could be useful to mitigate bias if original data or model exhibits bias towards the opposite group. To that end, we want to investigate the fairness pattern of the data transformers. However, in our evaluation (Figure 3), some transformers have been used only in specific situations e.g., `SMOTE` has been only applied on *German Credit* dataset. What is the fairness of this transformer when used on other datasets and classifier? In this section, we setup experiments to evaluate the fairness of commonly used data transformers on different datasets and classifiers.

Table 2: Transformers collected from pipelines and libraries

Categories	Stages	Transformers
MV processing	Imputation	SimpleImputer, IterativeImputer
Categorical encoding	Encoder	Binarizer, KBinsDiscretizer, LabelBinarizer, LabelEncoder, OneHotEncoder
Standardization	Scaling	StandardScaler, MaxAbsScaler, MinMaxScaler, RobustScaler
	Normalization	l1-normalizer, l2-normalizer
Feature engineering	Non-linear transformation	QuantileTransformer, PowerTransformer
	Polynomial feature generation	PCA, SparsePCA, MiniBatchSparsePCA, KernelPCA
	Feature selection	SelectKBest, SelectFpr, SelectPercentile
Sampling	Oversampling	SMOTE
	Undersampling	AllKNN
	Stratification	Random, Stratified

First, we collected the classifiers used in each dataset category from the benchmark. Then, for each dataset, we created a set of vanilla pipelines. A vanilla pipeline is a classification pipeline which contain only one classifier. Second, we found a few categories of preprocessing stages from our benchmark shown in Table 2. For each transformer used in each stage, we collected the alternative transformers from corresponding library. For example, in our benchmark, `StandardScaler` from Scikit-Learn library has been used for scaling data distribution in many pipelines. We collected other standardizing algorithms available in Scikit-Learn. We found that besides `StandardScaler`, Scikit-Learn also provides `MaxAbsScaler`, `MinMaxScaler`, and `Normalizer` standardize data [60]. Similarly, a data oversampling technique `SMOTE` has been used in the benchmark, we collected another undersampling technique `ALLKNN` and a combination of over- and undersampling sampling technique `SMOTENN` from IMBLearn library [47]. Third, in each of the vanilla pipelines, we applied the transformers and evaluated fairness using the method used in §4.2 with respect to four metrics. We found that pipelines under *Titanic* uses custom transformation, and most of the built-in transformers are not appropriate for this dataset. So, to be able to make the comparison consistent, we conducted this evaluation on four datasets: *German Credit*, *Adult Census*, *Bank Marketing*, *Compas*. Finally, we did not use transformers for imputation and encoder stages. Encoding transformers (LabelEncoder,

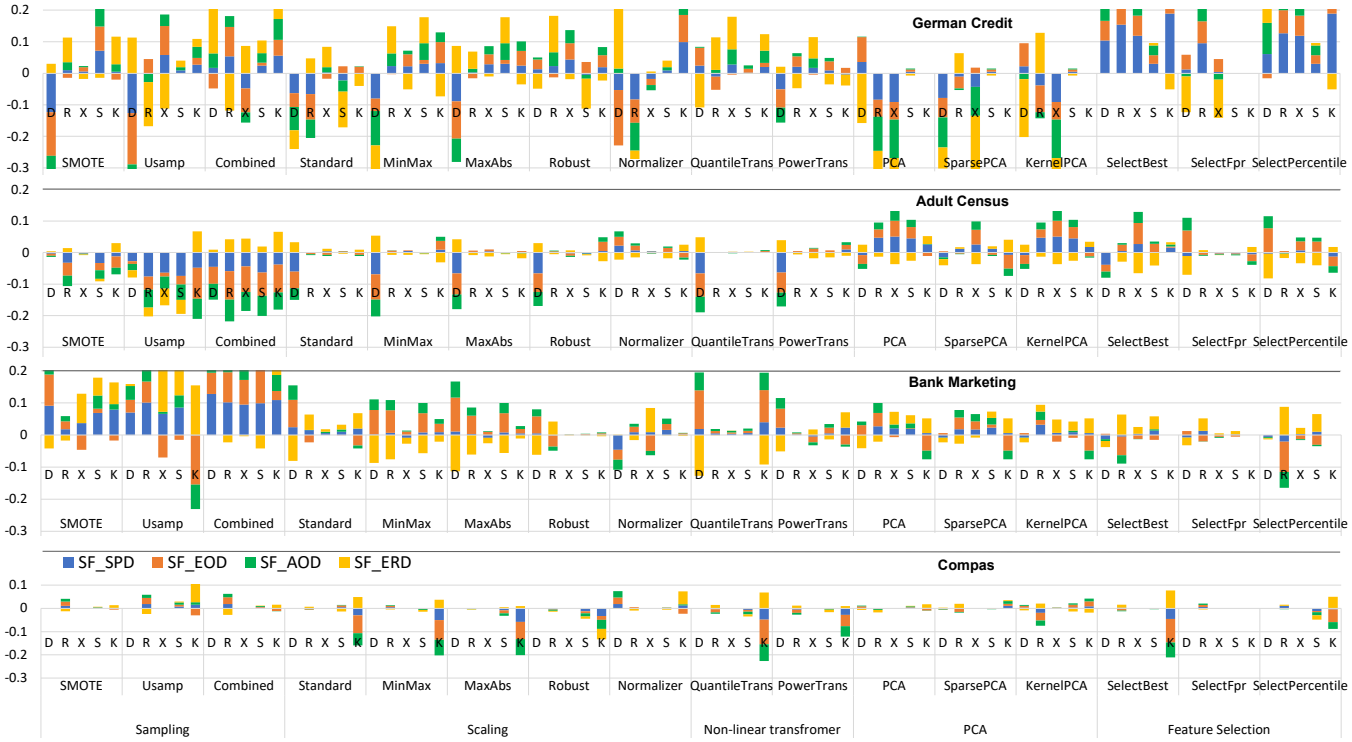

Figure 5: Fairness of transformers on classifiers, D: `DecisionTree`, R: `RandomForest`, X: `XGBoost`, S: `SupportVector`, K: `KNeighbors`

OneHotEncoder), have been applied on most of the pipelines and their behavior has been understood. The fairness measures of each transformer on different classifiers have been plotted in Figure 5.

Fairness among the datasets follows a similar pattern. This further confirms that the unfairness is rooted in data. The *Compas* dataset shows the least bias. Although racial discrimination has been reported for this dataset [5], this is a more curated dataset than the other three. By looking at the overall trend of fairness, we observe that sampling techniques have the most biased impact on prediction. Other than that feature selection transformers have more impact than other ones.

Finding 6: *Among all the transformers, applying sampling technique exhibits most unfairness.*

Sampling techniques are often used in ML tasks when dataset is class-imbalanced. Unlike the other transformers, sampling techniques make horizontal transformation to the training data. The oversampling technique `SMOTE` creates new data instances for the minority class by choosing the nearest data points in the feature space. Undersampling techniques balance dataset by removing data items from majority class. Although balancing dataset has been shown to increase fairness [22], our evaluation suggest that in three out of four datasets, it increases bias.

From Figure 5, we can see that sampling techniques exhibit the most unfairness. In *German Credit* dataset, different classifier reacts differently when sampling is done. `DecisionTree` classifier exhibits most unfairness for both oversampling and undersampling towards privileged group i.e., *male*. Interestingly, the combination of over- and undersampling also fails to show fairness. Furthermore, both *German Credit* and *Bank Marketing* pipelines exhibit bias towards unprivileged group, which might be desired when compared to bias towards privileged.

Finding 7: *Selecting subset of features often increase unfairness.*

Selecting the best performing feature can give performance improvement of the pipeline. However, unfairness can be encoded in specific features [31]. While selecting best features, some features which encodes unfairness, can dominate the outcome. Thus, many classifiers in *German Credit*, *Adult Census*, and *Bank Marketing* show unfairness. Surprisingly, `SelectFpr` exhibited very little or no bias compared to the other feature selection methods. A detailed investigation suggests that `SelectBest` and `SelectPercentile` select only the k most contributing features. However, `SelectFpr` performs false positive rate test on each feature, and if it falls below a threshold, the feature is removed [59]. Therefore, it does not apply harsh pruning, which contributes to the fairness of the prediction.

Finding 8: *In most of the pipelines, feature standardization and non-linear transformers are fair transformers.*

These transformers modify the mean and variance of the data by applying linear or non-linear transformation. However, they do not change the feature importance on the classifiers. Therefore, in most of the cases, these transformers (especially, `StandardScaler` and `RobustScaler`) are fair. Some classifiers show bias after applying these transformers such as, KNC in *Compas*. The unfairness exhibited by those pipelines are introduced by the classifiers, since these classifiers show similar bias pattern for other transformers as well. The scalers can impact the fairness significantly if there

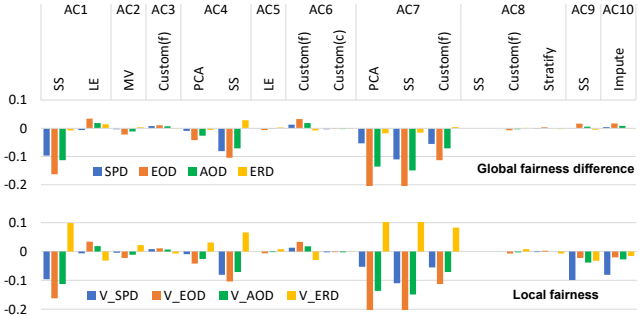

Figure 6: Comparison of global fairness change and local fairness for *Adult Census* dataset pipelines.

are many outliers in data. That is why we see more bias for the scalers in *German Credit* dataset. Therefore, although standardizing transformers are fair in general, they can be biased in composition with specific classifier or data property.

7 FAIRNESS COMPOSITION OF PREPROCESSING STAGES

From our evaluation, we found that many data transformers have fairness impact on ML pipeline. In this section, we compare the *local fairness* (fairness measures of preprocessing stages) with the *global fairness* (fairness measures of whole pipeline). First, we answer whether the local fairness composes in the global fairness. Second, we investigate if we can leverage the composition to mitigate bias by choosing appropriate transformers.

7.1 Composition of local and global fairness

We evaluated the global fairness of *Adult Census* pipelines (Table 1) using the four existing metrics from (1). We calculated the fairness difference of these pipelines before and after applying the preprocessing stages. Additionally, we have evaluated the stage-specific fairness metrics. Both the local fairness and difference in global fairness of those pipelines have been plotted in Figure 6.

We can see that local and global fairness follow the same trend in most of the pipelines. This confirms that local fairness is directly contributing to the global fairness. However, the global fairness is computed based on the overall change in the prediction, whereas the local fairness considers the predictions for only those data instances which have been altered after applying a transformer (3). For example, in Figure 6, for some pipelines (e.g., AC9, AC10), global and local fairness exhibit different trends. In these cases, the overall classification rate difference is not similar to the rate difference of altered labels. This means that the stages changed the labels such that it shows bias towards privileged. But when those changes in the labels are considered in addition to all the labels (global fairness), the bias difference could not capture the actual impact of that stage. We have verified this observation by manually inspecting the altered prediction labels. Thus, we can conclude that the local fairness composes to the global fairness. Specifically, if a preprocessing stage shows bias for privileged group, it pulls the global fairness towards the fairness direction of privileged group. However, only observing the global fairness difference, we can not measure the fairness of a given stage or transformer.

7.2 Fairness mitigation using appropriate transformers

For a given transformer in an ML pipeline, a *downstream* transformer operates on data already processed by the given transformer and an upstream transformer is applied before the given one. Since the fairness of a preprocessing stage composes to the global fairness, can we choose a downstream transformer to mitigate bias in ML pipeline? In this section, we empirically show that the global unfairness can be mitigated by choosing the appropriate downstream transformer.

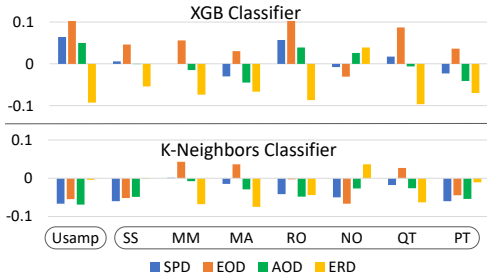

Figure 7: Global fairness after applying the upstream transformer (left), and after applying both the upstream and one downstream transformer (right). Usamp: undersampling.

Consider the use-case of classification task on *German Credit* dataset with different classifiers similar to Figure 5. Suppose, the original pipeline is constructed using undersampling technique. Since this pipeline exhibits bias, as shown in Figure 5, can we choose a downstream data standardizing transformer that mitigates that bias? In this use case, undersampling is the upstream transformer, and any standardizing transformer is the downstream transformer.

We showed the evaluation for XGB classifier and KNC classifier, since these two exhibits most bias when the upstream transformer was applied in §6. We plotted the global fairness after applying only the upstream transformer in the left of Figure 7. We also reported the local fairness of the standardizing transformers in Table 3. Now, since undersampling method exhibits bias towards privileged group for XGB, we look for the transformer that is biased towards privileged group. In Figure 7, among other transformers, `Normalizer` is the most successful to mitigate bias of the upstream transformer. Similarly, for KNC, the upstream operator exhibits bias towards privileged group. From Table 3, we can see that `MinMaxScaler` is the most biased transformer towards the opposite direction. As a result, applying `MinMaxScaler` mitigates bias the most. Note that the other downstream transformers also follow the fairness composition with its upstream transformer. Therefore, by measuring fairness of the preprocessing stages, developers would be able to instrument the biased transformers and build fair ML pipelines.

8 DISCUSSION

We took the first step to understand the fairness of components in ML pipelines. Our method helps to provide causality in software and reason about behavior of components based on the impact on outcome. This method can be extended further to evaluate the fairness of other software modules [51] in ML pipeline and localize faults [69]. Moreover, we found most of the stages exhibited bias, to a low

Table 3: Local fairness of stages as downstream transformer.

Model	Stage	SF_SPD	SF_EOD	SF_AOD	SF_ERD
XGB	SS	0.001	0.021	-0.016	-0.073
	MM	-0.006	0.031	-0.029	-0.12
	MA	-0.036	0.005	-0.059	-0.128
	RO	0.051	0.082	0.025	-0.113
	NO	-0.014	-0.056	0.012	0.135
	QT	0.011	0.062	-0.02	-0.165
	PT	-0.029	0.011	-0.055	-0.132
KNC	SS	0.035	0.025	0.055	0.059
	MM	0.095	0.12	0.096	-0.05
	MA	0.079	0.114	0.075	-0.078
	RO	0.052	0.074	0.056	-0.036
	NO	0.045	0.010	0.077	0.134
	QT	0.077	0.104	0.078	-0.052
	PT	0.035	0.032	0.050	0.036

SS: StandardScaler, MM: MinMaxScaler, MA: MaxAbsScaler, RO: RobustScaler, NO: Normalizer, QT: QuantileTransformer, PT: PowerTransformer

or higher degree. The fairness measures of different components can be leveraged towards fairness-aware pipeline optimization to satisfy fairness constraints. For example, US Equal Employment Commission suggests selection-rate difference between groups less than 20% [67]. Also, pipeline optimization techniques, e.g., TPOT [50], Lara [45] can be potentially utilized for pipeline optimization.

Furthermore, research has been conducted to understand the impact of preprocessing stages with respect to performance improvement [18, 20, 68]. This paper will open research directions to develop preprocessing techniques that improve performance by keeping the fairness intact. We also reported a number of fairness patterns of preprocessing stages that inducing bias in the pipeline such as missing value processing, custom feature generation, feature selection. Moreover, instrumentation of the stages can mitigate the inherent bias of the classifiers. It shows opportunities to build automated tools for identifying fairness bugs in AI systems and recommending fixes [36, 37]. Finally, current fairness tools (e.g., AIF 360 [8], Aequitas [57]) can be augmented by incorporating data preprocessing stages into the pipelines and letting users have control over the data transformers and observe or mitigate bias. Similarly, the libraries can provide API support to monitor fairness of the transformers.

9 THREATS TO VALIDITY

Internal validity refers to whether the fairness measures used in this paper actually captures the fairness of preprocessing stages. To mitigate this threat, we used existing concepts and metrics to build new set of metrics. Causality in software [52, 53] has been well-studied, and causal reasoning in fairness has also been popular [46, 56, 58, 75], since it can provide explanation with respect to change in the outcome. Besides, this method do not require an oracle because the prediction equivalences provide necessary information to measure the impact of the intervention [27]. Furthermore, in §7, we conducted experiments on local and global fairness to show how new metrics composes in the pipeline.

External validity is concerned about the extent the findings of this study can be generalized. To alleviate this threat, we conducted experiments on a large number of pipeline variations. We collected the pipelines from three different sources. Moreover, we collected alternative transformers from the ML libraries for comparative analysis. Finally, for the same dataset categories, we used multiple classifiers and fairness metrics so that the findings are persistent.

10 RELATED WORKS

Fairness in ML Classification. The machine learning community has defined different fairness criteria and proposed metrics to measure the fairness of classification tasks [15, 19, 22, 23, 25, 32, 53, 64, 71, 73]. Following the measurement of fairness in ML models, many mitigation techniques have also been proposed to remove bias [15, 19, 22, 25, 29, 32, 40–42, 54, 71, 74]. This body of work mostly concentrates on the theoretical aspect of fairness in a single classification task. Recently, software engineering community has also focused on the fairness in ML, mostly on fairness testing [3, 27, 65, 66]. These works propose methods to generate appropriate test data inputs for the model and prediction on those inputs characterizes fairness. Some research has been conducted to build automated tools [2, 63, 66] and libraries [8] for fairness. In addition, empirical studies have been conducted to compare, contrast between fairness aspects, interventions, tradeoffs, developers concerns, and human aspects of fairness [10, 26, 33, 35].

Fairness in Composition. Dwork and Ilvento argued that fairness is dynamic in a multi-component environment [24]. They showed that when multiple classifiers work in composition, even if the classifiers are fair in isolation, the overall system is not necessarily fair. Bower *et al.* discussed fairness in ML *pipeline*, where they considered *pipeline* as sequence of multiple classification tasks [12]. They also showed that when decisions of fair components are compounded, the final decision might not be fair. For example, while interviewing candidates in two stages, fair decision in each stage may not guarantee a fair selection. D'Amour *et al.* studied the dynamics of fairness in multi-classification environnement using simulation [21]. In these research, fairness composition is shown over multiple tasks and the authors did not consider fairness of components in single ML pipeline. We position our paper here to study the impact of preprocessing stages in ML pipeline and evaluate the fairness composition.

11 CONCLUSION

Data preprocessing techniques are used in most of the machine learning pipelines in composition with the classifier. Studies showed that fairness of machine learning predictions depends largely on the data. In this paper, we investigated how the data preprocessing stages affect fairness of classification tasks. We proposed the causal method and leveraged existing metrics to measure the fairness of data preprocessing stages. The results showed that many stages induce bias in the prediction. By observing fairness of these data transformers, fairer ML pipelines can be built. In addition, we showed that existing bias can be mitigated by selecting appropriate transformers. We released the pipeline benchmark, code, and results to make our techniques available for further usages. Future research can be conducted towards developing automated tools to detect bias in ML pipeline stages and instrument that accordingly.

ACKNOWLEDGMENTS

This work was supported in part by US NSF grants CNS-15-13263, CCF-19-34884, and Facebook Probability and Programming Award. We also thank the reviewers for their insightful comments. All opinions are of the authors and do not reflect the view of sponsors.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*. 265–283.

[2] Julius Adebayo and Lalana Kagal. 2016. Iterative orthogonal feature projection for diagnosing bias in black-box models. *arXiv preprint arXiv:1611.04967* (2016).

[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] Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. 2019. Software engineering for machine learning: A case study. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 291–300.

[5] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. 2016. Machine bias: There's software used across the country to predict future criminals. *And it's biased against blacks. ProPublica* (2016). https://github.com/propublica/compas-analysis

[6] Niels Bantilan. 2018. Themis-ml: A fairness-aware machine learning interface for end-to-end discrimination discovery and mitigation. *Journal of Technology in Human Services* 36, 1 (2018), 15–30.

[7] Denis Baylor, Eric Breck, Heng-Tze Cheng, Noah Fiedel, Chuan Yu Foo, Zakaria Haque, Salem Haykal, Mustafa Ispir, Vihan Jain, Levent Koc, et al. 2017. TFX: A tensorflow-based production-scale machine learning platform. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1387–1395.

[8] 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).

[9] Reuben Binns. 2017. Fairness in machine learning: Lessons from political philosophy. *arXiv preprint arXiv:1712.03586* (2017).

[10] Sumon Biswas and Hridesh Rajan. 2020. Do the Machine Learning Models on a Crowd Sourced Platform Exhibit Bias? An Empirical Study on Model Fairness. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Virtual Event, USA). 642–653. https://doi.org/10.1145/3368089.3409704

[11] Sumon Biswas and Hridesh Rajan. 2021. Replication Package for "Fair Preprocessing: Towards Understanding Compositional Fairness of Data Transformers in Machine Learning Pipeline". (2021). https://github.com/sumonbis/FairPreprocessing

[12] Amanda Bower, Sarah N Kitchen, Laura Niss, Martin J Strauss, Alexander Vargas, and Suresh Venkatasubramanian. 2017. Fair pipelines. *arXiv preprint arXiv:1707.00391* (2017).

[13] Yuriy Brun and Alexandra Meliou. 2018. Software fairness. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 754–759.

[14] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, et al. 2013. API design for machine learning software: experiences from the scikit-learn project. *arXiv preprint arXiv:1309.0238* (2013).

[15] Toon Calders and Sicco Verwer. 2010. Three naive Bayes approaches for discrimination-free classification. *Data Mining and Knowledge Discovery* 21, 2 (2010), 277–292.

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

[17] Joymallya Chakraborty, Tianpei Xia, Fahmid M Fahid, and Tim Menzies. 2019. Software engineering for fairness: A case study with hyperparameter optimization. *arXiv preprint arXiv:1905.05786* (2019).

[18] Priyanga Chandrasekar and Kai Qian. 2016. The impact of data preprocessing on the performance of a naive bayes classifier. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, Vol. 2. IEEE, 618–619.

[19] Alexandra Chouldechova. 2017. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big data* 5, 2 (2017), 153–163.

[20] Sven F Crone, Stefan Lessmann, and Robert Stahlbock. 2006. The impact of preprocessing on data mining: An evaluation of classifier sensitivity in direct marketing. *European Journal of Operational Research* 173, 3 (2006), 781–800.

[21] Alexander D'Amour, Hansa Srinivasan, James Atwood, Pallavi Baljekar, D Sculley, and Yoni Halpern. 2020. Fairness is not static: deeper understanding of long term fairness via simulation studies. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. 525–534.

[22] 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.

[23] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. 2012. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*. 214–226.

[24] Cynthia Dwork and Christina Ilvento. 2018. Fairness under composition. *arXiv preprint arXiv:1806.06122* (2018).

[25] 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.

[26] 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.

[27] Sainyam Galhotra, Yuriy Brun, and Alexandra Meliou. 2017. Fairness testing: testing software for discrimination. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. 498–510.

[28] Damien Garreau and Ulrike Luxburg. 2020. Explaining the explainer: A first theoretical analysis of LIME. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 1287–1296.

[29] 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.

[30] Noah J Goodall. 2016. Can you program ethics into a self-driving car? *IEEE Spectrum* 53, 6 (2016), 28–58.

[31] Nina Grgic-Hlaca, Muhammad Bilal Zafar, Krishna P Gummadi, and Adrian Weller. 2018. Beyond Distributive Fairness in Algorithmic Decision Making: Feature Selection for Procedurally Fair Learning.. In *AAAI*. 51–60.

[32] Moritz Hardt, Eric Price, and Nati Srebro. 2016. Equality of opportunity in supervised learning. In *Advances in neural information processing systems*. 3315–3323.

[33] 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.

[34] Dr. Hans Hofmann. 1994. German Credit Dataset: UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data)

[35] 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.

[36] Md Johirul Islam, Giang Nguyen, Rangeet Pan, and Hridesh Rajan. 2019. A Comprehensive Study on Deep Learning Bug Characteristics. In *ESEC/FSE'19: The ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE) (ESEC/FSE 2019)*.

[37] Md Johirul Islam, Rangeet Pan, Giang Nguyen, and Hridesh Rajan. 2020. Repairing Deep Neural Networks: Fix Patterns and Challenges. In *ICSE'20: The 42nd International Conference on Software Engineering* (Seoul, South Korea).

[38] Kaggle. 2017. Home Credit Dataset. https://www.kaggle.com/c/home-credit-default-risk.

[39] Kaggle. 2017. Titanic ML Dataset. https://www.kaggle.com/c/titanic/data.

[40] Faisal Kamiran and Toon Calders. 2012. Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems* 33, 1 (2012), 1–33.

[41] 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.

[42] 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.

[43] Keith Kirkpatrick. 2017. It's not the algorithm, it's the data. *Commun. ACM* 60, 2 (2017), 21–23.

[44] Ron Kohavi. 1996. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid.. In *KDD*, Vol. 96. 202–207. https://archive.ics.uci.edu/ml/datasets/adult

[45] Andreas Kunft, Asterios Katsifodimos, Sebastian Schelter, Sebastian Breß, Tilmann Rabl, and Volker Markl. 2019. An intermediate representation for optimizing machine learning pipelines. *Proceedings of the VLDB Endowment* 12, 11 (2019), 1553–1567.

[46] Matt Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. 2017. Counterfactual fairness. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 4069–4079.

[47] Guillaume Lemaître, Fernando Nogueira, and Christos K Aridas. 2017. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *The Journal of Machine Learning Research* 18, 1 (2017), 559–563.

[48] Sérgio Moro, Paulo Cortez, and Paulo Rita. 2014. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems* 62 (2014), 22–31. https://archive.ics.uci.edu/ml/datasets/Bank+Marketing

[49] P Olson. 2011. The algorithm that beats your bank manager. *CNN Money* (2011). https://www.forbes.com/sites/parmyolson/2011/03/15/the-algorithm-that-beats-your-bank-manager/

[50] Randal S Olson and Jason H Moore. 2016. TPOT: A tree-based pipeline optimization tool for automating machine learning. In *Workshop on automatic machine learning*. PMLR, 66–74.

[51] Rangeet Pan and Hridesh Rajan. 2020. On Decomposing a Deep Neural Network into Modules. In *ESEC/FSE'2020: The 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Sacramento, California, United States).

[52] Judea Pearl. 2000. Causality: Models, reasoning and inference cambridge university press. *Cambridge, MA, USA*, 9 (2000), 10–11.

[53] Judea Pearl. 2009. Causal inference in statistics: An overview. *Statistics surveys* 3 (2009), 96–146.

[54] 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.

[55] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[56] Christopher Russell, Matt J Kusner, Joshua R Loftus, and Ricardo Silva. 2017. When worlds collide: integrating different counterfactual assumptions in fairness. *Advances in Neural Information Processing Systems 30. Pre-proceedings* 30 (2017).

[57] Pedro Saleiro, Benedict Kuester, Loren Hinkson, Jesse London, Abby Stevens, Ari Anisfeld, Kit T Rodolfa, and Rayid Ghani. 2018. Aequitas: A bias and fairness audit toolkit. *arXiv preprint arXiv:1811.05577* (2018).

[58] Babak Salimi, Luke Rodriguez, Bill Howe, and Dan Suciu. 2019. Interventional fairness: Causal database repair for algorithmic fairness. In *Proceedings of the 2019 International Conference on Management of Data*. 793–810.

[59] Scikit Learn. 2019. Feature Selection Methods. https://scikit-learn.org/stable/modules/feature_selection.html.

[60] Scikit Learn. 2019. Preprocessing API Documentation. https://scikit-learn.org/stable/modules/classes.html#module-sklearn.preprocessing.

[61] Scikit Learn. 2019. Scikit Learn SimpleImputer. https://scikit-learn.org/0.18/modules/generated/sklearn.preprocessing.Imputer.html.

[62] Scikit-Learn Pipeline. 2020. Scikit-Learn API Documentation. https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html.

[63] 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).

[64] 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.

[65] 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.

[66] 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.

[67] US Equal Employment Opportunity Commission. 1979. Guidelines on Employee Selection Procedures. https://www.eeoc.gov/laws/guidance/questions-and-answers-clarify-and-provide-common-interpretation-uniform-guidelines.

[68] Alper Kursat Uysal and Serkan Gunal. 2014. The impact of preprocessing on text classification. *Information Processing & Management* 50, 1 (2014), 104–112.

[69] Mohammad Wardat, Wei Le, and Hridesh Rajan. 2021. DeepLocalize: Fault Localization for Deep Neural Networks. In *ICSE'21: The 43nd International Conference on Software Engineering* (Virtual Conference).

[70] Ke Yang, Biao Huang, Julia Stoyanovich, and Sebastian Schelter. 2020. Fairness-Aware Instrumentation of Preprocessing Pipelines for Machine Learning. In *Workshop on Human-In-the-Loop Data Analytics (HILDA'20)*.

[71] 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).

[72] Carlos Vladimiro González Zelaya. 2019. Towards Explaining the Effects of Data Preprocessing on Machine Learning. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2086–2090.

[73] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. 2013. Learning fair representations. In *International Conference on Machine Learning*. 325–333.

[74] 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.

[75] Junzhe Zhang and Elias Bareinboim. 2018. Fairness in decision-making—the causal explanation formula. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.