An experimental analysis of multi-step pipelines for fair classifications: More than the sum of their parts? [Extended]

Nico Lässig
University of Stuttgart
Stuttgart, Germany
nico.laessig@ipvs.uni-stuttgart.de

Melanie Herschel
University of Stuttgart
Stuttgart, Germany
melanie.herschel@ipvs.uni-stuttgart.de

ABSTRACT

The problem of biased machine learning predictions has led to many alternative approaches to mitigate the problem. These approaches are typically studied and evaluated by focusing on the input data, the trained model, and the performance of the model predictions. We take a broader perspective by considering approaches in the context of a multi-step pipeline. We study fair classification considering a pipeline that consists of multiple data preparation steps, parameter optimization, and three types of approaches (pre-, in-, and post-processing) designed to reduce bias that may be applied consecutively. This pipeline eventually leads to a trained model to be evaluated in terms of quality (e.g., accuracy) and fairness. We experimentally evaluate the effect different implementations of the pipeline components have on the performance of more than 30 fairness-inducing algorithms (plus combinations thereof). Key findings made possible through this novel pipeline perspective are: (1) Some data preparation steps that are often assumed to have positive (e.g., removal of protected attributes for training, class balancing) or negative effects (e.g., binarization of non-binary protected attributes and groups) on the performance of fairness-inducing methods prove to have little or even contrary effects to the expected ones. (2) When using parameter optimization, there is no significant difference in performance between types of bias mitigation algorithms, which refutes earlier results observed on algorithm performance with default settings. (3) Pipelining multiple fairness-inducing algorithms of different types most frequently degrades the overall performance. Our results show that synergetic effects between pipeline components do not necessarily prevail, demonstrating the need for further research on fair end-to-end data processing. Our extensible experimental framework is publicly available and can springboard such follow-up research.

PVLDB Reference Format:
Nico Lässig and Melanie Herschel. An experimental analysis of multi-step pipelines for fair classifications: More than the sum of their parts? [Extended]. PVLDB, 17(X): XXX-XXX, 2024.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/fwenjd-anon/fair-data-processing-pipeline-exp.

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit https://creativecommons.org/licenses/by-nc-nd/4.0/ to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.
Proceedings of the VLDB Endowment, Vol. 17, No. X ISSN 2150-8097.
doi:XX.XX/XXX.XX

1 INTRODUCTION

Today, many companies and systems rely on machine learning (ML) to predict various events and recommend actions that should be taken. In many scenarios, it is critical for these predictions to be fair and not to exhibit bias towards specific groups of people. Example scenarios are credit scoring [52], hiring systems [55], college admissions [9], or crime recidivism [12]. There are several real-world examples in which discriminative ML models were applied. An illustration of bias surfacing in using ML models is seen in a recruitment tool developed by Amazon, which demonstrated bias against female candidates for technical jobs [23]. Similarly, a study found that Google's ad-serving algorithm also showed discriminative behavior towards females, as the frequency of high-paying job ads was much less than for males [15]. Another example is the PredPol software employed for crime forecasting in certain U.S. regions, which has been linked to a rise in instances of racial profiling [57].

The above examples illustrate the large variety of data-driven applications that require the incorporation of fairness such as not to discriminate people from a certain protected group (e.g., based on a specific gender, race, or a combination of both). A first step, already incorporated in some laws, is to entirely disregard protected attributes, such as gender or race, during data processing, e.g., for training ML models. In addition, research has explored algorithmic solutions for fair clustering [20], fair ranking [79, 80], fair natural language processing [10, 33], and fair regression [16], among others. Yet, most research has focused on fair classifications [16, 42, 61, 66], with a large majority concentrating on binary classification. A binary classification classifies tuples into two possible classes. One class has the favorable outcome (e.g., a person gets a loan), while the other is the unfavorable class (e.g., a person is denied a loan). *This paper focuses on the experimental evaluation of fairness-inducing algorithms for binary classification, as it has high practical relevance and, given the plethora of available solutions, insights into when or how to use these algorithms are most valuable.*

Multiple algorithmic approaches have been proposed to mitigate the issue of bias in ML predictions, including the aforementioned binary classifications. These algorithms broadly fall into three categories: *pre-processing*, *in-processing*, and *post-processing* [16, 39, 42, 46, 61, 66], depending on which stage of the training process they apply. Most fairness-inducing approaches target binary classifications, sometimes even confined to situations with a single binary protected attribute, thereby requiring some "binarization" before being applicable to a dataset with multiple or non-binary protected attributes. Binarization may lead to discrimination against individuals with multiple protected attributes [31, 59]. Many of the proposed algorithms also rely on parameters that may have a varying impact on the overall performance. Similarly, removing protected attributes as part of data preparation may impact the overall performance [14, 38]. In practice, further "standard" data preparation steps such as sampling or dimensionality reduction precede ML model training, which may also affect the model performance. *Overall, this paper evaluates fair classification algorithms in the context of a processing pipeline that comprises several data preparation steps, parameter optimization for fair classification algorithms, and the three possible stages of pre-, in-, and post-processing, coupled with model training.* This goes beyond earlier work on comparative evaluation, focusing on fair classification algorithms with their input and output "in isolation". While this paper focuses on a few variations of a baseline data processing pipeline, the general methodology and publicly available code base [1] can serve as a blueprint for further valuable experimental studies in the future.

The variety of fairness-inducing algorithms is accompanied by multiple fairness notions and metrics [46, 61]. While some algorithms are specifically designed to support one (type of) fairness definition or metric, others have been defined to accommodate multiple ones. As demonstrated in previous studies [21, 46], no single algorithm may optimally cater to all definitions simultaneously (and thus make all other algorithms superfluous). *In our evaluation, we study to what extent different component implementations of our pipeline impact the "gap" between different algorithms on different fairness definitions and metrics.*

Contributions. The overarching contribution of our experimental analysis is to evaluate algorithms for fair binary classification when integrated into a processing pipeline, thereby looking at a larger context than previous studies. Fig. 1 outlines the general pipeline we consider and summarizes which "implementations" of each component we focus on. It also highlights our contributions:

(1) We evaluate 36 fairness-inducing algorithms over six real-world datasets and two types of synthetic datasets. To the best of our knowledge, this is the most extensive evaluation of such algorithms compared to previous experimental studies and the only one taking a broader pipeline perspective. Hort et al. [43] evaluate the algorithms implemented via AIF360 [5], a well-known and widely used state-of-the-art framework for fair classifications. Friedler et al. [32] compare four bias mitigation algorithms. Islam et al. [46] provide an empirical evaluation of 9 algorithms of the AIF360 framework and four additional fair classification strategies. One interesting result based on this broad perspective (broad in both the number of approaches and pipeline context) is that *opposed to [46], we observe* no *significant difference between different types of fairness-inducing algorithms for properly configured pipelines (i.e., performing parameter optimization).*

(2) We discuss the performance of fairness-inducing algorithms in terms of quality and fairness (for several fairness definitions) in the context of a processing pipeline [2]. We consider several data preparation steps, the effect of parameter optimization for fairness-inducing approaches (using grid-search), and pre-, in-, and post-processing approaches (separate or in combination). Our goal is to investigate the possible impact of the individual steps in the pipeline and their possible combinations on the final classification performance to offer some guidance on how to configure the pipeline for different application requirements. The configuration recommendations resulting from our evaluation include the following: *(i) While it is commonly recommended to discard protected attributes from training, we observe that in most experiments, algorithms fare better when these attributes are* not*removed. (ii) As binarization may lead to discrimination against individuals with multiple protected attributes, a frequent criticism of existing approaches is that they require binarization as they only support binary protected attributes or groups. In the majority of our experiments, binarization does* not *have a negative effect or even improves performance, even when applied as a data preparation step for algorithms with native support of non-binary protected attributes. (iii) Performing data balancing is typically not necessary as the majority of algorithms perform best when this pipeline component is switched off. (iv) Both feature selection and dimensionality reduction, both common data preparation tasks in ML pipelines, have a noticeable and varying effect on fair classification results, preventing a general recommendation but showcasing that it is important to consider such data preparation steps as an integral part of the problem when studying fair classification. (v) Parameter optimization (that, not surprisingly, typically improves performance compared to default settings) simplifies the choice of a fairness-inducing algorithm when targeting group fairness, as it makes algorithms not originally designed for a specific metric competitive with algorithms specialized to a metric. (vi) While it may be tempting to combine pre-, in-, and post-processing techniques to tackle bias from all its angles, we determine that combining multiple approaches often has a negative effect.*

(3) In addition to the fundamentally new insights into the behavior of a large variety of fairness-inducing algorithms for binary classifications gained through our extensive evaluation, this work presents a new way to evaluate such algorithms in a systematic and holistic way. Our publicly available implementation can be extended to further pipeline steps and benchmark settings, thereby serving as an ideal springboard for follow-up research. The need for such research is apparent from our insights that revealed that synergetic effects between pipeline components do not necessarily prevail, and effects of different pipeline steps may be counter-intuitive, less pronounced than expected, or in need to be considered as an integral part of the fairness-optimizing system.

Structure. In Sec. 2, we summarize the pipeline configurations that our experimental evaluation considers. Sec. 3 and Sec. 4 provide details on fair classification algorithms and the fairness metrics we focus on, respectively. Sec. 5 presents our experimental setup and experimental analysis. We summarize our insights and discuss possible implications in Sec. 6.

2 DATA PROCESSING PIPELINE OVERVIEW

Fig. 1 summarizes the processing pipeline framework for fair binary classifications that we consider and the component implementations our experimental study covers.

2.1 Pipeline framework components

The pipeline input is a *dataset*, split into training and validation data. The pipeline also relies on the specification of a *fairness metric*.

The *data preparation* component allows the integration of data manipulations and transformations common when preparing data

[1]https://github.com/fwenjd-anon/fair-data-processing-pipeline-exp
[2]Our study on scalability is relegated to an extended version due to space constraints, available in our repository

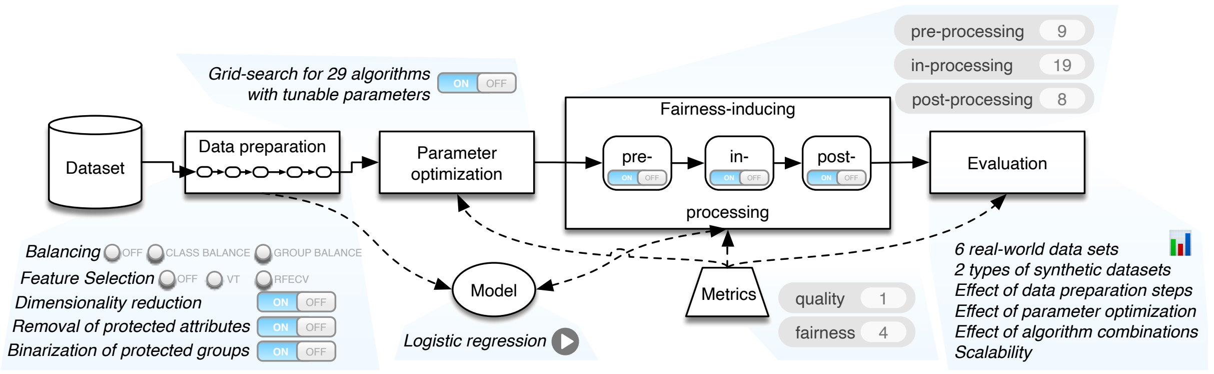
pre-processing 9
in-processing 19
post-processing 8
Grid-search for 29 algorithms with tunable parameters
ON OFF
Fairness-inducing
Dataset
Data preparation
Parameter optimization
pre-
in-
post-
processing
Evaluation
Balancing OFF CLASS BALANCE GROUP BALANCE
Feature Selection OFF VT RFECV
Dimensionality reduction
Removal of protected attributes
Binarization of protected groups
Model
Logistic regression
Metrics
quality 1
fairness 4
6 real-world data sets
2 types of synthetic datasets
Effect of data preparation steps
Effect of parameter optimization
Effect of algorithm combinations
Scalability

Figure 1: Processing pipeline overview with implementation options for different components and evaluation contributions

for ML training, e.g., projecting out certain attributes, discretizing values, performing dimensionality reduction, etc. These data preparation steps are generally orthogonal to data transformations specifically targeted towards fair classification (which are the focus of dedicated pre-processing algorithms). However, projecting out protected attributes usually does impact fairness, so clearly, there is some overlap between data preparation for ML in general and data preparation for fair ML.

The *parameter optimization* component specifically targets the optimal setting of fairness techniques, depending on the (prepared) data and the considered fairness metric. In general, there are several hyperparameter optimization techniques [74], like grid search, random search [6], or Bayesian optimization [71].

The *fairness-inducing* component has three sub-components corresponding to different types of algorithms (pre-, in-, and post-processing) for fair classification. In principle, these can be applied in sequence. Most existing algorithms target specific fairness metrics but are amenable to supporting a larger variety of metrics. Sec. 3 covers the implemented algorithms in more detail. Some approaches require an additional *model*. For instance, the pre-processing algorithms mutate or return sample weights for the training data. This requires an additional classifier that uses the adapted training data as input. Likewise, post-processing algorithms either modify classifiers directly or the output of classifier results.

The *evaluation* component assesses the trained model in terms of classification quality and fairness by applying it to the test data. Evaluation *metrics* (for fairness and quality) are, in principle, independent from those specified to be the focus for data processing for model training (usually accuracy and a specific fairness metric).

2.2 Implemented pipeline configurations

Our pipeline implementation focuses on five data preparation steps. *Sampling*, *feature selection*, and *dimensionality reduction* represent data preparation steps that are commonly used in feature engineering. In selecting implementation variants for these steps, we reviewed what is common practice in publicly availalbe ML pipelines. For sampling, we use a combination of *SMOTE* oversampling and *Tomek links* undersampling [3]. For balancing, we consider two strategies. The first strategy balances the label classes. We also adapted this approach to include the protected attributes, i.e., it will balance all group combinations, including protected attributes and labels. We distinguish these approaches as *class balancing* and *group balancing*, respectively. For the feature selection component, we offer two options. The *variance threshold* (short: VT) approach [54] as an unsupervised feature selection technique and the supervised *recursive feature elimination with cross-validation* (short: RFECV) [37] approach. Finally, for dimensionality reduction, we use the *principal component analysis* (short: PCA) approach with the automatic choice of dimensionality by Minka [62]. We also consider two data preparation steps that are frequent when focusing on the fairness of trained models. The first such step considers the potential *removal of protected attributes*, as these can either be kept or projected out for the training step. Note they are still kept as metadata, as algorithms typically require them for their fairness-inducing mechanisms. We name the second step *binarization*: if the input data contains multiple protected attributes or a non-binary protected attribute, this step transforms the data to include only a singly binary protected attribute, an expected input to many fair classification algorithms. As an example, when we have two protected attributes, e.g., {race, sex}, then we form one single privileged group (e.g., {white, male}), while all other combinations form the discriminated group.

In our implementation, we can either activate or deactivate parameter optimization. When activated, we opt for *grid search* to find an optimal parameter setting. It builds a grid of all hyperparameters to be tuned and the designated values that we want to observe. Then, the algorithm to be optimized is run for all different combinations of parameter values, returning the parameter values that yield the optimal results. To evaluate the results, a loss function $\hat{L}$ is used that determines the quality of each parameter set. As the goal is to achieve fairness while retaining high accuracy, we choose a loss function that equally weighs the error rate and the considered fairness metric. When deactivating parameter optimization, the algorithms run using their default parameter settings. We determine these as follows: we either obtain them from (1) the official implementation or (2) the best settings of the experiments in the original paper. If the parameter settings cannot be obtained from (1), then we obtain them from (2). Note that we choose grid search over more efficient parameter optimization techniques as it offers a systematic way to reach our main goal of assessing algorithm performance through optimized parameter settings compared to the default ones.

The sub-components of the fairness-inducing component can be implemented by choosing among 9 pre-processing, 19 in-processing, and 8 post-processing algorithms (introduced in Sec. 3), either separately or in combination. Note, however, that not every algorithm from one type is compatible with every algorithm of another type. For instance, post-processing algorithms may adjust a specific type of classifier not supported by in-processing techniques, or they perform adjustments to the input data that would require another "loop" of in-processing. When an algorithm requires an additional model (as explained in Sec. 2.1), we train a *Logistic Regression* classifier [34]. We justify this choice as follows. In the experiments by Islam et. al [46], Logistic Regression consistently finished among the top. Furthermore, Logistic Regression is a common choice for experiments of bias mitigation techniques [42] used to verify the models in the original research papers or in the documentation of their implementation.

For each pipeline configuration, our implementation can target one of four fairness metrics (covered in Sec. 4). When additionally considering all the datasets underlying our evaluation, we end up with over 10.000 pipeline configurations that we ran and analyzed.

3 FAIR CLASSIFICATION ALGORITHMS

Numerous fair classification algorithms have been proposed over the last few years. These approaches are typically categorized based on the processing stage to which they apply [16, 42, 61, 66]. (1) *Pre-processing* techniques modify the training data or assign different weights to the samples. (2) *In-processing* approaches induce fairness during the learning/training process. (3) *Post-processing* strategies adjust the input, classifiers, or output the classifiers produce.

Tab. 1 summarizes the approaches we consider in our evaluation, which we briefly review in Sec. 3.1. Sec. 3.2 describes features we surveyed about these approaches pertinent to our evaluation.

3.1 Algorithm overview

This section provides an overview of pre-, in-, and post-processing algorithms. In discussing algorithms along these three categories, we follow further subcategories defined in a recent survey [42], ensuring that we cover each subcategory with at least one competitive algorithm. Note that our framework and evaluation focus on approaches functioning without additional domain knowledge of the underlying data. Consequently, we neither cover causal fairness metrics nor include causally fair classification approaches in our evaluation [46, 64], as these compute fairness based on causal relationships between the protected attributes and other features.

3.1.1 Pre-processing. We start by introducing the nine pre-processing algorithms we discuss in our evaluation.

Relabelling and perturbation are methods that adjust the original labels or slightly modify the input data to remove or reduce bias. *DisparateImpactRemover* [30] alters the labels of the input data while preserving the rankings within a group. Thus, this algorithm can also be applied to fair ranking problems. *LTDD* [56] applies perturbation by building a linear regression model to see the influence of a protected attribute on the non-protected attributes. If a significant bias is determined, this data is perturbated. This strategy is not only applied to the training data but also to the test data later used. Additionally, after the perturbation of the attributes, the protected attribute is removed from the data. In the *FairSSL* [19] framework, biased instances are unlabeled and relabelled based on semi-supervised learning techniques. Tab. 1 refers to four variants of FairSSL, depending on the semi-supervised technique they use: *Self Training (= ST)* [75], *Label Propagation (= LP)* [85], *Label Spreading (= LS)* [84], or *Co-Training (= CT)* [11].

Sampling techniques alter the distribution of the training data. They can involve oversampling underrepresented groups or undersampling overrepresented ones to achieve more balanced datasets. *Fair-SMOTE* [18] starts by oversampling to balance all subgroups of the data. A combination of protected attribute values and label values defines one subgroup. Then, biased training data instances are removed from the dataset.

Reweighing methods retain the instances of the original dataset but assign different weights to them based on their labels and protected characteristics. The *Reweighing* approach [48] assigns higher weights to tuples from the protected group assigned to the positive class while assigning lower weights to tuples that are from the protected group but are assigned to the negative class. It proceeds analogously with the weights of the unprotected groups.

Representation learning aims to transform the input data into a representation that preserves the essential information for the predictions but removes the information related to the bias. This can be achieved by modifying the objective function to encourage the model to learn fair representations. *LFR* [81] uses clustering and the probability distribution to which cluster each sample belongs to have a representative dataset.

3.1.2 In-processing. We now summarize the 19 in-processing algorithms underlying our evaluation along four subcategories.

Regularization techniques modify the loss function by adding a fairness term to induce fairness during the classifiers' learning phase. *PrejudiceRemover* [50] uses mutual information metrics and hereby detects and removes indirect discrimination. Mary et al. [60] propose *HGR*, which adds a regularization term to the existing error minimization term. Here, the goal is to minimize the dependency between the protected attribute and the prediction. The approach can be applied to continuous variables. *FairDummies* [69], constructs a fair dummy protected attribute. The optimization function then adds a term that computes the discrepancy between the probability distribution using the real protected attribute and the dummy attribute. *HSICLinearRegression* [65] proposes a penalty to achieve high independence between the protected attributes and the predictions. To this end, the Hilbert-Schmidt independence criterion (HSIC) is used, and a linear regression model is trained. Berk et al. [7] apply two penalties for individual fairness and group fairness, respectively. This approach can be applied to linear regression or logistic regression. The goal behind *SquaredDifferenceFairLogistic* [4] is to achieve similar false positive and negative rates across the protected groups. For this, it adds penalizing terms and trains a logistic regression model following the adapted optimization problem. *FairGeneralizedLinearModel* [25] leverages a regularizer that has the advantage of surpassing the limits of binary classifications, as it can also be used for multi-class classification problems as well as regression problems. It uses the Newton-Raphson method [76] to optimize the objective function. *GradualCompatibility* [41] optimizes cross-entropy, l_2 regularization, and fairness regularization.

	Ref.	Model	Author	Metric	Tuned Parameters	Ignore Protected	Multiple
	[48]	Reweighing	Kamiran & Calders	dp	-	implementation-based	no
	[81]	LFR	Zemel et al.	dp, consistency	3 (2x continuous, 1x discrete)	implementation-based	no
	[30]	DisparateImpactRemover	Feldman et al.	dp	1 (continuous)	built-in	no
	[18]	Fair-SMOTE	Chakraborty et al.	dp, eod, eop	-	implementation-based	yes
pre	[19]	FairSSL-CT	Chakraborty et al.	dp, eod, eop	1 (binary)	implementation-based	yes
	[19]	FairSSL-LP	Chakraborty et al.	dp, eod, eop	1 (binary)	implementation-based	yes
	[19]	FairSSL-LS	Chakraborty et al.	dp, eod, eop	1 (binary)	implementation-based	yes
	[19]	FairSSL-ST	Chakraborty et al.	dp, eod, eop	1 (binary)	implementation-based	yes
	[56]	LTDD	Li et al.	dp, eod, eop	-	built-in	no
	[50]	PrejudiceRemover	Kamishima et al.	dp	1 (continuous)	-	no
	[4]	SquaredDifferenceFairLogistic	Bechavod et al.	treq	1 (continuous)	built-in	no
	[78]	FairnessConstraintModel	Zafar et al.	dp	1 (continuous)	built-in	no
	[77]	DisparateMistreatmentModel	Zafar et al.	treq	1 (continuous)	built-in	no
	[7]	ConvexFrameworkModel	Berk et al.	dp, other indiv.	2 (1x continuous, 1x discrete)	built-in	no
	[65]	HSICLinearRegression	Perez-Suay et al.	other	1 (continuous)	built-in	no
	[51]	GerryFairClassifier	Kearns et al.	dp, other	1 (continuous)	-	yes
	[83]	AdversarialDebiasing	Zhang et al.	dp, eod, eop	1 (continuous)	built-in	no
	[1]	ExponentiatedGradientReduction	Agarwal et al.	dp, eod	3 (2x continuous, 1x binary)	parameter-based	yes
in	[1]	GridSearchReduction	Agarwal et al.	dp, eod	2 (1x continuous, 1x binary)	parameter-based	yes
	[17]	MetaFairClassifier	Celis et al.	dp, other	1 (continuous)	-	no
	[44, 45]	AdaFair	Iosifidis et al.	eod, eop, dp, treq	2 (1x continuous, 1x discrete)	-	no
	[36]	FAGTB	Grari et al.	dp, eod	3 (2x continuous, 1x discrete)	implementation-based	no
	[60]	HGR	Mary et al.	eod	4 (2x continuous, 2x discrete)	built-in	no
	[69]	FairDummies	Romano et al.	eod	5 (3x continuous, 2x discrete)	built-in	no
	[63]	GeneralFairERM	Oneto et al.	eod	1 (continuous)	built-in	no
	[2]	MultiAdversarialDebiasing	Ball-Burack et al.	dp, eod	1 (continuous)	built-in	no
	[25]	FairGeneralizedLinearModel	Do et al.	eod, other	2 (1x continuous, 1x discrete)	built-in	no
	[41]	GradualCompatability	Hertweck et al.	other	3 (2x continuous, 1x discrete)	built-in	no
	[49]	RejectOptionClassification	Kamiran et al.	dp, eod, eop	-	implementation-based	no
	[40]	EqOddsPostprocesssing	Hardt et al.	eod	-	implementation-based	no
	[67]	CalibratedEqOddsPostprocessing	Pleiss et al.	treq	-	implementation-based	no
	[47]	JiangNachum	Jiang & Nachum	dp, eod, eop	2 (1x continuous, 1x discrete)	implementation-based	yes
post	[70]	DPAbstention	Schreuder & Chzhen	dp	1 (continuous)	-	yes
	[35]	FaX	Grabowicz et al.	consistency, other	-	built-in	no
	[82]	FairBayesDPP	Zeng et al.	other	1 (continuous)	-	no
	[72]	GetFair	Sikdar et al.	dp, eod, eop, treq	4 (1x continuous, 3x discrete)	built-in	no

Table 1: List of fair binary classification approaches. Metric abbreviations: dp = demographic parity; eod = equalized odds; eop = equal opportunity; treq = treatment equality. The parameter optimization component potentially affects all algorithms with tuned parameters. Highlighted rows indicate algorithms potentially influenced by fairness-specific data preparation steps. All algorithms may be affected by further data preparation steps.

Constraint optimization techniques directly incorporate fairness constraints into the optimization problem the learning algorithm is trying to solve. The learning algorithm would find the best model that meets this constraint. Agarwal et al. [1] define two constraint-based approaches for fair classifications. In *GridSearchReduction*, hyperparameter tuning is performed via grid search over a grid of values for the fairness and accuracy tradeoff. For each point in the grid, a constrained optimization problem is solved to find the best model that meets both fairness and accuracy objectives. The *ExponentiatedGradientReduction* algorithm reduces a fair classification problem to a sequence of cost-sensitive classification problems, in which the costs denote the losses for the predictions. Celis et al. [17] provide a multi-objective optimization approach via the *MetaFairClassifier*. Several fairness notions are unified and converted to a linear form. Zafar et al. [77, 78] propose two methods (*DisparateMistreatmentModel* [77] and *FairnessConstraintModel* [78]) which solve a constraint optimization problem, where either the loss function of the training set is minimized while satisfying fairness constraints or the fairness is minimized while satisfying accuracy constraints. Here, the distance of the instance t from the decision boundary is used as a proxy of Z_t. *GeneralFairERM* [63] adds a fairness constraint that can be reduced to the equalized odds metric to the optimization problem. The approach relies on a threshold that determines the amount of tolerated bias. The approach applies to classification and regression problems with categorical or continuous protected features. The *GerryFairClassifier* [51] creates subgroups of the dataset based on the combination of the protected attribute values. Then, fairness constraints hold on these subgroups and are audited.

Adversarial learning involves training classification models and their adversaries simultaneously. While the classification model is trained on accuracy, an adversary is used to determine if the classifier is fair. The adversary is then used to tune the model. In *Adversarial Debiasing* [83], the first step is to train a model. The next step is to predict the protected attributes based on the predicted label Z and, depending on the chosen fairness notion, the true label Y. At each iteration of the training step, the ability to predict the protected attribute, hence the adversary, is reduced. *FAGTB* (short for *Fair Adversarial Gradient Tree Boosting*) [36] applies adversarial learning during the gradient tree boosting phase. *GerryFairClassifier* [51] audits the data and adds constraints regarding the audited data. *MultiAdversarialDebiasing* [2] adapts *AdversarialDebiasing* to multi-class classification problems and combines it with the pre-processing preferential sampling technique.

Compositional methods train several models, forming a model ensemble used to classify instances. The single models alone might not be suited to fairly and accurately classify instances on the whole data, but as an ensemble, fairness is achieved.*AdaFair* [44] is inspired by the AdaBoost ensemble learning technique. The prediction error rate and fairness constraints influence the reweighing of the instances in each iteration. In [45], Iosifidis et al. propose an extension that adds more fairness notions. *FAGTB* [36] uses gradient tree boosting as a basis and adapts it via adversarial learning.

	Metric	Definition	Description
Correctness	Error rate	$\frac{FP+FN}{\lvert D\rvert}$	Fraction of falsely classified tuples (complement of accuracy)
Group fairness	Demographic parity [30]	$\frac{1}{\lvert G\rvert}\sum\limits_{i\in G}\left\lvert P(Z{=}1\lvert G_i) - P(Z{=}1)\right\rvert$	Protected and unprotected groups have an equal probability of a positive outcome.
	Equalized odds [40]	$\frac{1}{\lvert G\rvert}\sum\limits_{i\in G}\left(\frac{1}{2}\left\lvert FPR_{G_i} - FPR_{total}\right\rvert + \frac{1}{2}\left\lvert TPR_{G_i} - TPR_{total}\right\rvert\right)$	The probability of a positive outcome for tuples with a real positive label should be equal among all groups, and the probability of a positive outcome for tuples with a real negative label.
	Treatment equality [8]	$\frac{1}{\lvert G\rvert}\sum\limits_{i\in G}\left\lvert \frac{FP_{G_i}}{FP_{G_j}+FN_{G_j}} - \frac{FP_{total}}{FP_{total}+FN_{total}}\right\rvert$	Ratio of false positives (FP) to false negatives (FN) is equal among all groups.
Individual fairness	Consistency [81]	$\frac{1}{\lvert D\rvert}\sum\limits_{i\in D}\left\lvert Z_i - \frac{1}{k}\sum_{j\in\text{kNN}(X_i)} Z_j\right\rvert$	Similar persons, defined via k-nearest neighbor algorithm, have a similar outcome.

Table 2: Metrics

Notation	Description
D	The whole dataset
G	A set of protected groups
X	Set of non-protected attributes
Y	Attribute denoting the ground-truth class label
Z	Attribute denoting the predicted class label
G_i	The protected group G for tuple $i \in D$
X_i	The non-protected attributes X for tuple $i \in D$
Y_i	The ground truth Y for tuple $i \in D$
Z_i	The predicted class Z for tuple $i \in D$

Table 3: Notations

	$Y = 0$	$Y = 1$
$Z = 0$	True Negative (TN) TNR: $Pr(Z = 0 \lvert Y = 0)$	False Negative (FN) FNR: $Pr(Z = 0 \lvert Y = 1)$
$Z = 1$	False Positive (FP) FPR: $Pr(Z = 1 \lvert Y = 0)$	True Positive (TP) TPR: $Pr(Z = 1 \lvert Y = 1)$

Table 4: Confusion matrix

3.1.3 Post-processing. This section summarizes the eight post-processing algorithms listed in Tab. 1 along three subcategories.

Input adjustments modify the input data. The difference from the pre-processing approaches is that the previous sample run's information is used to adapt the input data targeted for a specific classifier. One example is the approach by Jiang and Nachum [47], which uses a reweighing strategy iteratively and adapts the weights after the classifier has predicted the class labels on the dataset with the weights of the previous iteration.

Output adjustments alter the output of the classifiers. The main approach for binary classification tasks is to determine a threshold for each protected and unprotected group. In the *RejectOptionClassification* [49] approach, instances are assigned to the positive class if their prediction probability reaches these thresholds. Using thresholds requires a probability distribution output of the trained classifiers. *FairBayesDPP* [82] is another adaptive thresholding approach built on the results of a Bayes-optimal classifier.

Classifier adjustment methods modify the predictive model after its training to correct for any bias. This can be achieved, for instance, by calibrating the classifier's predictions or learning a completely new transformation of the outputs that reduces the bias. *EqOddsPostprocessing* [40] modifies a Bayes regressor such that the "equalized odds" fairness constraint is not violated. For the classification, a threshold is then chosen for all protected groups. In *CalibratedEqOddsPostprocessing* [67], calibration constraints are added along the fairness constraint. Furthermore, here, the classifiers are trained separately on the groups. *DPAbstention* [70] takes a trained classifier as input along an additional set of unlabeled samples. These samples are then used to adapt the classifier to satisfy the "demographic parity" metric. *FaX* [35] reduces the discrimination by minimizing SHAP (Shapley additive explanations) [58] and the MDE (marginal direct effect). *GetFair* [72] takes any parameterized classifier already trained on accuracy, followed by a policy gradient parameter tuning process. The reward function of this tuning procedure can take several fairness metrics as input.

3.2 Characteristics

Tab. 1 summarizes several algorithm characteristics relevant for our evaluation. The characteristics in the table refer to the publicly available implementation of these approaches.

Metric. Indicates for which fairness metric algorithms are designed or optimized. Most algorithms support demographic parity, equalized odds, and/or equal opportunity. Only a few approaches consider treatment equality or individual fairness (in particular, using the consistency metric). Some support other metrics, e.g., equalized expected log-likelihoods [25] or predictive parity [82].

Tuned parameters. While some of the pre-processing and post-processing approaches do not need any parameters (labeled –), all in-processing algorithms have at least one parameter that is tuned (see number in the column). To keep aligned with the original work, we tune parameters also considered in the experiment section of the original research papers. We further tune some available parameters commonly tuned in related approaches, i.e., the continuous fairness weight λ, a threshold for allowed fairness constraint violation ϵ, and the learning rate η. Some algorithms have additional parameters for which we use the default values.

Ignore protected. One pipeline component can be switched on to ignore protected attributes. Tab. 1 identifies the algorithms that support this component. Many algorithms have this component built-in, meaning that the protected attributes are automatically ignored during the training phase and only kept as metadata to optimize the models. Thus, this component does not affect these algorithms. *GridSearchReduction* [1] and *ExponentiatedGradientReduction* [1] provide this as a parameter option, thereby supporting our component's functionality "out of the box". For the algorithms labeled as implementation-based, we extended their implementations to support this option. This was not possible for all.

Multiple. This column relates to binarization, also part of data preparation in our pipeline (see Sec. 2). Only a few approaches natively handle multiple protected attributes (labeled as yes). Some papers claim to allow or describe how to possibly extend to multiple protected attributes, but this remains theoretical by neither being implemented nor evaluated.

4 EVALUATION METRICS

Tab. 2 defines the metrics we use in our evaluation, relying on the information given in Tab. 3 and 4. We focus on metrics frequently used and not related to one another. Their range is within $[0, 1]$. The lower the value, the better the performance. For more details on further metrics, see [16, 46, 66, 86] (fairness) and [73] (correctness).

dataset	sensitive attr.	# of samples	# of features	Pr(Y=1|G=1)	Pr(Y=1|G=0)	Pr(G=1)
ACS2017 [13]	race	$72k$	23	49.6%	28.2%	58.8%
Adult Data [26]	race	$46k$	21	26.3%	16%	85.7%
Adult Data [26]	sex	$46k$	21	31.3%	11.4%	67.6%
Adult Data [26]	race, sex	$46k$	21	32.4%	12.3%, 22.6%, 7.6%	59.6%
Communities [27]	race	$2k$	91	19.4%	62.6%	51.4%
COMPAS [68]	race	$6.1k$	7	38.5%	50.2%	40.1%
COMPAS [68]	race, sex	$6.1k$	7	39.9%	53%, 33.6%, 36.5%	31.4%
Credit Card [28]	sex	$30k$	23	20.8%	24.2%	60.4%
German [29]	sex	$1k$	21	75.5%	78.8%	31%
Implicit	sensitive	$15k$	8	65%	35%	50%
Social	sensitive	$15k$	8	65%	35%	50%

Table 5: Metadata about datasets, including probabilities wrt group association.

To measure the quality or correctness of a classifier, we use the error rate (complement of accuracy). To assess fairness, we use metrics for both group fairness and individual fairness.

For *group fairness* [16, 61], protected groups should have equal probabilities of an outcome. Thus, e.g., the probability of having a favorable outcome in a binary classification scenario should be independent of the protected attributes. The broad concept of global fairness incorporates various specific definitions, such as *demographic parity* [30], *equalized odds* [40], or *treatment equality* [8].

Individual fairness aims for similar individuals being treated similarly [61]. Clearly, this needs further specification of what similar means. One prominent individual fairness metric is *consistency* [81]. It calculates the average difference of the prediction of a person, compared to the average prediction of its k-nearest neighbors [22].

The "perfect" algorithm would obtain low scores for all metrics simultaneously. However, it has been shown that there is a trade-off between correctness and fairness, which should be kept small. Defining a good balance between different fairness metrics is more challenging. For instance, laws may enforce one fairness definition, but we believe even in these cases, other biases should be kept low.

5 EVALUATION

5.1 Setup

5.1.1 Datasets. We evaluate the algorithms both on real-world and synthetic datasets. As real-world data, we use several datasets commonly used in experiments of fairness-aware machine learning approaches [53]. The main features of these benchmark datasets are summarized in Tab. 5. Note that these statistics describe the datasets after some pre-processing that was necessary to map data to numerical values for selected algorithms. We further evaluate the algorithms on two synthetic datasets. Each exhibits one of two biases: *Social* bias (aka direct bias) is the bias resulting solely from the sensitive attribute. For *implicit* bias, the sensitive attribute itself has no direct influence on the overall prediction but correlates with several of the other features that do.

Since some algorithms do not apply to non-binary sensitive groups, we generate binary sensitive groups for the experiments, except for the binarization experiment in Sec.5.3. All datasets are randomly split as follows: 70% for training and 30% for validation.

5.1.2 Algorithms. We compare the performance of the algorithms introduced in Section 3. We use the implementations provided by the AIF360 framework [5] for the *Reweighing* [48], *LFR* [81], *DisparateImpactRemover* [30], *PrejudiceRemover* [50], *GerryFairClassifier* [51], *AdversarialDebiasing* [83], *GridSearchReduction* [1], *ExponentiatedGradientReduction* [1], *MetaFairClassifier* [17], *RejectOptionClassification* [49] *EqOddsPostprocessing* [40], and *CalibratedEqOddsPostprocessing* [67] algorithms. Similarly, for the *FairGeneralizedLinearModel* [25], *FairnessConstraintModel* [78], *DisparateMistreatmentModel* [77], *ConvexFrameworkModel* [7], *HSICLinearRegression* [65], *SquaredDifferenceFairLogistic* [4] and *GeneralFairERM* [63] algorithms, we use the implementation provided by the authors of the *FairGeneralizedLinearModel* paper [24]. For the other algorithms we use the implementations of their respective GitHub repositories linked in the original research papers. Some approaches had to be slightly adapted in order to work within our evaluation framework, although we did not tweak the algorithmic approaches themselves.

Note that due to some memory usage issues, some algorithms did not run on large datasets. In our repository, we show an extended version of the experiments, including the scalability.

5.1.3 Evaluation goals. In Sec. 3, we compared algorithms based on their underlying approach and key properties. Our experimental evaluation focuses on the following research questions that all only arise when considering multi-step pipelines and hence were not in the scope of previous studies:

- (RQ1) How beneficial is hyperparameter tuning in improving the performance of individual fairness-inducing algorithms, and how does this affect the relative performance of fairness-inducing algorithms compared to the typically manually curated default settings considered in previous experimental studies?
- (RQ2) What are the effects of data preparation steps, i.e., balancing, feature selection, dimensionality reduction, removal of protected attributes, and binarization, both in isolation and in combination?
- (RQ3) Does the combination of several bias mitigation algorithms, i.e., a pre-processing algorithm to adapt the dataset combined with an in-processing/post-processing algorithm technique, further improve the results?

We focus on these questions to gain new insights into the possible impact of the individual steps in the pipeline and their combinations on the final classification performance to offer some guidance on how to configure the pipeline for different application requirements. Based on our extensive evaluation and **specific insights**, we can formulate several general $\boxed{\textit{configuration guidelines}}$.

5.2 Effect of hyperparameter tuning (RQ1)

Parameter tuning is a widely used component of ML systems as it allows to improve the quality of models. Focusing on fairness, it has been applied in experiments on very few algorithms, showcasing the gap between good and bad configurations ([32, 43] each considers two algorithms). We integrate parameter optimization as an integral part of a fair classification pipeline and thoroughly evaluate both the scale of individual algorithm performance shift when applying it to fair classification approaches compared to their out-of-the-box default configurations and the impact this has on the comparative evaluation of different algorithms and/or metrics.

5.2.1 Scale of performance shift. To study the scale of the effect of parameter tuning on fairness-inducing algorithms, we compare the

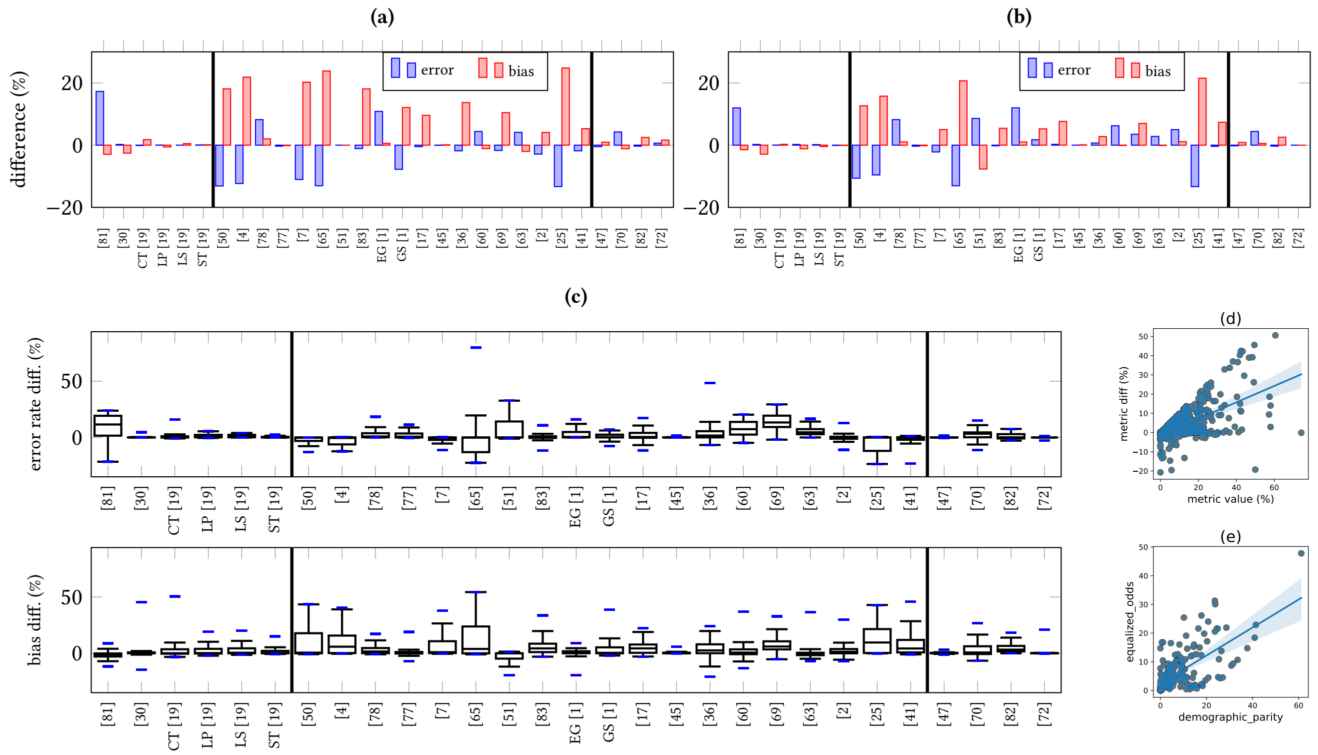

Figure 2: Improvements achieved via hyperparameter tuning compared to the default settings of the classifiers on the COMPAS dataset considering (a) demographic parity; (b) equalized odds. (c) shows the summary of change in error rate and bias metric across all metrics and datasets per algorithm. (d) depicts the correlation between bias in non-optimized algorithm and change in metric through optimization. (e) shows the correlation between demographic parity and equalized odds.

performance of the algorithms with one or more tuneable parameters (see Table 1) in their default mode (cf. Section 2) with their performance when applying automated parameter tuning on the tuneable parameters. The rest of the pipeline components are set as follows for this series of experiments: We mute the data preparation component (all steps set to *off*).This entails that we run this experiment on the dataset versions with a single binary protected attribute. Fairness-inducing algorithms are considered one at a time, i.e., no combinations are performed.

To compare the change in performance between running the pipeline with or without parameter optimization, we compute the difference in scores between the score without optimization and with optimization. A positive value indicates an improvement in performance (i.e., error rate or bias has been reduced), while a negative value signifies that parameter optimization has led to performance degradation. Figures 2(a) – (b) show detailed results when using two different fairness metrics (demographic parity and equalized odds) on the COMPAS dataset. Figure 2(c) summarizes the results across all experiments on the datasets per fairness-boosting algorithm. The boxes depict the first quartile, median, and third quartile, while the error bars represent the minimum and maximum values using the $1.5 \cdot$ interquartile range rule. The blue lines include the outliers and represent the real maximum and minimum values across all experiments. Figure 2(d) relates the score difference back to the absolute score before optimization. Finally, Figure 2(e) shows the correlation between demographic parity and equalized odds.

From this set of experiments, we make the following observations. First, **in-processing algorithms benefit more from parameter optimization than pre- or post-processing solutions**. We show exemplarily in the detailed results in Figures 2(a) – (b) that most changes in both error rate and bias remain low for pre- and post-processing techniques, except *LFR* [81]. Figure 2(c) shows that there are some dataset/bias metric configurations that significantly improve or deteriorate performance for pre- and post-processing techniques, but the majority revolve around 0, i.e., neither improvement nor degradation. For pre-processing techniques, this may be due to the very few degrees of freedom for optimization (e.g., one binary attribute only for four pre-processing algorithms). For **in-processing techniques**, the second observation is that **the general compromise of trading fairness for accuracy appears to hold. However, no strong correlation wrt Pearson's correlation could be identified** in our experiments. The "net balance" between the two is positive in the majority of cases, e.g., the **improvement in fairness gained is more than the loss in accuracy or vice versa**. In Figure 2(c), we see that, except for two algorithms with a clear negative effect with respect to the error rate, most algorithms degrade very little in the majority of experiments in terms of error rate, while most improve in terms of bias.

Guideline 1. *Parameter optimization for fairness-inducing algorithms should be used for better performance in terms of fairness. This especially holds for in-processing techniques, as positive effects are less pronounced on pre- and post-processing techniques. Typically, the loss in accuracy is justifiable by the more substantial increase in fairness.*

Finally, Figure 2(d) illustrates a strong and statistically significant positive correlation of ≈ 0.62 between the bias metric value of the non-optimized algorithm and the difference achieved through optimization. Clearly, the larger the room for improvement (i.e., the higher the bias in the default setting), the larger the potential benefit gained through parameter optimization. Parameter optimization taps the full potential in the sense that **the scale in performance improvement through parameter optimization increases with the possible range of improvement**.

Guideline 2. *Parameter optimization is particularly advisable in settings where, otherwise, a large bias is observed, as its positive effect is larger the higher the bias in a default configuration setting.*

5.2.2 Impact on comparative evaluation. When we mute the parameter optimization component, i.e., run algorithms as in previous studies, we can generally confirm results from previous studies. These include that **fair approaches trade accuracy for fairness [46]** compared to baseline algorithms like *LogisticRegression*. We omitted the visualizations. Furthermore, **in their default setting, algorithms that are designed to tackle a specific metric also improve that metric, while their performance on other metrics is often unpredictable [46]**. However, this does not lead to the takeaway that algorithms designed for specific metrics always outperform other algorithms. For instance, considering equalized odds, *Reweighing* [48] (not designed for this metric) performs better compared to some of the competitors that are designed for that metric. This might relate to the correlation between demographic parity and equalized odds metrics, which is shown in Figure 2(e) (significant strong correlation of ≈ 0.66).

We now revisit the investigation of the absolute performance of algorithms in terms of correctness and fairness, considering different fairness metrics when parameter optimization is applied beforehand. We reuse the same setup as in Sec. 5.2.

As a representative detailed result, Fig. 3 shows the results on the *Communities* dataset for all algorithms for the *equalized odds* and *treatment equality* metrics. The values of the line chart denote the average score, defined by weighing the error rate and corresponding bias metric equally. For each metric and setting of the parameter optimization component (on or off), Fig. 4 highlights the overall performance of all algorithms on all datasets as a heatmap. This serves as a metric-level view on the experimental results. For a better comparison, the heatmap is zoomed in. Thus, very few outliers are not depicted. Furthermore, we also offer an algorithm-level summary view of results across all datasets and metrics in Fig. 5.

The results in Fig. 3 show that for treatment equality, we observe a much stronger fluctuation in performance across experiments than for demographic parity. While algorithms like *FAGTB* [36] or *HGR* [60] perform very well on treatment equality and equalized odds (after parameter optimization), other approaches like *LTDD* [56] struggle to be competitive. However, 5 out of 11 approaches that perform poorly are among those without any tunable parameters, while three additional approaches only have one binary parameter. Thus, with tunable parameters, approaches are more likely to satisfy fairness metrics for which they were originally not optimized. This is further highlighted by the results depicted in Fig. 4. Here, we observe a **positive performance development through parameter optimization for all group fairness metrics, irrespective of algorithm specialization to a specific metric, leading to a higher number of algorithms operating in best-attainable regions**. Treatment equality profits the most as the number of outliers noticeably decreases, and algorithms' performances are mostly bundled in one small region. This is due to the fact that most approaches do not consider this group fairness metric inherently. The second hot zone in the equalized odds results is due to the different datasets, on which the generally performance wrt the error rate differs. In contrast to this result on group fairness metrics, for individual fairness measured using the consistency metric, we do not observe any significant changes when applying parameter optimization compared to default settings. We further note that for consistency, the "densest region" does neither "hit" optimal performance in the error rate nor bias dimension, irrespective of whether an algorithm was optimized for consistency or not. In general, there is no noticeable gap between the performance of the two algorithms optimized for consistency and the majority of the other algorithms, albeit there are minor improvements and a few algorithms that have a big individual bias. Overall, parameter optimization levels the field between algorithms specialized to one metric and non-specialized algorithms when targeting group fairness as significantly more algorithms condense in a "hot zone" of best performance compared to their default configurations. This potentially simplifies the choice of a specific algorithm for a particular application targets, as it is more likely that most algorithms perform comparably well.

Guideline 3. *Parameter optimization reduces the potential impact on performance entailed by the choice of algorithm in applications targeting group fairness. Many algorithms exhibit comparable performance profiles, irrespective of their specialization to a particular group fairness metric.*

Across all experiments, no clear winner that works well for all metrics can be determined. This is further underlined by Fig. 5, which shows that many algorithms perform similarly across all experiments. Islam et al. [46] concluded in their evaluation that pre- and in-processing algorithms typically perform better than post-processing algorithms. Looking at our results using parameter optimization and considering a larger set of algorithms for each processing stage, we can refute the general takeaway given by Islam et al. as we observe that in the majority of experiments, **there is no noticeable difference between bias mitigation algorithms of the different processing stages when the algorithms are properly parameterized**. We explain this revision of a previous result by the "homogenization" supported by parameter optimization of different algorithms' performance (see discussion above)

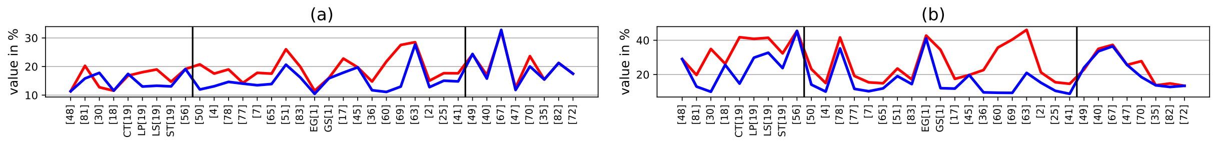
Figure 3: Results on the communities dataset, before (red) and after (blue) applying hyperparameter tuning, with (a) equalized odds and (b) treatment equality as chosen metrics. The corresponding line charts show the average score 0.5·error + 0.5·bias.

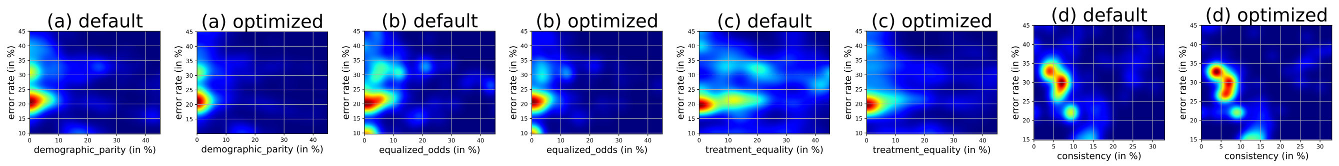
Figure 4: Heatmaps showing the distribution of the results (before and after applying parameter optimization) over all datasets and algorithms for (a) demographic parity, (b) equalized odds, (c) treatment equality, (d) consistency.

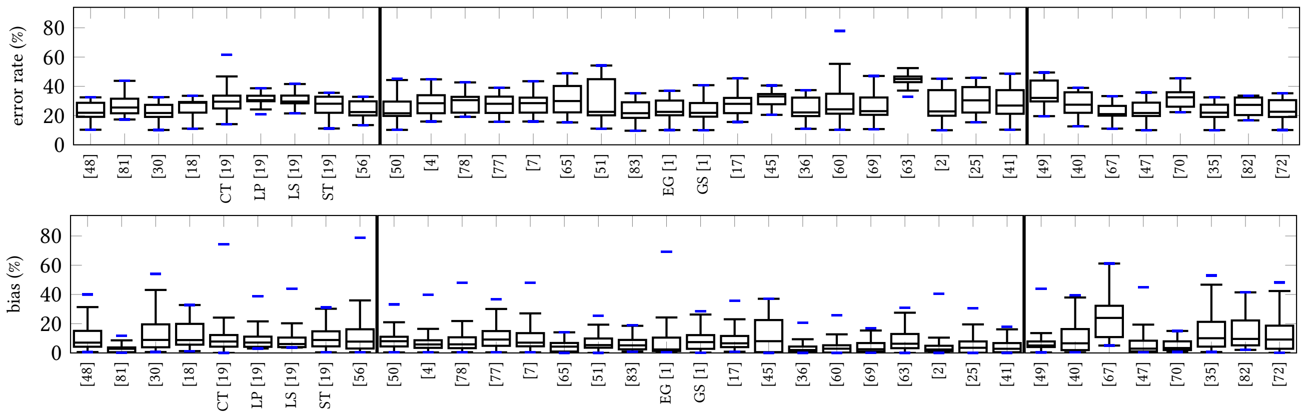
Figure 5: Summary of the overall error rate and bias metric across all metrics and datasets per algorithm

and our larger set of considered algorithms, as, e.g., the outlier post-processing algorithm *CalibratedEqOddsPostprocessing* [67] was part of the small set of considered post-processing algorithms in [46].

Guideline 4. *We cannot state any preference for or superiority of particular types of bias-mitigation algorithms when these are optimally parameterized.*

5.3 Effect of data preparation (RQ2)

This section focuses on the effect of different configuration options in the data preparation component. Due to the high complexity, we split this section into two parts: (1) data preparation steps tied to fairness, i.e., removal of protected attributes and binarization of the protected groups - with the other preparation steps turned off; and (2) general data preparation steps (balancing, feature selection, dimensionality reduction) - with the fairness-related preparation steps turned off. We report the results when parameter optimization is activated. Furthermore, we do not combine multiple fairness algorithms in this set of experiments.

5.3.1 Fairness-related data preparation steps. According to Tab. 1 and our discussion in Sec. 3, not all algorithms are amenable to the fairness-tied data preparation steps. Thus, the experiments are limited to the algorithms highlighted in Tab. 1 allowing some variation in these data preparation steps. Concerning the datasets, the binarization experiments are further limited to the *COMPAS* and *Adult Data Set*, as these are the only datasets with four protected groups to which binarization can be applied. For the evaluation, we ran the bias metrics over all groups.

	[48]	[81]	[18]	CT[19]	LP[19]	LS[19]	ST[19]	[51]	EG[1]	GS[1]	[36]	[49]	[40]	[67]	[47]	[70]
Binarization	×	×	●	-	-*	-*	-*	●	🟥	●	×	×	×	×	●	●
Removal	🟥	-	-	🟥	-*	-*	-*	×	🟥	-	🟥	🟥	●	●	🟥	×

Table 6: Fair data preparation configuration recommendations. Legend: 🟥: Deactivate, ●: Activate, -: Free choice, *: High uncertainty, ×: Activation not possible

We run all possible experiments, i.e., whenever applicable (see Tab. 1), we consider both available options (on or off) for the binarization and protected attribute removal steps. This leaves us with two or four configurations per algorithm, dataset, and metric.

Table 6 summarizes the result of our analysis of the performance data across all these experiments (raw data being available in our repository). It **provides general configuration recommendations for each algorithm** based on the configurations performing among the best across at least 80% of experiments. A result is classified among the top, if its score, defined by an average of 50% error rate and 50% bias for the chosen metric, is within at most 2% of the optimal solution. For instance, for *Fair-SMOTE* [18], for which both binarization and removal of protected attributes can be switched on or off, we observe that, in general, the removal of protected attributes vs keeping them does not make a significant difference, while binarization should be switched on. While the table summarizes general recommendations, the detailed results are, of course, more nuanced. As an example, *FairSSL-LP* achieves better results on the *Adult Data Set* when activating the removal component (for the demographic parity and equalized odds metric), while it performs better on the *COMPAS* data when deactivating it (for all metrics). Such entries with an asterisk denote a high uncertainty, i.e., no configuration combination is among the top results in at least 80% of the experiments.

Removing protected attributes from the training phase rarely boosts the quality of fairness algorithms (14%) and the quality might degrade in some cases as the algorithms are designed to reduce bias, irrespective of the protected attributes. The exceptions are the two post-processing algorithms *EqOddsPostprocessing* [40] and *CalibratedEqOddsPostprocessing* [67].

Guideline 5. *Removing protected attributes from the training phase is generally not recommended.*

For 50% of the algorithms, **binarizing the datasets tends to improve the overall results**. Only the *ExponentiatedGradientReduction* [1] clearly achieves better results (in all experiments) when all protected groups are treated separately. In the previous experiments, this algorithm showcased good results when considering *demographic parity* or *equalized odds*. Thus, we can recommend this algorithm for handling classification problems with multiple protected groups if we want to optimize for one of these metrics. For *FairSSL* [19] no clear statement can be made. This opens a new discussion on whether handling multiple protected groups is necessary or advisable.

Guideline 6. *Binarization can be considered as a viable option to broaden the possible set of algorithm choices for different application needs.*

5.3.2 General data preparation steps. We now focus on the experimental results for balancing, feature selection, and dimensionality reduction. Each preparation step can be combined with each algorithm, which leaves us with 18 configurations per algorithm, dataset, and metric. We show the results when parameter optimization is activated, while we do not combine multiple fairness algorithms and deactivate the fairness-tied preparation techniques.

Table 7 depicts data preparation configuration recommendations for each algorithm, further differentiated depending on the fairness metric (last row). In some cases, two of the three choices perform equally well, e.g., ■/♦ denotes that both deactivating the feature selection component and choosing the *RFECV* feature selection strategy work well. We use the same methodology as in our summarization of fairness-related data preparation steps, thereby recommending a configuration if its score, defined by an average of 50% error rate and 50% bias for the chosen metric, is within at most 2% of the optimal solution in at least 80% of the experiments. We again acknowledge that, while offering some general guidance, these summaries possibly hide a few exceptions where other configurations contribute to better results. For some combinations of algorithm/metric, several configurations can be suggested. For instance, if we look at the consistency metric for the *Reweighing* [48] algorithm, all experiments achieve a good performance as long as the data is not balanced beforehand. Thus, all configuration recommendations for the other metrics also apply to the consistency metric. Likewise, for some combinations of algorithm/metric, no configuration can be recommended, as the results highly fluctuate and highly depend on the chosen dataset. For example, for *ExponentiatedGradientReduction* [1], recommendations can only be made for the demographic parity and equalized odds metrics.

The results show that the **common *class balance* rarely improves the overall results**, as it only appears in recommendations for ≈ 5.5% of the algorithms and only for certain metrics. The reason is that it does not take protected groups into consideration, possibly leading to even higher discrepancy between the protected groups. The adapted *group balance* method is better suited than the class balance approach (appears in 25% of the algorithms) as it takes into consideration both groups and labels. However, this sampling approach does not consider possibly harming correlations to other non-protected attributes. Thus, it is often outperformed by not balancing the data at all.

Guideline 7. *In general, there is no need to perform balancing beforehand when applying fairness-aware binary classification algorithms.*

The supervised feature selection approach *RFECV* performs better across all experiments than the unsupervised feature selection technique *VT* since it uses the information of the label as an advantage. However, the overall results showcase that **a general statement on the influence or preference of considered feature selection approaches cannot be made**, as both methods, as well as deactivating the component, appear frequently in the recommendations. Considering **dimensionality reduction**, in many cases, **turning this component on/off does not majorly impact the overall results**. We explain this by the fact that dimensionality reduction aims to retain as much information as possible of the original data. However, we can see discrepancies in some datasets.

Guideline 8. *Feature selection and dimensionality reduction have a noticeable and varying effect on fair classification results, preventing a general recommendation but showcasing that it is important to consider such data preparation steps as an integral part of the problem when studying fair classification.*

5.4 Combining fairness algorithms (RQ3)

We have seen that applying fairness-inducing techniques can significantly reduce bias while retaining reasonably high accuracy based on parameter optimization, while the results for the data preparation components are mixed. Now, we study what happens when we combine algorithms from different processing stages. For this set of experiments, we combine several algorithms of the pre-processing phase (Fair-SMOTE, FairSSL-*, LTDD, and LFR) with the other algorithms for the in-processing and post-processing stages. Their

	[48]			[81]		[30]	[18]		CT[19]	LP[19]		LS[19]		ST[19]	[56]
Balancing	■	■	■	▲	■	■	■	■	▲	▲	⬟	▲	▲	■	■
Feat. sel.	-	■/♦	✳	✳	■	♦	♦	-	♦	-	♦	■	✳/♦	♦	■
Dim. red.	●	■	■	●	■	●	■	●	■	■	■	■	●	■	■
	◍	◑	◒	◓	◒	●	◷	◶	◶	◴	◑	◓	◶	◵	◍

	[50]	[4]		[78]	[77]	[7]	[65]	[51]			[83]	EG[1]	GS[1]		[17]	[45]	[36]	[60]	[69]	[63]		[2]	[25]	[41]	
Balancing	■	■	⬟	■	■	■	■	▲	▲	■	■	■	■	■	■	■	■	■	■	▲	■	■	■	■	■
Feat. sel.	♦	♦	■	✳/♦	✳	✳/♦	-	♦	■/♦	✳	■/♦	■/♦	■	✳	-	♦	♦	■	♦	✳	✳	♦	■	■	■
Dim. red.	■	■	■	■	●	■	●	●	■	●	-	-	■	●	-	■	■	●	■	●	●	-	-	■	●
	◍	◕	◵	●	●	◕	◍	◷	◵	◶	◶	◓	◕	◵	◶	◶	●	◕	◍	◕	◕	◕	●	◐	◷

	[49]	[47]	[70]		[35]	[72]		[82]
Balancing	▲	■	■	▲	■	▲	■	■
Feat. sel.	-	♦	■/♦	■/♦	✳	■	-	■/♦
Dim. red.	●	■	■	-	■	■	●	●
	◶	●	◐	◷	●	◍	◵	◶

Table 7: Classical data preparation configuration recommendations and a respective certainty for each metric. Legend: ■: Deactivate, ⬟: Class bal., ▲: Group bal., ✳: VT, ♦: RFECV, ●: PCA, -: Free choice. If the corresponding quarter of a metric is filled, then these recommendations apply, else they were too inconclusive/unstable: demographic parity: ◴, equalized odds: ◷, treatment equality: ◵, consistency: ◶

combination can be achieved by applying the pre-processing algorithm and then using the updated training dataset as input for an in-processing or post-processing algorithm. To obtain optimal parameters, it would be necessary to combine the parameters for both the pre-processing and in-/post-processing algorithm to achieve optimal results. This would immensely increase the grid size and the computational cost. Thus, when the parameter optimization component is not muted, we optimize the parameters consecutively: we first optimize the pre-processing algorithm, and then we use the optimized dataset output as input for the in-processing/post-processing algorithms, on which we perform parameter optimization as well. We compare the results of the combinations with the corresponding baselines, e.g., *LFR-HGR* will be compared to *LFR* and *HGR*. No data preparation step is applied.

Fig. 6 depicts representative detailed results on the *Communities* dataset when we combine multiple algorithms. The heatmap indicates if there has been an improvement by combining multiple methods (dark green for improvements of ≥ 30%) or if there is an increase in the error rate or bias (dark red for an increase of ≥ 30%). The results of the combined algorithm are compared to the better of the two individual algorithms. For instance, *LFR* achieves lower bias than *HGR*, but has a higher error rate. Thus, *LFR-HGR* is compared to *LFR* in terms of bias and to *HGR* in terms of error rate. In the heatmaps a few values are missing. This is due to errors thrown due to some pre-processing algorithms altering the data so that it is no longer usable for the subsequent algorithm.

We first notice that **combining approaches generally does not pay off** (a result we confirmed in general across all experiments). Throughout all experiments, the error rate has a high tendency to increase. For treatment equality, we see that the tendency of an improvement, if any, is higher than for demographic parity. However, looking at the previous experiment's result in Fig. 3(b), we see that most of the algorithms where the bias seems to improve through combination did perform poorly (even after boosting the results via hyperparameter tuning). For instance, *LTDD* [56] and *ExponentiatedGradientReduction* [1] performed well on the demographic parity metric, but poorly on the treatment equality metric. Combining these methods decreases the overall bias and achieves an improvement over their baseline for treatment equality as chosen fairness metric. However, a general consensus is that there are still better options where solely applying parameter optimization on a single classifier achieve better results. The experiments on other datasets validate this.

Guideline 9. *Combining approaches from different stages is not advisable, as there is a high tendency to worsen the results.*

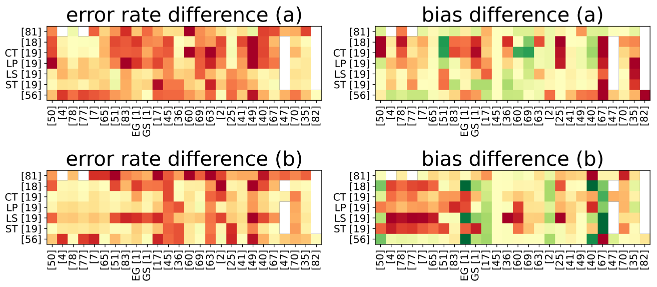

Figure 6: Result of the combination experiment on the communities dataset for (a) demographic parity and (b) treatment equality. Dark green indicates an improvement of ≥ 30% and dark red indicates an increase in bias or error rate of ≥ 30%.

6 CONCLUSION

We studied the performance of fair classification algorithms within a multi-step pipeline based on an extensive set of experiments that covered various implementation configurations of the pipeline.

Our experimental study allowed us to gain several new insights. Hyperparameter tuning proves to be an important component and makes algorithms competitive particularly for fairness metrics they were not originally designed for. The data preparation subcomponent of ignoring the protected attributes is rarely advisable, either due to very fluctuating and unpredictable behavior or little effect. The data preparation subcomponent of binarizing the protected groups did not generally have a negative impact. Thus, when datasets comprise multiple or non-binary protected attributes, no algorithm has to be dismissed a priori, as binarization does not degrade results in general. For the common data preparation steps, the optimal configuration is highly dependent on the chosen algorithm. However, unsupervised feature selection techniques and balancing the data are rarely advised. While individual tuned models achieve good results, combining several approaches typically degrades the result performance. Using altered training datasets (by the pre-processing algorithm) as input for the in- and post-processing algorithms especially takes a toll on the accuracy, while the room for improvement in regards to fairness is often small. While there are always exceptions for each of these takeaways, these results allow us to give some general guidelines to practitioners facing the problem of choosing an appropriate algorithm and pipeline configuration for their application requirements.

This paper shows that studying fair classifications in the broader context of an end-to-end processing pipeline allows us to gain new insights relevant to building robust pipelines in practice. We hope that this work will spawn follow-up research to better, that can take advantage of our methodology and evaluation framework.

REFERENCES

[1] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudík, John Langford, and Hanna Wallach. 2018. A reductions approach to fair classification. In *International Conference on Machine Learning*. PMLR, 60–69.

[2] Ari Ball-Burack, Michelle Seng Ah Lee, Jennifer Cobbe, and Jatinder Singh. 2021. Differential tweetment: Mitigating racial dialect bias in harmful tweet detection. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. 116–128.

[3] Gustavo EAPA Batista, Ana LC Bazzan, Maria Carolina Monard, et al. 2003. Balancing training data for automated annotation of keywords: a case study. *Wob* 3 (2003), 10–8.

[4] Yahav Bechavod and Katrina Ligett. 2017. Penalizing unfairness in binary classification. *arXiv preprint arXiv:1707.00044* (2017).

[5] Rachel K. E. Bellamy, Kuntal Dey, Michael Hind, Samuel C. Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, Aleksandra Mojsilovic, Seema Nagar, Karthikeyan Natesan Ramamurthy, John Richards, Diptikalyan Saha, Prasanna Sattigeri, Moninder Singh, Kush R. Varshney, and Yunfeng Zhang. 2018. AI Fairness 360: An Extensible Toolkit for Detecting, Understanding, and Mitigating Unwanted Algorithmic Bias. https://arxiv.org/abs/1810.01943

[6] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of machine learning research* 13, 2 (2012).

[7] Richard Berk, Hoda Heidari, Shahin Jabbari, Matthew Joseph, Michael Kearns, Jamie Morgenstern, Seth Neel, and Aaron Roth. 2017. A convex framework for fair regression. *arXiv preprint arXiv:1706.02409* (2017).

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

[9] Peter J Bickel, Eugene A Hammel, and J William O'Connell. 1975. Sex Bias in Graduate Admissions: Data from Berkeley: Measuring bias is harder than is usually assumed, and the evidence is sometimes contrary to expectation. *Science* 187, 4175 (1975), 398–404.

[10] Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. 2020. Language (technology) is power: A critical survey of" bias" in nlp. *arXiv preprint arXiv:2005.14050* (2020).

[11] Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*. 92–100.

[12] Tim Brennan, William Dieterich, and Beate Ehret. 2009. Evaluating the predictive validity of the COMPAS risk and needs assessment system. *Criminal Justice and behavior* 36, 1 (2009), 21–40.

[13] US Census Bureau. 2019. US Census Demographic Data. https://www.kaggle.com.

[14] Toon Calders and Sicco Verwer. 2010. Three naive bayes approaches for discrimination-free classification. *Data mining and knowledge discovery* 21 (2010), 277–292.

[15] Julia Carpenter. 2015. *Google's algorithm shows prestigious job ads to men, but not to women. Here's why that should worry you.* https://www.washingtonpost.com/news/the-intersect/wp/2015/07/06/googles-algorithm-shows-prestigious-job-ads-to-men-but-not-to-women-heres-why-that-should-worry-you/

[16] Simon Caton and Christian Haas. 2020. Fairness in machine learning: A survey. *arXiv preprint arXiv:2010.04053* (2020).

[17] L Elisa Celis, Lingxiao Huang, Vijay Keswani, and Nisheeth K Vishnoi. 2019. Classification with fairness constraints: A meta-algorithm with provable guarantees. In *Proceedings of the conference on fairness, accountability, and transparency*. 319–328.

[18] Joymallya Chakraborty, Suvodeep Majumder, and Tim Menzies. 2021. Bias in machine learning software: Why? how? what to do?. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 429–440.

[19] Joymallya Chakraborty, Suvodeep Majumder, and Huy Tu. 2022. Fair-SSL: Building fair ML Software with less data. In *Proceedings of the 2nd International Workshop on Equitable Data and Technology*. 1–8.

[20] Anshuman Chhabra, Karina Masalkovaitė, and Prasant Mohapatra. 2021. An overview of fairness in clustering. *IEEE Access* 9 (2021), 130698–130720.

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

[22] Thomas Cover and Peter Hart. 1967. Nearest neighbor pattern classification. *IEEE transactions on information theory* 13, 1 (1967), 21–27.

[23] Jeffrey Dastin. 2018. Amazon scraps secret AI recruiting tool that showed bias against women. Reuters. *Reuters.com* (2018). https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-thatshowed-bias-against-women-idUSKCN1MK08G

[24] Hyungrok Do and Preston Putzel. 2022. Fair Generalized Linear Models with a Convex Penalty. https://github.com/hyungrok-do/fair-glm-cvx.

[25] Hyungrok Do, Preston Putzel, Axel S Martin, Padhraic Smyth, and Judy Zhong. 2022. Fair Generalized Linear Models with a Convex Penalty. In *International Conference on Machine Learning*. PMLR, 5286–5308.

[26] Dheeru Dua and Casey Graff. 2017. Adult Data Set. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml/datasets/adult

[27] Dheeru Dua and Casey Graff. 2017. Communities and Crime Data Set. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml/datasets/communities+and+crime

[28] Dheeru Dua and Casey Graff. 2017. default of credit card clients Data Set. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients

[29] Dheeru Dua and Casey Graff. 2017. Statlog (German Credit Data) Data Set. UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)

[30] 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.

[31] James R Foulds, Rashidul Islam, Kamrun Naher Keya, and Shimei Pan. 2020. An intersectional definition of fairness. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 1918–1921.

[32] 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.

[33] Ismael Garrido-Muñoz, Arturo Montejo-Ráez, Fernando Martínez-Santiago, and L Alfonso Ureña-López. 2021. A survey on bias in deep nlp. *Applied Sciences* 11, 7 (2021), 3184.

[34] Aurélien Géron. 2019. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems.* O'Reilly Media.

[35] Przemyslaw A Grabowicz, Nicholas Perello, and Aarshee Mishra. 2022. Marrying fairness and explainability in supervised learning. In *2022 ACM Conference on Fairness, Accountability, and Transparency*. 1905–1916.

[36] Vincent Grari, Boris Ruf, Sylvain Lamprier, and Marcin Detyniecki. 2019. Fair adversarial gradient tree boosting. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1060–1065.

[37] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. 2002. Gene selection for cancer classification using support vector machines. *Machine learning* 46 (2002), 389–422.

[38] Maryam Amir Haeri and Katharina Anna Zweig. 2020. The crucial role of sensitive attributes in fair classification. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2993–3002.

[39] Sara Hajian and Josep Domingo-Ferrer. 2013. Direct and indirect discrimination prevention methods. In *Discrimination and privacy in the information society*. Springer, 241–254.

[40] Moritz Hardt, Eric Price, and Nati Srebro. 2016. Equality of opportunity in supervised learning. *Advances in neural information processing systems* 29 (2016).

[41] Corinna Hertweck and Tim Räz. 2022. Gradual (in) compatibility of fairness criteria. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 11926–11934.

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

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

[44] Vasileios Iosifidis and Eirini Ntoutsi. 2019. Adafair: Cumulative fairness adaptive boosting. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 781–790.

[45] Vasileios Iosifidis, Arjun Roy, and Eirini Ntoutsi. 2022. Parity-based cumulative fairness-aware boosting. *Knowledge and Information Systems* 64, 10 (2022), 2737–2770.

[46] Maliha Tashfia Islam, Anna Fariha, Alexandra Meliou, and Babak Salimi. 2022. Through the data management lens: Experimental analysis and evaluation of fair classification. In *Proceedings of the 2022 International Conference on Management of Data*. 232–246.

[47] Heinrich Jiang and Ofir Nachum. 2020. Identifying and correcting label bias in machine learning. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 702–712.

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

[49] 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.

[50] Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma. 2012. Fairness-aware classifier with prejudice remover regularizer. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2012, Bristol, UK, September 24-28, 2012. Proceedings, Part II 23*. Springer, 35–50.
[51] Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. 2018. Preventing fairness gerrymandering: Auditing and learning for subgroup fairness. In *International conference on machine learning*. PMLR, 2564–2572.
[52] Amir E Khandani, Adlar J Kim, and Andrew W Lo. 2010. Consumer credit-risk models via machine-learning algorithms. *Journal of Banking & Finance* 34, 11 (2010), 2767–2787.
[53] Tai Le Quy, Arjun Roy, Vasileios Iosifidis, Wenbin Zhang, and Eirini Ntoutsi. 2022. A survey on datasets for fairness-aware machine learning. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 12, 3 (2022), e1452.
[54] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P Trevino, Jiliang Tang, and Huan Liu. 2017. Feature selection: A data perspective. *ACM computing surveys (CSUR)* 50, 6 (2017), 1–45.
[55] Lan Li, Tina Lassiter, Joohee Oh, and Min Kyung Lee. 2021. Algorithmic hiring in practice: Recruiter and HR Professional's perspectives on AI use in hiring. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*. 166–176.
[56] Yanhui Li, Linghan Meng, Lin Chen, Li Yu, Di Wu, Yuming Zhou, and Baowen Xu. 2022. Training data debugging for the fairness of machine learning software. In *Proceedings of the 44th International Conference on Software Engineering*. 2215–2227.
[57] Kristian Lum and William Isaac. 2016. To predict and serve? *Significance* 13, 5 (2016), 14–19.
[58] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems* 30 (2017).
[59] Timo Makkonen. 2002. Multiple, compoud and intersectional discrimination: bringing the experiences of the most marginalized to the fore. (2002).
[60] Jérémie Mary, Clément Calauzenes, and Noureddine El Karoui. 2019. Fairness-aware learning for continuous attributes and treatments. In *International Conference on Machine Learning*. PMLR, 4382–4391.
[61] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)* 54, 6 (2021), 1–35.
[62] Thomas Minka. 2000. Automatic choice of dimensionality for PCA. *Advances in neural information processing systems* 13 (2000).
[63] Luca Oneto, Michele Donini, and Massimiliano Pontil. 2020. General fair empirical risk minimization. In *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.
[64] Judea Pearl. 2009. *Causality*. Cambridge university press.
[65] Adrián Pérez-Suay, Valero Laparra, Gonzalo Mateo-García, Jordi Muñoz-Marí, Luis Gómez-Chova, and Gustau Camps-Valls. 2017. Fair kernel learning. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2017, Skopje, Macedonia, September 18–22, 2017, Proceedings, Part I*. Springer, 339–355.
[66] Dana Pessach and Erez Shmueli. 2022. A Review on Fairness in Machine Learning. *ACM Computing Surveys (CSUR)* 55, 3 (2022), 1–44.
[67] Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q Weinberger. 2017. On fairness and calibration. *Advances in neural information processing systems* 30 (2017).
[68] ProPublica. 2017. COMPAS Recidivism Racial Bias. https://www.kaggle.com.
[69] Yaniv Romano, Stephen Bates, and Emmanuel Candes. 2020. Achieving equalized odds by resampling sensitive attributes. *Advances in neural information processing systems* 33 (2020), 361–371.
[70] Nicolas Schreuder and Evgenii Chzhen. 2021. Classification with abstention but without disparities. In *Uncertainty in Artificial Intelligence*. PMLR, 1227–1236.
[71] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. 2015. Taking the human out of the loop: A review of Bayesian optimization. *Proc. IEEE* 104, 1 (2015), 148–175.
[72] Sandipan Sikdar, Florian Lemmerich, and Markus Strohmaier. 2022. GetFair: Generalized Fairness Tuning of Classification Models. In *2022 ACM Conference on Fairness, Accountability, and Transparency*. 289–299.
[73] Ž Vujović et al. 2021. Classification model evaluation metrics. *International Journal of Advanced Computer Science and Applications* 12, 6 (2021), 599–606.
[74] Li Yang and Abdallah Shami. 2020. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* 415 (2020), 295–316.
[75] David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*. 189–196.
[76] Tjalling J Ypma. 1995. Historical development of the Newton–Raphson method. *SIAM review* 37, 4 (1995), 531–551.
[77] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P Gummadi. 2017. Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. In *Proceedings of the 26th international conference on world wide web*. 1171–1180.
[78] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rogriguez, and Krishna P Gummadi. 2017. Fairness constraints: Mechanisms for fair classification. In *Artificial intelligence and statistics*. PMLR, 962–970.
[79] Meike Zehlike, Ke Yang, and Julia Stoyanovich. 2022. Fairness in ranking, part I: Score-based ranking. *Comput. Surveys* 55, 6 (2022), 1–36.
[80] Meike Zehlike, Ke Yang, and Julia Stoyanovich. 2022. Fairness in ranking, part ii: Learning-to-rank and recommender systems. *Comput. Surveys* 55, 6 (2022), 1–41.
[81] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. 2013. Learning fair representations. In *International conference on machine learning*. PMLR, 325–333.
[82] Xianli Zeng, Edgar Dobriban, and Guang Cheng. 2022. Fair Bayes-Optimal Classifiers Under Predictive Parity. *arXiv preprint arXiv:2205.07182* (2022).
[83] 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.
[84] Dengyong Zhou, Olivier Bousquet, Thomas Lal, Jason Weston, and Bernhard Schölkopf. 2003. Learning with local and global consistency. *Advances in neural information processing systems* 16 (2003).
[85] Xiaojin Zhu and Zoubin Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation. *ProQuest Number: INFORMATION TO ALL USERS* (2002).
[86] Indrė Žliobaitė. 2017. Measuring discrimination in algorithmic decision making. *Data Mining and Knowledge Discovery* 31, 4 (2017), 1060–1089.

A ADDITIONAL EXPERIMENTS

We further conducted other experiments, which regard the performance on the synthetic datasets with a varying bias and experiments regarding the scalability of these methods.

A.1 Types of bias suited for mitigation

For the experiments within this paper, we evaluate the algorithms on real-world data as well as two synthetic datasets. Each exhibits one of two biases: *Social* bias induces direct bias, while for the *implicit* bias datasets indirect bias was induced. We used the datasets that have a 30% difference in bias for the groups. For this set of experiments, we vary the bias between 10% and 50%. To better understand what type of bias may be "addressed" best by the individual algorithms, we now apply the algorithms to the synthetic datasets. The parameter optimization component is activated, while the preparation components are muted. To compare the bias mitigation algorithms, we further use two *Logistic Regression* (short: *LR*) classifiers as the baselines that are trained for optimizing accuracy and are applied on the original data. Hereby, we provide another version where we set the component to ignore the protected attributes to true, resulting in *LogisticRegressionRemoved* (short: *LRR*).

For each fairness metric, Fig. 7 depicts boxplots that show the evolution of the difference in performance between the *LRR* baseline and optimized algorithms when increasing the degree of bias in the generated data (from 10 to 50 %). In these boxplots, for less clutter, the error bars directly denote the minimum/maximum value over all algorithms. As before, positive values indicate improvements over *LRR*.

As we can see, (optimized) fair classification algorithms are especially effective in tackling implicit bias globally. Indeed, while the error rate slightly increases with increased bias within the input data, the improvements of bias increase the more bias is present in the data for the majority of the algorithms. This holds for demographic parity, equalized odds, and, to a lesser extent, for treatment equality. The picture is different when focusing on the individual fairness metric of consistency. Here, we observe very little change in the opposite direction, i.e., performance gradually degrades as

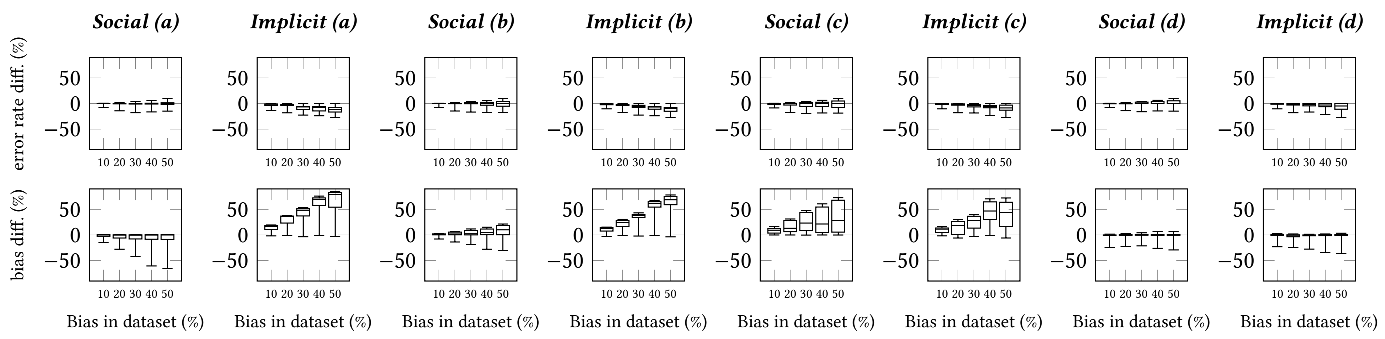

Figure 7: Boxplot of the bias mitigation algorithms compared to *LRR* in terms of error rate and group bias applied with (a) demographic parity, (b) equalized odds, (c) treatment equality, (d) consistency as chosen group fairness metric (positive values depict improvements over the baseline LRR algorithm)

implicit bias increases. This is in accord with our earlier observations that *LRR* already appears to be a good enough choice when aiming for consistency. On the artificial datasets, *LRR* has achieved a low bias (up to 7.2%), leaving all algorithms only little room for improvement. While 33 of the 36 algorithms we study are not specialized for individual fairness, we verified that this statement also holds when focusing on the specialized algorithms.

The error rate remains stable across all experiments on the social datasets. For the fairness metrics, the results vary. *LRR* is the method of choice when targeting demographic parity, which is not surprising for this artificial dataset since the bias within the dataset is *solely* caused by the protected attribute. Otherwise, if *LR* is chosen instead of *LRR* as the baseline, we can see huge improvements by the fairness classifiers. For this metric, most approaches have similar results, with a tendency to slightly increase the overall bias, and very few algorithms to increase the bias significantly. When optimizing (on the social datasets) for equalized odds, slight improvements can be detected. When optimizing for treatment equality, we observe a wide range of (mostly) improved performance in fairness that increases with increasing social bias in the input data. The discussion for consistency is analogous to its discussion of the datasets with implicit bias.

On the real-world datasets, we see a varying amount of improvement over *LRR*, although an improvement is generally identified.

Guideline 10. *The type of bias within the datasets greatly influences the overall quality improvements of fairness-inducing approaches. The approaches typically shine in tackling implicit bias when focusing on group bias. For individual fairness quantified using the consistency metric, LRR typically achieves satisfactory performance in most cases.*

A.2 Scalability of fair approaches

While previous experiments have proven the need for bias mitigation algorithms and have shown the utter importance of parameter optimization (for some of the algorithms/settings), we want to test the scalability of these algorithms. In contrast to the other experimental paper, we do not only evaluate the runtime, but the memory

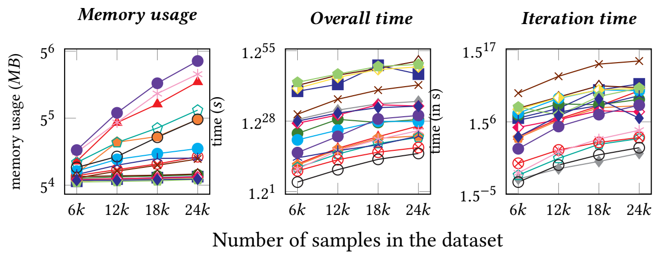

Figure 8: Results of the memory usage and runtime experiments. (1) LFR (■); (2) FairSSL-LP (⬟); (3) FairSSL-LS (▲); (4) PrejudiceRemover (●); (5) SquaredDifferenceFairLogistic (○); (6) ConvexFrameworkModel (✳); (7) DisparateMistreatmentModel (●); (8) FairnessConstraintModel (⊗); (9) AdaFair (◆); (10) HGR (◆); (11) FairDummies (◆); (12) MultiAdversarialDebiasing (◆); (13) FairGeneralizedLinearModel (⬠); (14) GradualCompatibility (⬟); (15) RejectOptionClassification (⅄); (16) DPAbstention (●); (17) FairBayesDPP (×); (18) GetFair (◆)

usage as well. For this set of experiments, we scaled the COMPAS dataset and evaluated how the memory usage and runtime scales with the increase of the dataset. We choose to scale the small dataset, as the results show the trend when we increase the sample amount. We also received out-of-memory errors on some of the larger real-world datasets for some of these algorithms. In this set of experiments, we activate the parameter optimization component and measure the overall runtime and the runtime of a single iteration. We choose the runtime of the optimal parameter settings for the single iteration.

The results of the experiments are depicted in Fig. 8. The graph only depicts algorithms that did not scale well with respect to one of the dimensions (memory usage, overall runtime or iteration runtime). Algorithms, like *DPAbstention* [70], *FairSSL-LS* [19] or *ConvexFrameworkModel* [7] require ≈ 10*GB* of memory on datasets with 24*k* samples only. That means that they are barely usable on larger datasets. Furthermore *FairSSL-LP* [19], *FairGeneralizedLinearModel* [25] and *SquaredDifferenceFairLogistic* [4] also scale poorly regarding memory usage. Although their memory usage scales badly, some of the algorithms show good qualitative results

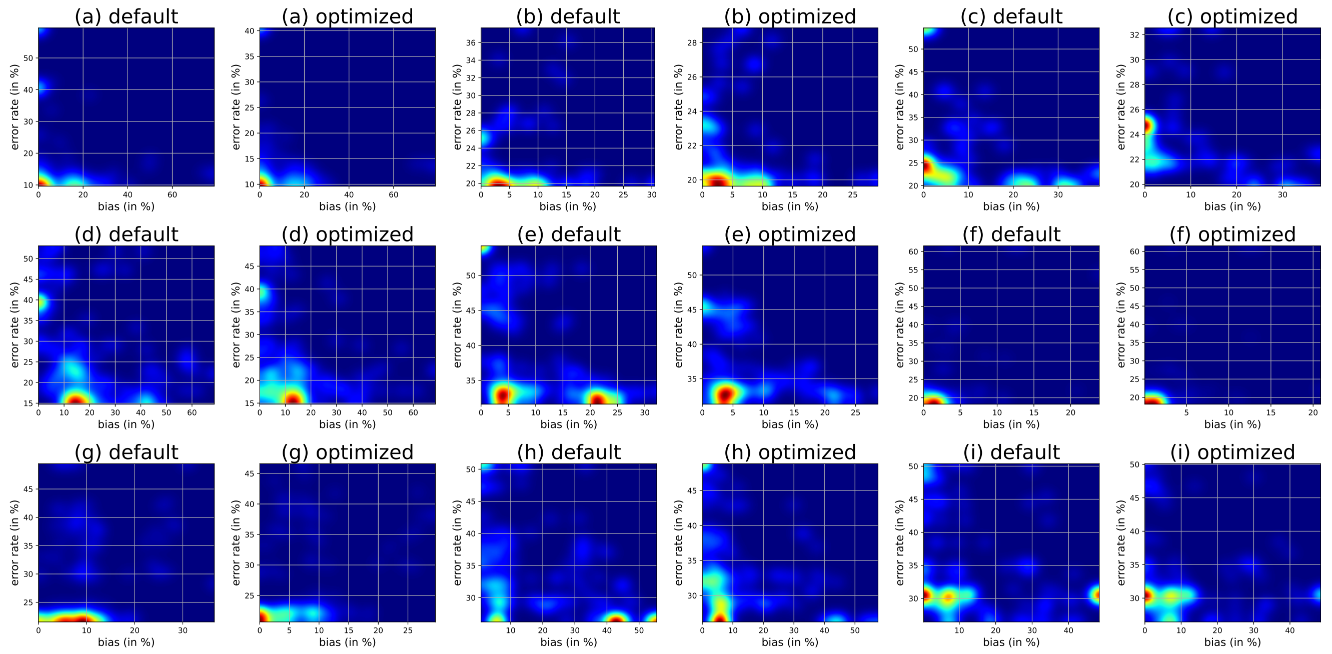

Figure 9: Heatmaps showing the distribution of the results (before and after applying parameter optimization) over all metrics and algorithms for (a) ACS2017, (b) Adult Data (*race* as sensitive), (c) Adult Data (*sex* as sens.), (d) Communities, (e) COMPAS, (f) Credit Card, (g) German, (h) Implicit, (i) Social.

in previous experiments, so future work on these algorithms improving their scalability may be worthwhile.

When the runtime is important, then *LFR* [81], *FairDummies* [69], *GradualCompatibility* [41], *HGR* [60] and *DPAbstention* also have to be reconsidered. In-processing algorithms typically perform worse, due to the amount of tunable parameters and the overall grid size.

In the experiments of Islam et al. [46] the runtime was measured for single iterations. The key takeaway was that post-processing approaches are more efficient than pre- and post-processing algorithms. In our set of experiments, *FairBayesDPP* [82] has the highest runtime of a single iteration. Another post-processing algorithm, *DPAbstention* [70], also requires a comparably high running time. However, in terms of single-iteration runtime, we cannot clearly state which stage scales better on our extended benchmark. With the increased amount of algorithms we consider, we cannot corroborate the general observation of Islam et al. [46] that post-processing algorithms scale better.

Note that we could not identify a correlation between memory usage and runtime.

Guideline 11. *Before applying algorithms, we need to consider the dataset sizes, as some approaches are not suited for big datasets, which limits their applicability. This might be either due to the memory usage or due to their long runtime. Most of the runtime is caused by the parameter tuning step. However, the results have shown partially significant improvements over the default settings, and thus, turning the parameter optimization component off is not a viable option for them.*

B DETAILED RESULTS

In this section, we want to give a different and more detailed view into some of the results.

B.1 Dataset-level view

So far, we have shown a metric-level view in Fig. 4 and an algorithm-level view in Fig. 5. In Fig. 9, we show the dataset-level view of the results over all algorithms and metrics, for the default and parameter optimized solution. This showcases, that the results of the improvement gained via parameter optimization is highly dependent on the datasets as well. While for the *COMPAS* or *Implicit* dataset, the difference is huge, the effect on the *ACS2017* data is a lot less. The overall bias within the data plays another important factor. For the *Adult Data*, we can see big improvements if *sex* is the chosen attribute, while the parameter optimization improvement if *race* is the chosen attribute is a lot less. Taking a look at the metadata of the data (cf. Table 5), we see that there is a higher gap for the probability of a positive outcome for the *sex* attribute than for the *race* attribute.

B.2 Performance of the algorithms per metric

In Fig. 10 – Fig. 13, we show the summarized results of Fig. 5 in a more detailed version. These plots can help the user in guidelining the choice of an appropriate model if a certain fairness metric is given as optimization function.

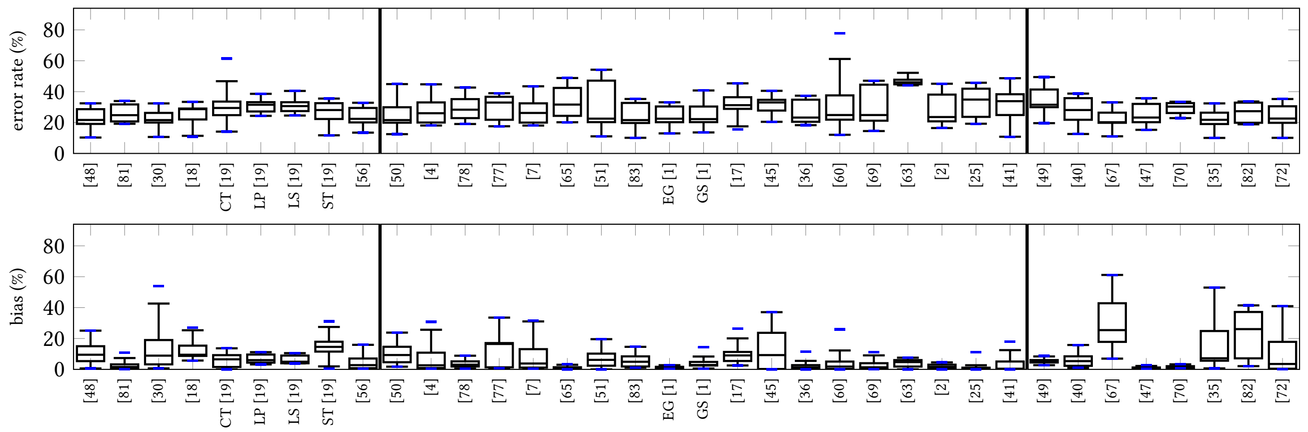

Figure 10: Summary of the overall error rate and bias metric across all datasets per algorithm for the demographic parity metric

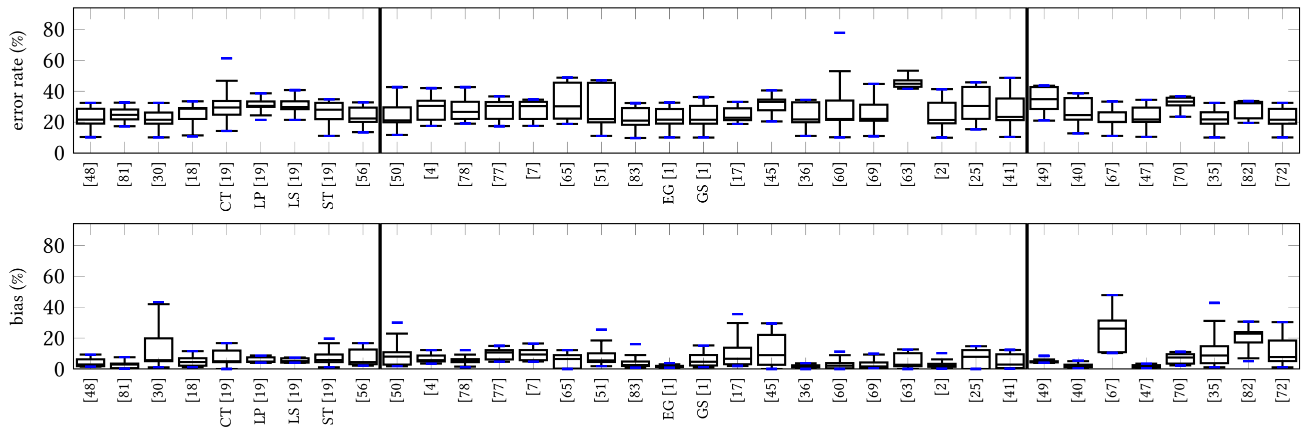

Figure 11: Summary of the overall error rate and bias metric across all datasets per algorithm for the equalized odds metric

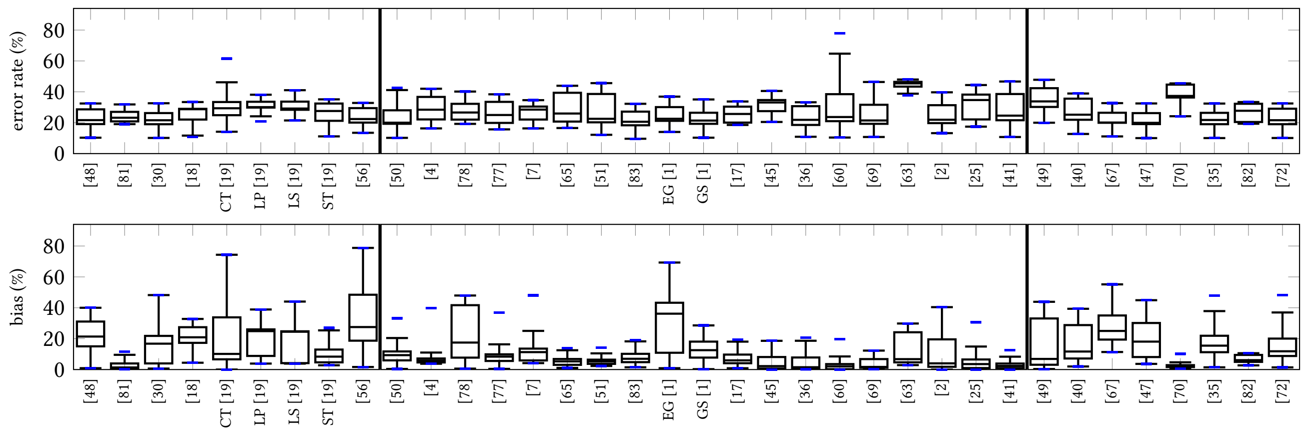

Figure 12: Summary of the overall error rate and bias metric across all datasets per algorithm for the treatment equality metric

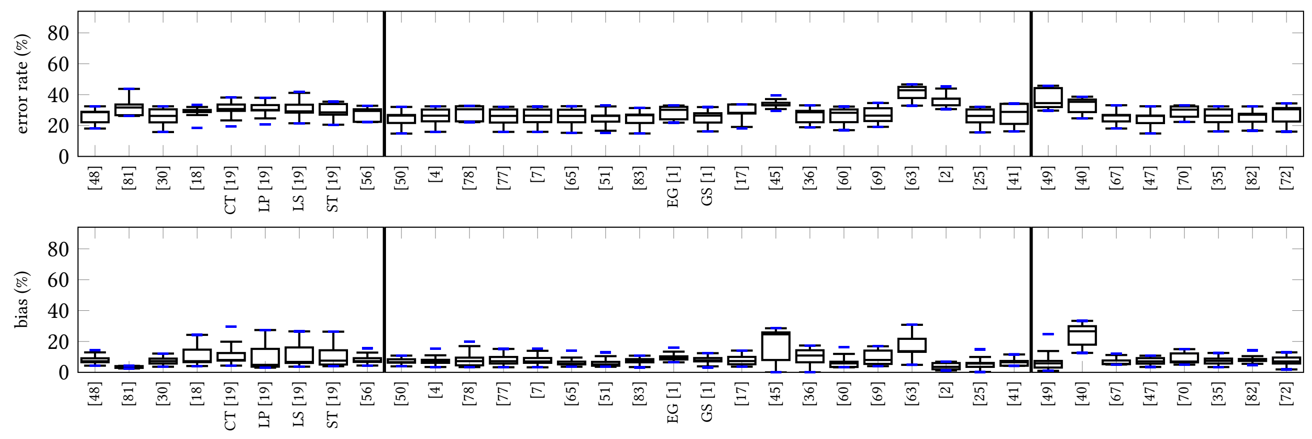

Figure 13: Summary of the overall error rate and bias metric across all datasets per algorithm for the consistency metric