Speeding up fairness reductions

Anonymous authors
Paper under double-blind review

Abstract

We study the problem of fair classification, where the goal is to optimize classification accuracy subject to fairness constraints. This type of problem occurs in many real-world applications, where we seek to assure that a deployed AI system does not disproportionally impact historically disadvantaged groups. One of the leading approaches in the literature is the *reduction approach* (Agarwal et al., 2018; 2019), which enjoys many favorable properties. For instance, it supports a wide range of fairness constraints and model families and is usually easy to incorporate in existing ML pipelines. The reduction approach acts as a wrapper around a standard ML algorithm and obtains a model that satisfies fairness constraints by repeatedly running a fairness-unaware base algorithm. A typical number of iterations is around 100, meaning that the reduction approach can be up to 100 times slower than the base algorithm, which limits its applicability. To overcome this limitation, we introduce two algorithmic innovations. First, we interleave the exponentiated gradient updates of the standard reduction approach with *column-generation updates*, which leads to a decrease in the number of calls to the base algorithm. Second, we introduce *adaptive sampling*, which decreases the sizes of the datasets used in the calls to the base algorithm. We conduct comprehensive experiments to evaluate efficacy of our improvements, showing that our two innovations speed up the reduction approach by an order of magnitude without sacrificing the quality of the resulting solutions.

1 Introduction

As artificial intelligence (AI) systems are deployed in a growing range of applications, there is an increased need to ensure that their deployment does not disproportionately impact historically disadvantaged populations and groups (Crawford, 2013; O'Neil, 2016; Broussard, 2018; Noble, 2018; Benjamin, 2019). Fairness of AI systems is a topic of multiple academic venues,[1] a priority for regulators (The European Parliament & The Council of the European Union, 2024) as well as corporations (Crampton, 2022; Philomin, 2024; Google, 2025; Microsoft, 2025), and a focus of several open source projects (Lee & Singh, 2021).

There are many different kinds of fairness harms (Barocas et al., 2017; Crawford, 2017; Wallach & Dudík, 2021; Shelby et al., 2023), and many different ways to mitigate them by intervening at different points of the AI lifecycle (Wallach & Dudík, 2021), including during task definition, data collection, model training, and after model deployment. Here we study algorithmic techniques that seek to mitigate two broad categories of harms, *allocative harms* and *quality-of-service harms*, at the model training stage. Allocative harms occur when AI systems are used to allocate opportunities or resources in ways that can have significant negative impacts on people's lives such as in hiring, policing, education, and access to health-care (Angwin et al., 2016; Obermeyer et al., 2019; Raghavan et al., 2020; Smith, 2020). Quality-of-service harms occur when a system does not work as well for members of one group as it does for members of another group (Buolamwini & Gebru, 2018; Koenecke et al., 2020).

For example, suppose a hospital is training an AI model to predict 30-day hospital readmission to prioritize high-risk patients for more intensive post-discharge care. Allocative harms might occur if certain subgroups of patients (e.g., Black patients) are disproportionately under-prioritized for more intensive care (i.e., have

[1] `https://facctconference.org/` (FAccT), `https://www.aies-conference.com/` (AIES), `https://responsiblecomputing.org/` (FORC)

low selection rates) or are under-selected relative to their observed rate of readmission (i.e., have high false negative rates); this type of harm has been noted, for example, by Obermeyer et al. (2019). To mitigate this harm, the hospital could train a model that incorporates suitable fairness constraints by constraining, for example, the difference between selection rates across different race groups.

Although there is a large body of research on algorithmic mitigation of fairness harms (Barocas et al., 2019; Pessach & Shmueli, 2023; Caton & Haas, 2023; Mehrabi et al., 2021), there are many frictions in applying these techniques in real world (Holstein et al., 2019), including lack of flexibility in the choice of fairness metrics and model families, lack of compatibility with existing machine learning pipelines, and computational cost of training.

In this paper, we focus on the reduction approach to unfairness mitigation (Agarwal et al., 2018; 2019), which helps address many of these usability frictions. It works with a wide range of fairness definitions, model families, and as a reduction approach, it can be "wrapped" around any existing supervised machine learning approach and so offers the flexibility to improve fairness of existing AI systems without a need to re-architect deployed systems. Because of its flexibility, it has been incorporated in several open-source toolkits, including *fairlearn* (Weerts et al., 2023) and *AIF360* (Bellamy et al., 2019). The main barrier for its broader adoption is that the training algorithm is substantially slower than fairness-unaware algorithms. Our work seeks to overcome this limitation.

We introduce two algorithmic innovations that speed up the reduction approach by several orders of magnitude. For simplicity we focus on the binary classification task and allocational harms and extend the algorithmic approach of Agarwal et al. (2018), but our innovations are also applicable in regression setting with quality-of-service harms and can be used to extend the corresponding approach of Agarwal et al. (2019), which is structurally similar to the one studied here.

The algorithm at the core of the reduction approach is the *exponentiated gradient* of Kivinen & Warmuth (1997), which we refer to as ExpGrad. The goal of ExpGrad is to find a classifier from some family (like linear classifiers or neural nets), which maximizes classification accuracy subject to fairness constraints. ExpGrad operates as a wrapper around any standard classification algorithm for the given family, which we refer to as an *oracle* or a *base algorithm*. To optimize accuracy subject to fairness constraints, ExpGrad repeatedly invokes the oracle on reweighted versions of the training data (much like boosting algorithms, Freund & Schapire, 1997), requiring up to 100 runs of the base algorithm. Thus, its running time is around 100 times slower than the running time of fairness-unaware methods.

Our first innovation is in decreasing the number of iterations of the optimization procedure and hence the number of oracle calls to the base algorithm with the use of *column generation*. Column generation (Eisemann, 1957; Griva et al., 2008) is a classical optimization approach particularly suited for solving large linear programming problems with a special structure like the one studied here. While column generation tends to work well in practice, we are not aware of any convergence guarantees similar to those enjoyed by ExpGrad. We show how to combine the two approaches to obtain the best of both worlds. In practice, column generation decreases the number of ExpGrad iterations from around 100 to around 10 (see our experiments in Section 5) and the combined approach retains the convergence guarantees of ExpGrad.

While the column generation decreases the number of oracle calls, the goal of our second innovation is to decrease the cost of each oracle call. One simple approach, which we call *static sampling*, would be to subsample the training data uniformly at random and solve the constrained optimization problem for the smaller dataset; this improves the running time but leads to some loss in accuracy. We improve upon this naïve strategy by noting that datasets passed to the oracle are weighted, with a different weighting in each iteration. By sampling the data adaptively, according to the weights, we sacrifice less accuracy than we would if we used static sampling. For instance, if the dataset weights generated in a given iteration of ExpGrad put 90% of probability mass on 10% of examples, then picking the 10% examples with the largest weights results in a much smaller loss in accuracy than picking an arbitrary 10% of examples. In our experiments, we show that our adaptive sampling approach outperforms static sampling, and we do not see major losses in accuracy, even when the sample size is as small as 10% of the original training set (see Section 5).

Our experiments evaluate the efficacy of ExpGrad^{++} and compare it both with ExpGrad and other baselines. We show that our two innovations (column generation and adaptive sampling) speed up ExpGrad by an order of magnitude. The resulting approach still enjoys the flexibility of ExpGrad, while substantially improving its scalability.

2 Related work

We conceptualize fairness through the lens of fairness harms (Crawford, 2017; Wallach & Dudík, 2021; Shelby et al., 2023), by which we mean negative impacts on groups of people, such as those defined in terms of race, gender, age, or disability status. In fairness literature, this is also referred to as group fairness (Dwork et al., 2012). There are various alternative frameworks of fairness of AI systems, which we do not pursue here, such as individual fairness (Dwork et al., 2012) and fairness based on causal reasoning (Loftus et al., 2018).

Many algorithmic mitigation approaches have been proposed in the literature (see, e.g., the surveys of Mehrabi et al., 2021; Caton & Haas, 2023; Pessach & Shmueli, 2023). These can be broadly divided into three categories according to when they are applied relative to model training (Mehrabi et al., 2021; Caton & Haas, 2023; Barocas et al., 2019; Islam et al., 2022; Pessach & Shmueli, 2023). *Pre-processing* techniques seek to improve fairness by performing changes in the training data before passing it to an ML algorithm (Calmon et al., 2017; Feldman et al., 2015; Kamiran & Calders, 2012). *In-training* (or *in-processing*) techniques modify the ML algorithms to account for fairness during the training time (Zafar et al., 2017a;b; Woodworth et al., 2017; Kamishima et al., 2011; Zhang et al., 2018). *Post-processing* techniques analyze and modify the outputs of an already trained model (Hardt et al., 2016).

Pre-processing techniques are agnostic to the choice of the model family and typically also to the choice of fairness constraints. In practice this may lead to both suboptimal accuracy and suboptimal fairness (see our experiments as well as those of Agarwal et al., 2018). Post-processing techniques are designed to optimize tradeoff between fairness and acuracy and are also flexible in the choice of the model family, but they require access to the sensitive attribute at the deployment time. This may be inappropriate to use, or in some domains may be prohibited by law.[2] In-training techniques are capable of achieving optimal fairness–accuracy tradeoffs and do not require access to the sensitive attribute at the deployment time. However, many in-training techniques only incorporate one specific type of fairness constraint in one specific training algorithm (and one specific model family). This limits their usability in real-world scenarios where assessing the dataset's fairness using various metrics or testing different machine learning models is necessary. There are some exceptions to this. One of them is the reduction approach (Agarwal et al., 2018; 2019), which we study here. Another exception are direct optimization approaches, which assume the ability to model certain conditional densities (Menon & Williamson, 2018; Celis et al., 2019) and based on these directly optimize the accuracy subject to constraints. However, modeling such conditional densities is statistically feasible only when the data dimensionality is small, limiting the applicability of these techniques.

3 Preliminaries

3.1 Problem definition

We consider binary classification tasks, where the input is a dataset consisting of labeled examples $(X_1, A_1, Y_1), \ldots, (X_n, A_n, Y_n)$, where $X_i \in \mathbb{R}^d$ is a feature vector describing the i-th example, $A_i \in \mathcal{A}$ is a categorical sensitive attribute, and $Y_i \in \{0, 1\}$ is a binary label.

Consider the example from Section 1, in which the goal is to train an AI system that refers patients into a post-discharge care program based on the risk of a 30-day readmission. In this case, X_i contains clinical information about a patient; A_i is a categorical variable encoding patient's race and ethnicity; and Y_i indicates whether the patient was readmitted within 30 days of discharge.

[2]For example, under the U.S. Equal Credit Opportunity Act (15 U.S.C. 1691 et seq.), which regulates lending, it is possible to use the applicant's age (in a narrow sense described in Regulation B, 12 C.F.R. part 1026), but the use of other sensitive attributes like race or gender is prohibited.

The goal of a classification algorithm is to produce a classifier $h : \mathbb{R}^d \to \{0, 1\}$ that accurately predicts label Y on a new example represented by a feature vector $X \in \mathbb{R}^d$. We assume that h is chosen from some family $\mathcal{H}$ (like linear classifiers, decision trees, or neural nets). For example, logistic regression considers linear classifiers of the form $h_\beta(X) = 1\{\beta^T X \geq 0\}$ where $\beta \in \mathbb{R}^d$ and $1\{\cdot\}$ is an indicator function.

The sensitive attribute A is only required during training, but not during inference. However, since we make no assumptions about the relationship between X and A, it is possible for X to contain some information about A. In this way, our task definition encompasses both the settings when the sensitive attribute is available at inference time (in that case it is included as part of X) as well as the settings when the sensitive attribute is not available (in that case it is not included as part of X).

Standard classification algorithms seek $h \in \mathcal{H}$ that minimizes training classification error $err(h)$:[3]

$$\min_{h \in \mathcal{H}} err(h), \quad \text{where} \quad err(h) = \frac{1}{n} \sum_{i=1}^{n} 1\{h(X_i) \neq Y_i\}. \tag{1}$$

Algorithmic approaches to unfairness mitigation instead optimize training classification error under a fairness constraint, typically specified using the sensitive attribute A.

There are many notions of (un)fairness, appropriate in different applications (for instance, Islam et al., 2022, list 34 fairness notions collected from a review of the literature). In this paper, for simplicity, we focus on two standard quantitative definitions of fairness, but our technique encompasses many other notions (see Agarwal et al., 2018, for further details). Specifically, we consider *demographic parity* and *equalized odds* (Hardt et al., 2016; Barocas et al., 2019):

Definition 1 (Demographic parity: DP)**.** We say that a classifier h satisfies *demographic parity* with respect to a distribution over triples (X, A, Y) if its decision is statistically independent of A, that is, if $\mathbb{E}[h(X) \,|\, A = a] = \mathbb{E}[h(X)]$ for all $a \in \mathcal{A}$.

Definition 2 (Equalized odds: EO)**.** We say that a classifier h satisfies *equalized odds* with respect to a distribution over triples (X, A, Y) if its decision is statistically independent of A, conditional on Y, that is, if $\mathbb{E}[h(X) \mid A = a, Y = y] = \mathbb{E}[h(X) \mid Y = y]$ for all $a \in \mathcal{A}$ and $y \in \{0, 1\}$.

The degree to which a classifier h satisfies demographic parity on a data set $(X_1, A_1, Y_1), \ldots, (X_n, A_n, Y_n)$ can be quantified using the quantity

$$\begin{aligned} \Delta_{\mathrm{DP}}(h) &= \max_{a \in \mathcal{A}} \left| \hat{\mathbb{E}}[h(X) \mid A = a] - \hat{\mathbb{E}}[h(X)] \right| \\ &= \max_{a \in \mathcal{A}} \left| \frac{1}{n_a} \sum_{i:\, A_i = a} h(X_i) - \frac{1}{n} \sum_{i=1}^{n} h(X_i) \right|, \end{aligned} \tag{2}$$

where n_a is the number of examples with $A_i = a$. Compared with Definition 1, in Eq. (2) we replace true expectations $\mathbb{E}[\cdot]$ with empirical averages $\hat{\mathbb{E}}[\cdot]$, evaluated on the training data. The value of Δ_{DP} is equal to the largest deviation of $\hat{\mathbb{E}}[h(X) \mid A = a]$ from $\hat{\mathbb{E}}[h(X)]$, across all a. Definition 1 is satisfied (on the empirical distribution) exactly when the deviations across all a are equal to zero. Otherwise, Δ_{DP} quantifies an (additive) violation of fairness constraints. We refer to Δ_{DP} as the *DP difference*.

For equalized odds, we can quantify the violation of fairness constraints using an analogous quantity, called the *EO difference*:

$$\begin{aligned} \Delta_{\mathrm{EO}}(h) &= \max_{a \in \mathcal{A}, y \in \{0,1\}} \left| \hat{\mathbb{E}}[h(X) \mid A = a, Y = y] - \hat{\mathbb{E}}[h(X) \mid Y = y] \right| \\ &= \max_{a \in \mathcal{A}, y \in \{0,1\}} \left| \frac{1}{n_{a,y}} \sum_{i:\, A_i = a, Y_i = y} h(X_i) - \frac{1}{n_y} \sum_{i:\, Y_i = y} h(X_i) \right|, \end{aligned} \tag{3}$$

[3]Standard algorithms also include various mechanisms to control overfitting, such as regularization or early stopping. Similarly to Agarwal et al. (2018), we model regularization (and other similar mechanisms) by assuming that the family $\mathcal{H}$ has been appropriately restricted. For example, for regularized logistic regression, $\mathcal{H} = \{h_\beta : \beta \in \mathbb{R}^d, \|\beta\| \leq C\}$, where the value of C controls overfitting.

where n_y and $n_{a,y}$ refer to the number of examples with $Y_i = y$ and $A_i = a$, $Y_i = y$, respectively.

Continuing with the post-discharge care example, demographic parity states that the patients from all race/ethnicity groups should be referred to the post-discharge care program at equal rates. The fairness constraint $\Delta_{\mathrm{DP}}(h) \leq \epsilon$ states that the referral rate of every race/ethnicity group is only allowed to differ from the overall referral rate by at most ϵ.

The equalized odds condition states that false positive rates and false negative rates for all the race/ethnicity groups should be the same. The fairness constraints $\Delta_{\mathrm{EO}}(h) \leq \epsilon$ states that false positive rates and false negative rates of every race/ethnicity group are allowed to differ from the overall false positive rates and overall false negative rates, respectively, by at most ϵ.

3.2 Reduction approach

Reduction approach to fair classification (Agarwal et al., 2018) seeks to solve the problems of the form:

$$\min_{h \in \mathcal{H}} err(h) \quad \text{such that} \quad \Delta(h) \leq \epsilon, \tag{4}$$

where Δ formalizes fairness constraints and $\epsilon \geq 0$ is the degree to which we allow the fairness constraint to be violated.

Reduction approach works with a broad family of fairness constraints including Δ_{DP}, Δ_{EO} and many others (see Agarwal et al., 2018, for details). It also works with any family of ML models; it only requires the ability to solve weighted (but unconstrained) classification problems, meaning problems that minimize weighted classification error

$$\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} w_i 1\{h(X_i) \neq Y_i\} \tag{5}$$

for any set of weights $w_i \geq 0$. Eq. (5) can be solved by standard fairness-unaware classification algorithms, like those that fit logistic regression models, decision trees, or neural nets. The algorithms that solve Eq. (5) are viewed as *oracles* from the perspective of the reduction, and in practice they are implemented by library calls.

In order to leverage the strength of an oracle, we cast the constrained optimization problem in Eq. (4) as a linear optimization. For a classifier $h \in \mathcal{H}$, let $\mathbf{v}_h \in \mathbb{R}^n$ denote the vector with entries $v_{h,i} = h(X_i)$ for $i = 1, \ldots, n$. Then Eq. (4) can be rewritten as

$$\min_{h \in \mathcal{H}} \mathbf{c}^T \mathbf{v}_h + c_0 \quad \text{such that} \quad \mathbf{A}\mathbf{v}_h \leq \mathbf{b}, \tag{6}$$

where the vector $\mathbf{c} \in \mathbb{R}^n$ and the scalar $c_0 \in \mathbb{R}$ are determined by the choice of an error metric, in our case $err(h)$; the matrix $\mathbf{A} \in \mathbb{R}^{k \times n}$ and vector $\mathbf{b} \in \mathbb{R}^k$ are based on the choice of the fairness metric Δ and bound ϵ from Eq. (4). The dimension k is determined by the cardinality of $\mathcal{A}$ and the choice of the fairness metric Δ.

For example, using Eq. (2), the constraint $\Delta_{\mathrm{DP}}(h) \leq \epsilon$ can be written as $2|\mathcal{A}|$ constraints of the form

$$\begin{aligned} \hat{\mathbb{E}}[h(X) \mid A = a] - \hat{\mathbb{E}}[h(X)] &\leq \epsilon \\ -\hat{\mathbb{E}}[h(X) \mid A = a] + \hat{\mathbb{E}}[h(X)] &\leq \epsilon \end{aligned}$$

for all $a \in \mathcal{A}$, which can be expressed using a suitable matrix $\mathbf{A}$, with the vector $\mathbf{b}$ having all entries equal to ϵ. The constraint $\Delta_{\mathrm{EO}}(h) \leq \epsilon$ can be similarly expressed using $k = 4|\mathcal{A}|$ constraints (two constraints for each combination of $a \in \mathcal{A}$, $y \in \{0, 1\}$). (See Agarwal et al. (2018) for full derivation of $\mathbf{A}$ for DP, EO, as well as for more general fairness constraints.) We view k as a problem-specific constant. In much prior work, the sensitive attribute is binary ($|\mathcal{A}| = 2$), yielding $k = 4$ for DP and $k = 8$ for EO.

Instead of working with classifiers $h \in \mathcal{H}$, reduction approach considers a larger set consisting of randomized classifiers that randomize over a finite subset of $\mathcal{H}$. We write $\mathcal{P}(\mathcal{H})$ for the set of such randomized classifiers and identify them with the corresponding probability distributions over $\mathcal{H}$. A randomized classifier $p \in \mathcal{P}(\mathcal{H})$ makes a prediction on an example X by first sampling $h \sim p$ and then predicting $h(X)$. For example, if p

Algorithm 1 ExpGrad

Input: Lagrangian specified by $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{k \times n}$, $\mathbf{b} \in \mathbb{R}^k$;
bound B, learning rate η, convergence threshold ν, maximum iterations T

Set $\boldsymbol{\theta}_1 = \mathbf{0} \in \mathbb{R}^k$
for $t = 1, 2, \ldots, T$ **do**
  Set $\lambda_{t,j} = B \frac{\exp\{\theta_{t,j}\}}{1+\sum_{j'=1}^{k} \exp\{\theta_{t,j'}\}}$ for $j = 1, \ldots, k$
  $h_t \leftarrow \textsc{Best}_h(\boldsymbol{\lambda}_t)$, and let $\mathbf{v}_t = \mathbf{v}_{h_t}$
  $\mathbf{v}_{\text{EG}} \leftarrow \frac{1}{t} \sum_{t'=1}^{t} \mathbf{v}_{t'}$, $\boldsymbol{\lambda}_{\text{EG}} \leftarrow \frac{1}{t} \sum_{t'=1}^{t} \boldsymbol{\lambda}_{t'}$
  $\nu_{\text{EG}} \leftarrow \textsc{EvaluateDualityGap}(\mathbf{v}_{\text{EG}}, \boldsymbol{\lambda}_{\text{EG}})$
  if $\nu_{\text{EG}} \leq \nu$ or $t = T$ **then**
    Return p that randomizes uniformly over $h_1, \ldots, h_t$
  Set $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta\,(\mathbf{A}\mathbf{v}_t - \mathbf{b})$

function $\textsc{EvaluateDualityGap}(\mathbf{v}, \boldsymbol{\lambda})$
  $\overline{L} \leftarrow L(\mathbf{v}, \boldsymbol{\lambda}^*)$ where $\boldsymbol{\lambda}^* = \arg\max_{\boldsymbol{\lambda}' \in \mathbb{R}^k_+, \|\boldsymbol{\lambda}'\|_1 \leq B} L(\mathbf{v}, \boldsymbol{\lambda}')$
  $\underline{L} \leftarrow L(\mathbf{v}_{h^*}, \boldsymbol{\lambda})$ where $h^* = \textsc{Best}_h(\boldsymbol{\lambda})$
  Return $\max\{L(\mathbf{v}, \boldsymbol{\lambda}) - \underline{L},\ \overline{L} - L(\mathbf{v}, \boldsymbol{\lambda})\}$

function $\textsc{Best}_h(\boldsymbol{\lambda})$ // *returns* $\arg\min_{h \in \mathcal{H}} L(\mathbf{v}_h, \boldsymbol{\lambda})$
  Let $\mathbf{w} = \mathbf{c} + \mathbf{A}^T \boldsymbol{\lambda}$
  Find $h^* = \arg\min_{h \in \mathcal{H}} \mathbf{w}^T \mathbf{v}_h$
    by calling a standard classification algorithm for $\mathcal{H}$
    on the data set reweighted according to $\mathbf{w}$
  Return h^*

puts 0.5 probability mass on h_1 and 0.5 probability mass on h_2, then the randomized classifier specified by p predicts 1 on points X where $h_1(X) = h_2(X) = 1$, predicts 0 on points X where $h_1(X) = h_2(X) = 0$, and flips a coin on points X where $h_1(X) \neq h_2(X)$.

The linear optimization from Eq. (6) can be generalized to randomized classifiers. For any $p \in \mathcal{P}(\mathcal{H})$, we set $\mathbf{v}_p = \sum_{h \in \mathcal{H}} p(h) \mathbf{v}_h$, and the resulting optimization problem is

$$\min_{p \in \mathcal{P}(\mathcal{H})} \mathbf{c}^T \mathbf{v}_p \quad \text{such that} \quad \mathbf{A}\mathbf{v}_p - \mathbf{b} \leq 0. \tag{7}$$

This constrained optimization problem can be algorithmically solved by finding a solution to the min-max problem

$$\min_{p \in \mathcal{P}(\mathcal{H})} \max_{\boldsymbol{\lambda} \in \mathbb{R}^k_+, \|\boldsymbol{\lambda}\|_1 \leq B} L(\mathbf{v}_p, \boldsymbol{\lambda}) \tag{8}$$

where $L(\mathbf{v}, \boldsymbol{\lambda})$ is the Lagrangian

$$L(\mathbf{v}, \boldsymbol{\lambda}) = \mathbf{c}^T \mathbf{v} + \boldsymbol{\lambda}^T (\mathbf{A}\mathbf{v} - \mathbf{b}), \tag{9}$$

and $\boldsymbol{\lambda} \in \mathbb{R}^k_+$ is the vector of non-negative Lagrange multipliers.

Conceptually, the Lagrangian *scalarizes* the original constrained optimization problem in Eq. (7) by summing up the objective and individual constraint violations, multiplied by the Lagrange multipliers. The Lagrange multipliers λ_j, for $j = 1, \ldots, k$, specify the importance of not violating each of the constraints. The possibility of using sufficiently large λ_j in the inner maximization forces the outer minimization to choose the point p that (approximately) satisfies the constraints. It can be shown formally that with an appropriate choice of B, the solution of Eq. (8) approximately solves Eq. (7) (see Agarwal et al., 2018).

Reduction approach solves the min-max problem from Eq. (8) by an iterative algorithmic scheme developed by Freund & Schapire (1996) for finding an equilibrium in a zero-sum game. The min-max problem in

Eq. (8) can be viewed as a game between a $\boldsymbol{\lambda}$-player seeking to maximize the Lagrangian and $\mathbf{v}$-player seeking to minimize the Lagrangian. Freund & Schapire (1996) propose an iterative protocol where in each round t, the $\boldsymbol{\lambda}$-player plays the action $\boldsymbol{\lambda}_t$ according to a suitable online learning algorithm (in our case the exponentiated gradient algorithm of Kivinen & Warmuth, 1997) and $\mathbf{v}$-player plays the best response $\mathbf{v}_t$ to the other player's action $\boldsymbol{\lambda}_t$. Freund & Schapire (1996) show that the averages of the played actions $\boldsymbol{\lambda}_t$ and $\mathbf{v}_t$ converge to an equilibrium of the game, which coincides with the solution of the min-max problem.

This scheme is implemented in Algorithm 1. The vector $\boldsymbol{\theta}_t \in \mathbb{R}^k$ is used to obtain $\boldsymbol{\lambda}_t$ (the action of the $\boldsymbol{\lambda}$-player) via a soft-max transformation that guarantees that the components of $\boldsymbol{\lambda}_t$ are non-negative and sum to at most B (step 3). The best response of the $\mathbf{v}$-player is obtained in step 4 by calling the function $\textsc{Best}_h(\boldsymbol{\lambda}_t)$, which is implemented by calling the oracle for the family $\mathcal{H}$. In steps 5 and 6, we consider the current average play of the $\boldsymbol{\lambda}$- and $\mathbf{v}$-player and check how close these averages are to the equilibrium of the game by evaluating how much each player can unilaterally improve their objective (see function EvaluateDualityGap). If the possible improvement is below a convergence threshold ν, or if we have reached the maximum number of iterations, we return the current average play. Otherwise, we update $\boldsymbol{\theta}_t$, following the exponentiated-gradient update rule. Conceptually, we form a vector of constraint violations $\mathbf{A}\mathbf{v}_t - \mathbf{b}$ and increase the values of the components of $\boldsymbol{\theta}_t$ (and thus also of $\boldsymbol{\lambda}_t$) according to how much violation occurs. This means that in the next iteration of the protocol, the constraints that were more violated receive more importance in the Lagrangian.

Agarwal et al. (2018) show that for typical families $\mathcal{H}$ (like linear classifiers, neural nets, and boosted trees), given a dataset of size n, we should set $B \propto \sqrt{n}$, $\eta \propto 1/n$, $\nu \propto 1/\sqrt{n}$, and $T \propto n^2$ to guarantee that the solution returned by the algorithm satisfies the fairness constraints and matches the accuracy of the solution of Eq. (8) (up to an error of at most $O(1/\sqrt{n})$, which is on the same scale as the difference between the training error and error with respect to the true distribution). This means that for a dataset of size 1 000, the theory requires around 1 million iterations! The most costly operation in each iteration is the call to $\textsc{Best}_h$, which involves calling an oracle, on a weighted dataset of size n. So theory would yield the running time that is 1 million times slower than the running time of a fairness-unaware approach. In practice, on the datasets of size up to 50 000, we find that around 100 iterations suffice to reach the termination condition (see our experiments in Section 5). However, a 100-fold slowdown is still prohibitive, and presents the main obstacle for applying reduction approach with larger datasets.

4 Our approach: ExpGrad^{++}

We introduce two innovations to speed up Algorithm 1. First, we interleave exponentiated gradient with column generation (see, e.g., Griva et al., 2008, Section 7.3) to decrease the number of iterations to around 5–10 (instead of 100). Second, we use sampling to generate smaller training datasets when calling the oracle in $\textsc{Best}_h$. The resulting approach is presented in Algorithm 2, with new and revised steps marked with an asterisk.

4.1 Column generation

Assume that the family $\mathcal{H}$ is of finite cardinality, $|\mathcal{H}| = N$; this is without loss of generality, because the optimization in Eqs. (7) and (8) only considers predictions on a fixed dataset of size n, and hence we can assume $|\mathcal{H}| \leq 2^n$ (from the perspective of optimization).

Let $\mathbf{V} \in \mathbb{R}^{n \times N}$ be the matrix with columns corresponding to vectors $\mathbf{v}_h$ across $h \in \mathcal{H}$. Then a probability distribution $p \in \mathcal{P}(\mathcal{H})$ can be viewed as a vector in $\mathbb{R}_+^N$ with the components $p(h)$ summing to one, $\sum_{h \in \mathcal{H}} p(h) = 1$, and $\mathbf{v}_p$ is obtained by matrix-vector multiplication as $\mathbf{v}_p = \mathbf{V}p$. Thus, Eq. (8) can be written as a linear programming (LP) problem:

$$\max_{\boldsymbol{\lambda} \in \mathbb{R}_+^k, \|\boldsymbol{\lambda}\|_1 \leq B} \quad \min_{p \in \mathbb{R}_+^N, \sum_{h \in \mathcal{H}} p(h) = 1} \Big[\mathbf{c}^T \mathbf{V} p + \boldsymbol{\lambda}^T (\mathbf{A}\mathbf{V}p - \mathbf{b}) \Big]. \tag{10}$$

It is intractable to solve this problem with standard LP solvers (since N is exponential in n). To circumvent the dimensionality of N, the column generation (CG) approach considers a small subset of base classifiers

Algorithm 2 ExpGrad^{++}

Input: Lagrangian specified by $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{k \times n}$, $\mathbf{b} \in \mathbb{R}^k$;
bound B, learning rate η, convergence threshold ν, maximum iterations T, sampling ratio ρ

Set $\boldsymbol{\theta}_1 = \mathbf{0} \in \mathbb{R}^k$
*$cache \leftarrow \{h_{\text{init}}\}$ where $h_{\text{init}} = \arg\min_{h \in \mathcal{H}} err(h)$
for $t = 1, 2, \ldots, T$ **do**
  Set $\lambda_{t,j} = B \frac{\exp\{\theta_{t,j}\}}{1 + \sum_{j'=1}^{k} \exp\{\theta_{t,j'}\}}$ for $j = 1, \ldots, k$
  $h_t \leftarrow \text{Best}_h(\boldsymbol{\lambda}_t)$, and let $\mathbf{v}_t = \mathbf{v}_{h_t}$
  $\mathbf{v}_{\text{EG}} \leftarrow \frac{1}{t} \sum_{t'=1}^{t} \mathbf{v}_{t'}$, $\boldsymbol{\lambda}_{\text{EG}} \leftarrow \frac{1}{t} \sum_{t'=1}^{t} \boldsymbol{\lambda}_{t'}$
  $\nu_{\text{EG}} \leftarrow \text{EvaluateDualityGap}(\mathbf{v}_{\text{EG}}, \boldsymbol{\lambda}_{\text{EG}})$ *// see Algorithm 1*
  if $\nu_{\text{EG}} \leq \nu$ **then**
    Return p that randomizes uniformly over $h_1, \ldots, h_t$
  *Let $\tilde{\mathbf{V}}$ be the matrix with columns $\mathbf{v}_h$ across $h \in cache$
  *$(\boldsymbol{\lambda}_{\text{CG}}, p_{\text{CG}}) \leftarrow$ solve Eq. (11) with $\tilde{\mathcal{H}} = cache$
  *$\nu_{\text{CG}} \leftarrow \text{EvaluateDualityGap}(\tilde{\mathbf{V}} p_{\text{CG}}, \boldsymbol{\lambda}_{\text{CG}})$ *// see Algorithm 1*
  *__if__ $\nu_{\text{CG}} \leq \nu$ or $t = T$ **then**
  *Return p that randomizes over *cache* according to p_{CG}
  Set $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta \left(\mathbf{A}\mathbf{v}_t - \mathbf{b}\right)$

function $\text{Best}_h(\boldsymbol{\lambda})$ *// approximates* $\arg\min_{h \in \mathcal{H}} L(\mathbf{v}_h, \boldsymbol{\lambda})$
  Let $\mathbf{w} = \mathbf{c} + \mathbf{A}^T \boldsymbol{\lambda}$
  *Set $q_i = \frac{|w_i|}{\sum_{i=1}^{n} |w_i|}$ for $i = 1, \ldots, n$
  *$D \leftarrow \{\}$
  *__repeat__ $\lceil \rho n \rceil$ times
  *Sample $i \sim \mathbf{q}$
  *Add to D an example (X, Y) with $X = X_i$, $Y = 1\{w_i < 0\}$
  *Find $h^* = \arg\min_{h \in \mathcal{H}} \sum_{(X,Y) \in D} 1\{h(X) \neq Y\}$
    by calling a standard classification algorithm for $\mathcal{H}$
  *Add h^* to *cache* and return h^*

$\tilde{\mathcal{H}} \subseteq \mathcal{H}$, with $m = |\tilde{\mathcal{H}}|$, and replaces the large matrix $\mathbf{V} \in \mathbb{R}^{n \times N}$ by a smaller matrix $\tilde{\mathbf{V}} \in R^{n \times m}$ that only contains columns corresponding to the classifiers $h \in \tilde{\mathcal{H}}$. Instead of directly solving Eq. (10), CG solves a smaller restricted problem

$$\max_{\boldsymbol{\lambda} \in \mathbb{R}_+^k, \|\boldsymbol{\lambda}\|_1 \leq B} \quad \min_{p \in \mathbb{R}_+^m, \sum_{h \in \tilde{\mathcal{H}}} p(h) = 1} \left[\mathbf{c}^T \tilde{\mathbf{V}} p + \boldsymbol{\lambda}^T (\mathbf{A} \tilde{\mathbf{V}} p - \mathbf{b}) \right]. \tag{11}$$

The CG approach starts with $m = 1$ and $\tilde{\mathcal{H}}$ containing an arbitrary base classifier (for example, the classifier obtained by optimizing accuracy without any fairness constraints), and then repeatedly solves the restricted problem in Eq. (11). The solution to the restricted problem is checked for optimality using the function EvaluateDualityGap from Algorithm 1. If the duality gap is non-zero, it means that h^* obtained in the call to EvaluateDualityGap can be used to reduce the objective in Eq. (11), and so h^* is added to $\tilde{\mathcal{H}}$, its corresponding vector $\mathbf{v}_{h^*}$ is added to $\tilde{\mathbf{V}}$, and the LP in Eq. (11) is solved again. This is repeated until convergence.

In practice CG can converge in a very small number of iterations, but we are not aware of any non-trivial theoretical upper bounds. Therefore, we interleave column generation (CG) with exponentiated gradient (EG), which allows us to benefit from strong practical performance of CG while retaining the worst-case guarantees of EG.

To combine the two approaches, we introduce a data structure that contains all of the classifiers returned by $\textsc{Best}_h$ so far, which we refer to as *cache*. This *cache* plays a role of $\tilde{\mathcal{H}}$ in CG. The modified algorithm (Algorithm 2) initializes *cache* with the classifier returned by a standard (fairness-unaware) classification algorithm for $\mathcal{H}$ (step 2). In each iteration t, the algorithm first performs the EG update (steps 4–9), and if the convergence condition in step 8 is not satisfied, it runs a single iteration of CG with $\tilde{\mathcal{H}} = \mathit{cache}$ (steps 10–14). If the CG convergence condition is not satisfied (in step 13) then it finishes the EG update (in step 15) and proceeds to next iteration. Since the EG iterates are unaffected, we retain the original convergence guarantee. However, by introducing CG steps, it is possible to terminate much earlier if the duality gap of the CG iterates falls below ν. In practice, we observe that this termination condition is reached in around 10 iterations instead of 100 iterations that were required by EG to reach a similar quality of the solution (see Section 5).

4.2 Subsampling

While CG decreases the number of iterations and therefore the number of oracle calls, the goal of our second innovation is to decrease the cost of each oracle call. We do this by subsampling the data before passing it to the oracle.

Subsampling is a general and widely used strategy that makes the training process faster and more manageable in terms of storage and memory resources, but it can lead to loss of accuracy in the obtained solution if the sample does not represent the original data sufficiently well. A simple strategy, which we call *static sampling*, is to subsample the training data uniformly at random and solve the constrained optimization problem for the smaller dataset. We expect this strategy to do well in approximating the overall training error, but it might severely impact the accuracy of fairness metrics, which are calculated on various subgroups of data, and whose sizes might become too small as a result of subsampling.

We improve upon this naive strategy by noting that datasets passed to the oracle are weighted, with a different weighting in each iteration determined by the current vector of Lagrange multipliers $\boldsymbol{\lambda}_t$ (see step 4 and the implementation of the function $\textsc{Best}_h$ in Algorithm 1). The components of the weight vector $\mathbf{w} \in \mathbb{R}^n$ specify how important each data point is. Operationally, this tends to upweight the groups for which the fairness constraint is most violated. This suggests a natural adaptive sampling strategy, which subsamples the original data according to $\mathbf{w}$. We implement such an adaptive sampling strategy in the function $\textsc{Best}_h$ in Algorithm 2. The size of the subsampled dataset is controlled by a sampling ratio $\rho \in (0, 1]$, with the subsampled dataset of size $\lceil \rho n \rceil$.

By sampling the data adaptively, according to the weight vector $\mathbf{w}$, we sacrifice less accuracy than we would if we used static sampling. For instance, if the weight vector $\mathbf{w}$ in a given iteration of $\textsc{ExpGrad}^{++}$ puts 90% of probability mass on 10% of examples, then picking the 10% examples with the largest weights results in a much smaller loss in accuracy than picking an arbitrary 10% of examples. In our experiments, we show that adaptive sampling approach outperforms static sampling, and we do not see major losses in accuracy even when the sample size is only 25% or even less of the original dataset size (see Section 5).

5 Experimental evaluation

We next evaluate efficacy of our speedups. We first compare overall running times and quality of the solutions produced by $\textsc{ExpGrad}^{++}$ with those of $\textsc{ExpGrad}$, across a range of datasets, fairness constraints, and learning oracles. We contextualize performance metrics by including additional baselines from fairness literature. We then dig into our two innovations separately, and conduct two ablation studies. In the first, we remove the sampling component of $\textsc{ExpGrad}^{++}$ and evaluate the impact of column generation on the number of oracle calls. In the second, we evaluate impact of sampling on the overall running time and solution quality.

Table 1: Datasets used in experiments.

Dataset	File size (MB)	#examples	#features	Sensitive attribute	
				name	#values
Adult	4.67	4.5×10^4	9	Sex	2
COMPAS	0.37	4.2×10^3	3	Race	2
German	0.05	1.0×10^3	9	Sex	2
ACSEmployment	319.47	3.2×10^6	99	RAC1P	9
ACSPublicCoverage	163.26	1.1×10^6	140	RAC1P	9

5.1 Experimental setup

Tasks and data. We consider binary classification problems under two kinds of fairness constraints: demographic parity and equalized odds (Dwork et al., 2012; Hardt et al., 2016).

We consider 5 tabular datasets, whose main characteristics are shown in Table 1. Three of them (*Adult*, *COMPAS*, and *German*) are smaller datasets included in many previous studies (Islam et al., 2022; Mehrabi et al., 2021; Pessach & Shmueli, 2023). Two larger datasets (*ACSEmployment* and *ACSPublicCoverage*s) have been proposed more recently (Ding et al., 2021) to enable larger-scale evaluations.

Adult (Becker & Kohavi, 1996) describes demographic and occupational attributes of several thousand individuals extracted from the 1994 US Census database. The task is to predict whether an individual has an income higher than 50K, with sex as the sensitive attribute.

COMPAS (Larson et al., 2016) contains arrest records, demographic information, and criminal history of defendants arrested in 2013–2014. The task is to predict reoffense within two years, with race as the sensitive attribute.

German (Hofmann, 1994) contains records of individuals applying for a loan. The task is to predict whether the default risk of an individual is high or low, using sex as the sensitive attribute.

ACSPublicCoverage and *ACSEmployment* (Ding et al., 2021) have been constructed from the US Census data collected within the American Community Survey. These datasets include information related to ancestry, citizenship, education, race, employment, language proficiency, income, disability, etc. The datasets differ in their prediction tasks: *ACSPublicCoverage* contains data for predicting whether an individual is covered by public health insurance, *ACSEmployment* contains data for predicting whether an individual is employed. We select race (RAC1P in the dataset) as the sensitive attribute. Unlike other datasets, in this case, the sensitive attribute is multi-valued.

Models. We run the reduction approach (ExpGrad and ExpGrad^{++}) with two kinds of oracles: *LogisticRegression* and *HistGradientBoostingClassifier* from the *scikit-learn* library. Their hyperparameters are tuned separately for each dataset, without fairness constraints, using 5-fold crossvalidation. In the body of the paper, we report results only for logistic regression, and leave the results for gradient-boosted decision trees to the appendix.

For ExpGrad, we use the implementation available in the *fairlearn* library version 0.9.0 (Weerts et al., 2023). For ExpGrad^{++}, we augment the fairlearn implementation. In most experiments, we use the default settings of the optimization hyperparameters of ExpGrad. However, in some experiments we vary the learning rate via the hyperparameter *eta0*, which specifies a multiplicative constant applied to the theoretical value of the learning rate. (The default value is *eta0=2.0*.)

In addition to ExpGrad and ExpGrad^{++}, we also consider an unmitigated approach (UNMITIGATED), which corresponds to running the base classification algorithm, and 6 additional methods from fairness literature discussed in Section 2:

- Two pre-processing approaches: CALMON (Calmon et al., 2017) and FELD (Feldman et al., 2015).

- Two in-training approaches: ZAFAR DI (Zafar et al., 2017b) for demographic parity, and ZAFAR EO (Zafar et al., 2017a) for equalized odds.
- Two post-processing approaches: HARDT (Hardt et al., 2016) and PLEISS (Pleiss et al., 2017).

Pre-processing and post-processing approaches are applicable to both logistic regression and boosted tree models, while ZAFAR DI/EO is only applicable to logistic regression. Moreover, ZAFAR EO, CALMON and PLEISS cannot be applied to the two larger datasets since ZAFAR EO and PLEISS do not support multi-valued sensitive attributes, and CALMON requires a problem-specific distortion function, which was unavailable for these datasets. Finally, note that HARDT and PLEISS require access to the sensitive attribute at prediction time (which might not be available or allowed in some contexts).

Evaluation methodology and metrics. All of our experiments were conducted on a machine equipped with Intel(R) Xeon(R) Platinum 8370C CPU @ 2.80GHz and 62.8 GB RAM, running on Ubuntu 20.04.5 LTS operating system. The accuracy is evaluated in terms of classification error. Fairness is evaluated using the DP difference Δ_{DP} and EO difference Δ_{EO}, which were introduced in Section 3 (Eqs. 2 and 3) as a way to quantify the degree of violation of fairness constraints.

For each experimental configuration, we evaluate the performance of each algorithm using stratified 3-fold cross validation, executed twice with different random seeds, resulting in six replications of each experiment. Stratification is performed by jointly considering the sensitive attribute and the label.

5.2 Overall performance comparison

In our first set of experiments we compare ExpGrad^{++} and ExpGrad in terms of running times and the quality of the solutions they return. In ExpGrad^{++}, we use the sampling ratio $\rho = 0.25$ with the datasets *ACSPublicCoverage* and *ACSEmployment*, and do not use subsampling on the three smaller datasets. For both reduction algorithms, we consider several values of the constraint violation bound ϵ (see Eq. 4). Specifically, we consider $\epsilon \in \{0.001, 0.005, 0.01, 0.02, 0.05, 0.10, 0.15\}$, and evaluate the performance for each.

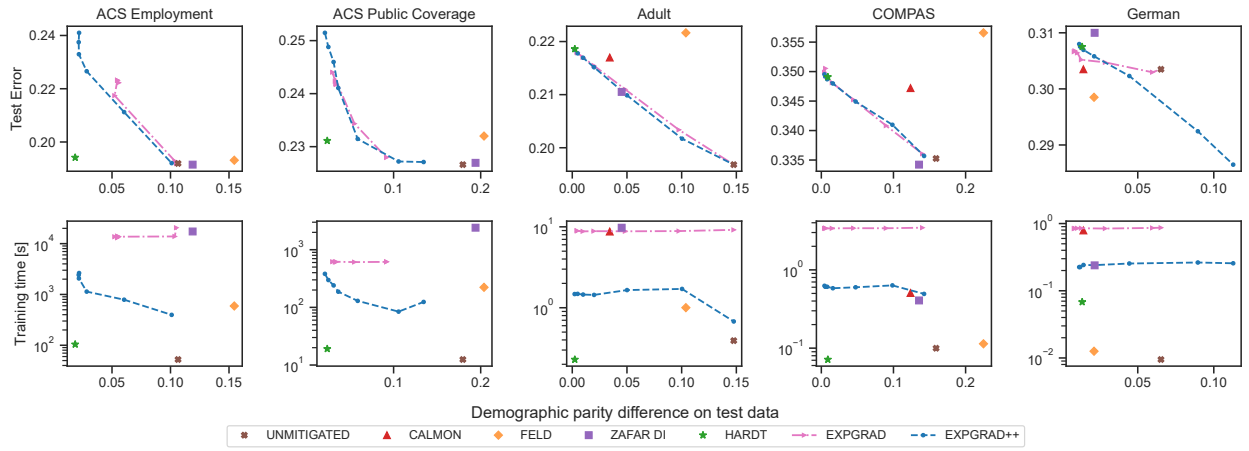
In Figure 1, we show the test error and training time as a function of test fairness violation, across 5 datasets and for two types of fairness constraints. ExpGrad^{++} and ExpGrad are plotted as curves obtained from runs for different values of ϵ, corresponding to different fairness–accuracy tradeoffs. Other methods optimize a fixed tradeoff and are plotted as points. For clarity, results are reported without error bars (plots with error bars are shown in Figure 4 in Appendix).

We first focus on democratic parity results in Figure 1a. In the top row, we plot test error as a function of test fairness violation. Points towards lower left represent better models. Comparing ExpGrad^{++} and ExpGrad, we see that ExpGrad^{++} can achieve the same fairness–accuracy tradeoffs as ExpGrad, but it can also reach additional points along the tradeoff curve. In particular, ExpGrad^{++} is able to achieve lower values of fairness constraint violation than ExpGrad, possibly because of better optimization. At the same time, as the second row shows, ExpGrad^{++} is substantially faster than ExpGrad (up to a factor of 10 on large datasets).

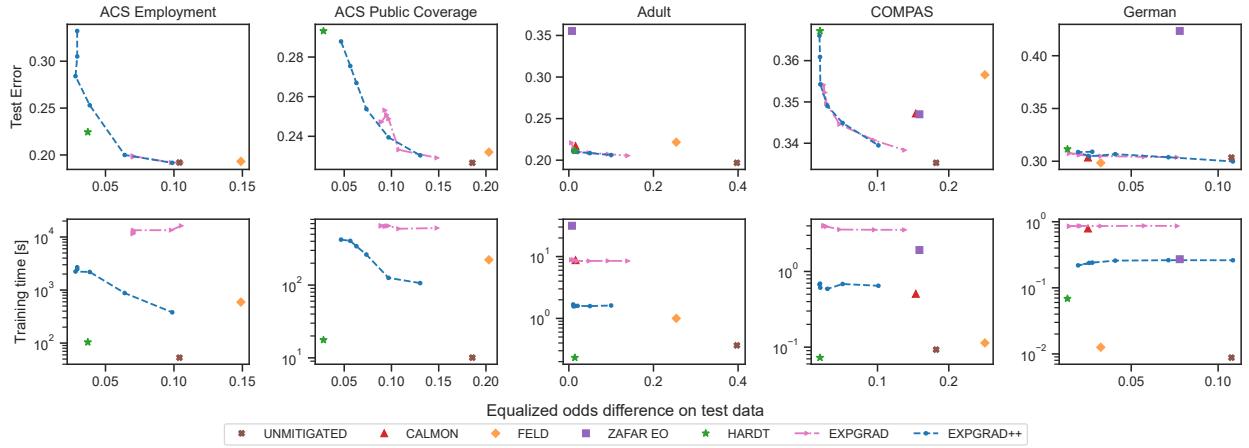
Considering again the top row of Figure 1a, and now comparing ExpGrad^{++} against other baselines, we see that on the three small datasets, ExpGrad^{++} successfully matches or improves upon fairness–accuracy tradeoffs achieved by other methods.[4] On the two larger datasets, the post-processing approach HARDT achieves a better fairness–accuracy tradeoff than ExpGrad^{++}. However, recall that HARDT requires access to the sensitive attribute at the inference time, which is not possible in many real-world scenarios. The second row of the figure shows that the unmitigated approach and post-processing are the fastest, whereas the running time of ExpGrad^{++} is similar or better than the running times of the pre-processing and in-training methods (with the exception of FELD on the two smallest datasets: COMPAS and German).

The evaluation of equalized odds in Figure 1b yields analogous conclusions. Altogether these experiments show that ExpGrad^{++} is able to speed up ExpGrad by an order of magnitude, without sacrificing the

[4] On *German*, it appears that at low constraint violations, CALMON and FELD achieve a lower test error than ExpGrad^{++}, but the differences are not statistically significant (see plots with error bars in Figure 4a in Appendix).



(a) Demographic Parity



(b) Equalized Odds

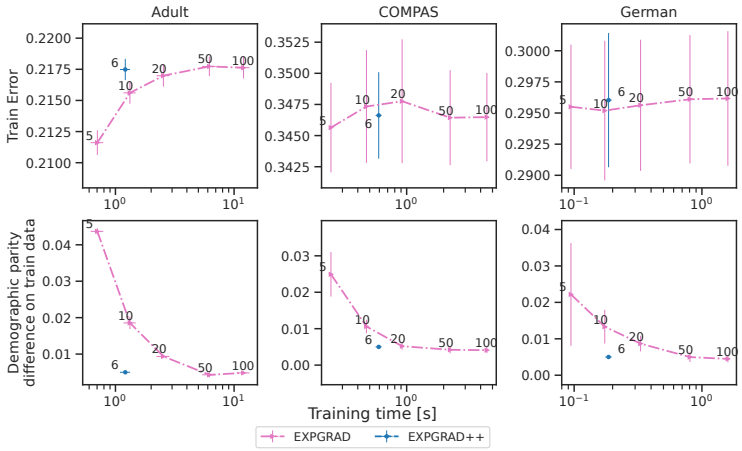
Figure 1: *Overall performance comparison (base learner: logistic regression).* Plotting test error and training time as a function of test fairness violation, averaged over 6 replicates. ExpGrad^{++} and ExpGrad shown as curves, because they allow tuning of fairness–accuracy tradeoff; other methods shown as points. Results closer to lower-left corner of subplots are more favorable. Plots show that ExpGrad^{++} achieves the same fairness–accuracy tradeoffs as ExpGrad, and matches or dominates the tradeoffs achieved by other methods except for HARDT, which however requires access to the sensitive attribute at inference time. ExpGrad^{++} successfully decreases the training runtime compared with ExpGrad and matches the runtime of other methods except HARDT and UNMITIGATED (and FELD on the two smallest datasets).

quality of the resulting solutions. HARDT remains the best choice in settings where sensitive attribute is available (and is allowed to be used) at test time.

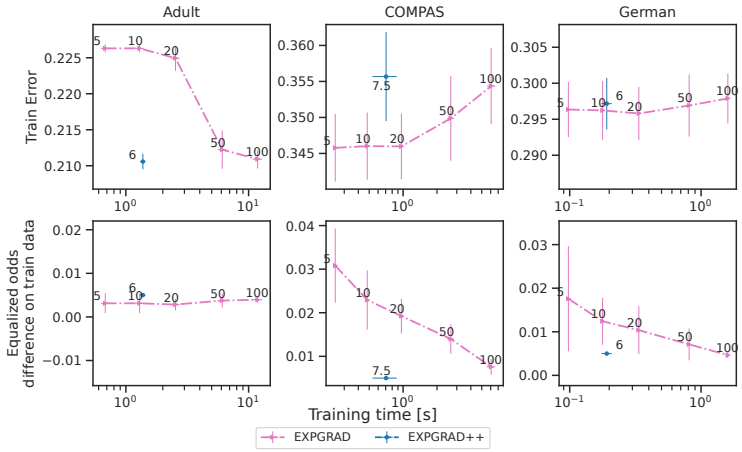
5.3 Efficacy of column generation

We next study the efficacy of column generation. We compare ExpGrad with ExpGrad^{++}, but to isolate the effect of column generation, we do not perform any subsampling in ExpGrad^{++}. Since ExpGrad is much slower than ExpGrad^{++} (as we saw in Section 5.2), we limit this ablation study to the three smaller datasets: *Adult*, *COMPAS*, and *German*.

In both ExpGrad and ExpGrad^{++}, we set the allowed constraint violation to $\epsilon = 0.005$. We run ExpGrad^{++} with the default learning rate and default termination condition. For ExpGrad, we consider



(a) Demographic Parity.



(b) Equalized odds.

Figure 2: *Efficacy of column generation (base learner: logistic regression).* Train error and fairness violation as a function of training time for 3 datasets, averaged over 6 replicates. For ExpGrad, showing performance at iterations ranging from 5 to 100 (indicated by numerical labels); for ExpGrad^{++}, showing performance at convergence (with a numerical label indicating the average number of iterations). ExpGrad^{++} requires fewer iterations and thus less time to reach a solution of the same quality as ExpGrad.

three different learning rates (specified in the *fairlearn* library via the hyperparameter `eta0` $\in \{0.5, 1.0, 2.0\}$), and for each, we take the best-performing solution among the three solutions (according to the duality gap). For ExpGrad, we only report the running time of the run with the best hyperparameter setting rather than the sum of all three runs—this gives an advantage to ExpGrad.

In Figure 2, we show how well the two algorithms optimize accuracy and fairness as a function of time. We focus on training metrics because these capture the progress of optimization. For ExpGrad^{++}, we plot the accuracy and fairness after reaching the termination condition based on the duality gap (since the algorithm always reaches the early termination condition), but for ExpGrad, we plot the performance at several different iterates, possibly corresponding to stopping before convergence condition is reached.

Figure 2 clearly shows that adding the column generation improves the efficiency of the original ExpGrad. The new algorithm, ExpGrad^{++}, reaches the solution of the same quality as eventually found by ExpGrad in substantially fewer iterations, and hence with a substantially lower running time. For instance, focusing on the *Adult* dataset in Figure 2a, note that ExpGrad starts with a solution that has a larger than desired

constraint violation, and as the constraint violation decreases, the training error slightly decreases. The final solution, reached after 50 iterations, has the same fairness–accuracy tradeoff as reached by ExpGrad^{++} after just 6 iterations.

5.4 Efficacy of sampling

Finally, we investigate how sampling impacts the running time and quality of solutions produced by ExpGrad^{++}. In all experiments, we set the allowed constraint violation to $\epsilon = 0.005$. We evaluate our adaptive sampling approach for the sampling ratios $\rho \in \{0.001, 0.004, 0.016, 0.063, 0.251\}$. As a baseline, we also consider static sampling, where the dataset is subsampled (uniformly at random) once at the beginning, and then ExpGrad^{++} is run on the subsampled dataset (without any further subsampling). In both cases, we run ExpGrad^{++} with column generation and a default setting of optimization hyperparameters. We also evaluate ExpGrad (that is, the exponentiated gradient algorithm without column generation) with both adaptive and static sampling, and show performance of unmitigated approach with static sampling.

In Figure 3 we show how the running time, test error, and test constraint violation of the evaluated algorithms vary as a function of sampling ratio.

Figure 3a shows results for demographic parity. The third row shows that ExpGrad^{++} with both adaptive and static sampling always achieves the desired level of fairness. In contrast, ExpGrad with static sampling (and for *ACS Employment* also with adaptive sampling) fails to achieve the desired level of fairness. This is because severe subsampling makes the underlying optimization problems harder, so the algorithm fails to find a feasible solution within the default number of iterations $T = 50$ (this does not affect ExpGrad^{++}, which converges in all cases). Unmitigated solution, as expected, violates fairness constraint in all cases.

Continuing with the second row of Figure 3a, we see that using adaptive sampling allows both ExpGrad^{++} and ExpGrad to achieve lower test errors compared with static sampling for the same sampling rates. In particular, note that adaptive sampling is able to achieve test error close to that of the entire dataset even at low sampling rates (0.01 for large datasets and 0.1 for small datasets).

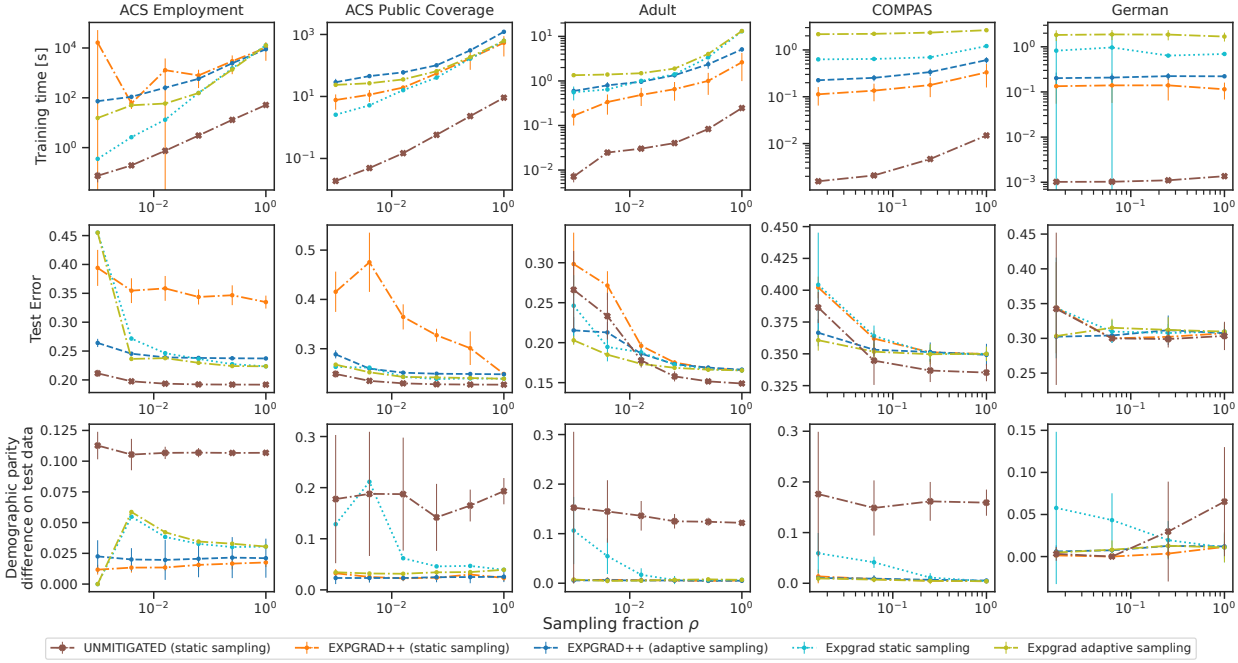
Now focusing on the adaptive sampling variants only, we see in the second row of Figure 3a that on the three smaller datasets, both ExpGrad^{++} and ExpGrad reach the same test error, but on the two larger datasets ExpGrad seems to be slightly better than ExpGrad^{++}. This however coincides with worse fairness due to the fact that ExpGrad has not fully converged (in particular, see Figure 7 in Appendix, showing that ExpGrad optimization does not achieve the desired training fairness bound).

Finally, the first row of Figure 3a shows that subsampling improves running times, especially for larger datasets. Adaptive sampling is generally slightly slower than static sampling, as expected, because of the overhead of resampling at each iteration. One exception to this is *ACS Employment*, where ExpGrad^{++} with adaptive sampling is faster than ExpGrad^{++} with static sampling at low sampling ratios. This is because static subsampling undersamples small groups which leads to harder optimization problems, requiring more iterations to solve.

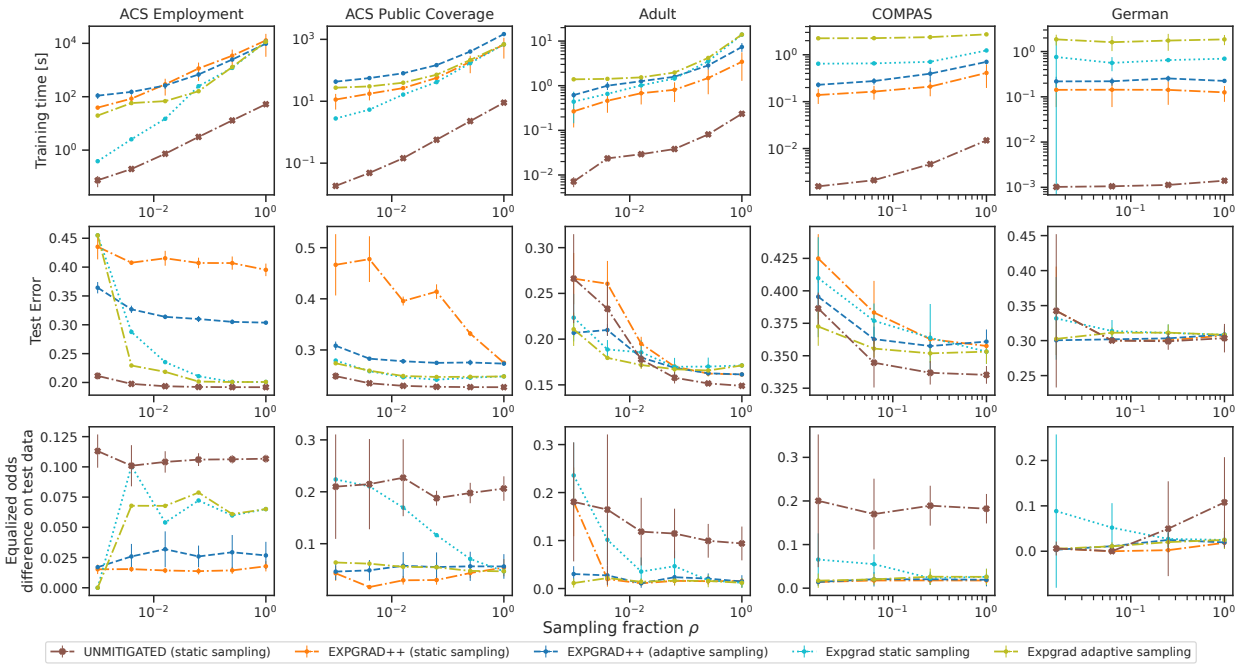
These general observations also carry over to the results for equalized odds in Figure 3b. Again, we see that adaptively selecting a small subsample of the dataset in ExpGrad^{++} (around 0.01 on larger datasets and 0.1 on smaller ones) yields substantial running time improvements without sacrificing the quality of the obtained solutions. The cases when ExpGrad with adaptive sampling achieves better test error than ExpGrad^{++} with adaptive sampling coincide with cases when ExpGrad has not achieve the desired training fairness (see Figure 8 in Appendix), underscoring the importance of including column generation in the optimization procedure.

6 Conclusion

In this work, we have introduced two speedups in the reduction approach to fair classification: column generation and adaptive sampling. In our experiments on both small and large datasets, we have shown that the resulting algorithm ExpGrad^{++} matches the quality of solutions of the standard reduction algorithm



(a) Demographic Parity.



(b) Equalized Odds.

Figure 3: *Efficacy of sampling (base learner: logistic regression).* Plotting train runtime, test error, and test fairness violation as a function of sampling ratio; comparing adaptive and static sampling in ExpGrad^{++}. Both adaptive and static sampling achieve low fairness violation, but adaptive sampling has lower test error. Ratios as low as 0.1 (or even less) yield improved runtime without sacrificing accuracy.

(ExpGrad), while substantially improving its runtime. As a result, reduction approach can be applied in a wider range of applications, including applications with larger datasets or more costly base algorithms.

References

Alekh Agarwal, Alina Beygelzimer, Miroslav Dudík, John Langford, and Hanna Wallach. A Reductions Approach to Fair Classification. In *International Conference on Machine Learning*, pp. 60–69, 2018.

Alekh Agarwal, Miroslav Dudík, and Zhiwei Steven Wu. Fair Regression: Quantitative Definitions and Reduction-Based Algorithms. In *International Conference on Machine Learning*, pp. 120–129, 2019.

Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias: There's software used across the country to predict future criminals. and it's biased against blacks. 2016. `https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing`.

Solon Barocas, Kate Crawford, Aaron Shapiro, and Hanna Wallach. The problem with bias: Allocative versus representational harms in machine learning. In *9th Annual conference of the special interest group for computing, information and society*, 2017.

Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and Machine Learning: Limitations and Opportunities*. fairmlbook.org, 2019. `http://www.fairmlbook.org`.

Barry Becker and Ronny Kohavi. Adult. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.

Rachel KE Bellamy, Kuntal Dey, Michael Hind, Samuel C Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, Aleksandra Mojsilović, et al. Ai fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias. *IBM Journal of Research and Development*, 63(4/5):4–1, 2019.

Ruha Benjamin. *Race After Technology: Abolitionist Tools for the New Jim Code*. Polity, 2019.

Meredith Broussard. *Artificial Unintelligence: How Computers Misunderstand the World*. MIT Press, 2018.

Joy Buolamwini and Timnit Gebru. Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. In *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, pp. 77–91, 2018.

Flávio P. Calmon, Dennis Wei, Bhanukiran Vinzamuri, Karthikeyan Natesan Ramamurthy, and Kush R. Varshney. Optimized pre-processing for discrimination prevention. In *NIPS*, pp. 3992–4001, 2017.

Simon Caton and Christian Haas. Fairness in machine learning: A survey. *ACM Comput. Surv.*, 2023.

L. Elisa Celis, Lingxiao Huang, Vijay Keswani, and Nisheeth K. Vishnoi. Classification with fairness constraints: A meta-algorithm with provable guarantees. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, pp. 319–328, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450361255. doi: 10.1145/3287560.3287586. URL `https://doi.org/10.1145/3287560.3287586`.

Natasha Crampton. Microsoft's framework for building AI systems responsibly, 2022. `https://blogs.microsoft.com/on-the-issues/2022/06/21/microsofts-framework-for-building-ai-systems-responsibly/`.

Kate Crawford. The Hidden Biases in Big Data. *Harvard business review*, 1(4), 2013.

Kate Crawford. The Trouble with Bias. NeurIPS keynote, 2017. `https://www.youtube.com/watch?v=fMym_BKWQzk`.

Frances Ding, Moritz Hardt, John Miller, and Ludwig Schmidt. Retiring adult: New datasets for fair machine learning. In *NeurIPS*, pp. 6478–6490, 2021.

Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard S. Zemel. Fairness through awareness. In *ITCS*, pp. 214–226. ACM, 2012.

Kurt Eisemann. The trim problem. *Management Science*, 3(3):279–284, 1957.

Michael Feldman, Sorelle A. Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and removing disparate impact. In *KDD*, pp. 259–268. ACM, 2015.

Yoav Freund and Robert E. Schapire. Game theory, on-line prediction and boosting. In *Proceedings of the Ninth Annual Conference on Computational Learning Theory (COLT)*, pp. 325–332, 1996.

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

Google. Responsible AI progress report, 2025. `https://ai.google/static/documents/ai-responsibility-update-published-february-2025.pdf`.

Igor Griva, Stephen G. Nash, and Ariela Sofer. *Linear and Nonlinear Optimization*. SIAM, 2nd edition, 2008.

Moritz Hardt, Eric Price, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. In *Advances in Neural Information Processing Systems*, 2016. `https://arxiv.org/abs/1610.02413`.

Hans Hofmann. Statlog (German Credit Data). UCI Machine Learning Repository, 1994. DOI: https://doi.org/10.24432/C5NC77.

Kenneth Holstein, Jennifer Wortman Vaughan, Hal Daumé, Miroslav Dudík, and Hanna Wallach. Improving fairness in machine learning systems: What do industry practitioners need? In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, pp. 1–16, 2019.

Maliha Tashfia Islam, Anna Fariha, Alexandra Meliou, and Babak Salimi. Through the data management lens: Experimental analysis and evaluation of fair classification. In *SIGMOD Conference*, pp. 232–246. ACM, 2022.

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

T. Kamishima, S. Akaho, and J. Sakuma. Fairness-aware learning through regularization approach. In *2011 IEEE 11th International Conference on Data Mining Workshops*, pp. 643–650, 2011.

Jyrki Kivinen and Manfred K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997.

Allison Koenecke, Andrew Nam, Emily Lake, Joe Nudell, Minnie Quartey, Zion Mengesha, Connor Toups, John R. Rickford, Dan Jurafsky, and Sharad Goel. Racial disparities in automated speech recognition. *Proceedings of the National Academy of Sciences*, 117(14):7684–7689, 2020. doi: 10.1073/pnas.1915768117. URL `https://www.pnas.org/doi/abs/10.1073/pnas.1915768117`.

Jeff Larson, Surya Mattu, Lauren Kirchner, and Julia Angwin. How we analyzed the COMPAS recidivism algorithm. ProPublica, 2016.

Michelle Seng Ah Lee and Jat Singh. The Landscape and Gaps in Open Source Fairness Toolkits. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, 2021.

Joshua R. Loftus, Chris Russell, Matt J. Kusner, and Ricardo Silva. Causal reasoning for algorithmic fairness. *CoRR*, abs/1805.05859, 2018. URL `http://arxiv.org/abs/1805.05859`.

Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM Comput. Surv.*, 54(6), jul 2021. doi: 10.1145/3457607. URL `https://doi.org/10.1145/3457607`.

A. K. Menon and R. C. Williamson. The cost of fairness in binary classification. In *Proceedings of the Conference on Fiarness, Accountability, and Transparency*, 2018.

Microsoft. Responsible AI transparency report, 2025. https://aka.ms/Responsible-AI-Transparency-Report-2025.

Safiya Umoja Noble. *Algorithms of Oppression: How Search Engines Reinforce Racism*. NYU Press, 2018.

Ziad Obermeyer, Brian Powers, Christine Vogeli, and Sendhil Mullainathan. Dissecting Racial Bias in an Algorithm Used to Manage the Health of Populations. *Science*, 366(6464):447–453, 2019.

Cathy O'Neil. *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*. Crown, 2016.

Dana Pessach and Erez Shmueli. A review on fairness in machine learning. *ACM Comput. Surv.*, 55(3): 51:1–51:44, 2023.

Vasi Philomin. A progress update on our commitment to safe, responsible generative AI, 2024. https://aws.amazon.com/blogs/machine-learning/a-progress-update-on-our-commitment-to-safe-responsible-generative-ai/.

Geoff Pleiss, Manish Raghavan, Felix Wu, Jon M. Kleinberg, and Kilian Q. Weinberger. On fairness and calibration. In *NIPS*, pp. 5680–5689, 2017.

Manish Raghavan, Solon Barocas, Jon Kleinberg, and Karen Levy. Mitigating bias in algorithmic hiring: evaluating claims and practices. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, pp. 469—-481, New York, NY, USA, 2020. Association for Computing Machinery.

Renee Shelby, Shalaleh Rismani, Kathryn Henne, AJung Moon, Negar Rostamzadeh, Paul Nicholas, N'Mah Yilla-Akbari, Jess Gallegos, Andrew Smart, Emilio Garcia, et al. Sociotechnical harms of algorithmic systems: Scoping a taxonomy for harm reduction. In *Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 723–741, 2023.

Helen Smith. Algorithmic bias: should students pay the price? *AI & Society*, 35:1077–1078, 2020.

The European Parliament and The Council of the European Union. Regulation (EU) 2024/1689 of the european parliament and of the council of 13 june 2024 laying down harmonised rules on artificial intelligence and amending regulations (EC) No 300/2008, (EU) No 167/2013, (EU) No 168/2013, (EU) 2018/858, (EU) 2018/1139 and (eu) 2019/2144 and directives 2014/90/EU, (EU) 2016/797 and (EU) 2020/1828 (Artificial Intelligence Act) (Text with EEA relevance), 2024. https://eur-lex.europa.eu/eli/reg/2024/1689/oj.

Hanna Wallach and Miroslav Dudík. Fairness-related harms in AI systems: Examples, assessment, and mitigation. Microsoft Research webinar, 2021. https://www.youtube.com/watch?v=1RptHwfkx_k.

Hilde Weerts, Miroslav Dudík, Richard Edgar, Adrin Jalali, Roman Lutz, and Michael Madaio. Fairlearn: Assessing and improving fairness of AI systems. *Journal of Machine Learning Research*, 24(257):1–8, 2023. URL http://jmlr.org/papers/v24/23-0389.html.

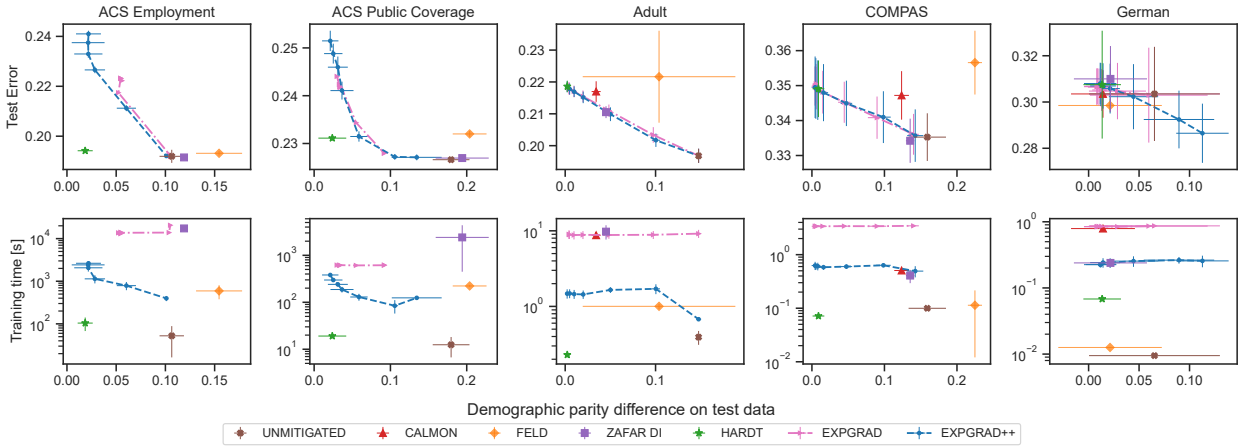
Blake E. Woodworth, Suriya Gunasekar, Mesrob I. Ohannessian, and Nathan Srebro. Learning non-discriminatory predictors. In *Proceedings of the 30th Conference on Learning Theory (COLT)*, pp. 1920–1953, 2017.

Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez-Rodriguez, and Krishna P. Gummadi. Fairness constraints: Mechanisms for fair classification. In *AISTATS*, volume 54 of *Proceedings of Machine Learning Research*, pp. 962–970. PMLR, 2017a.

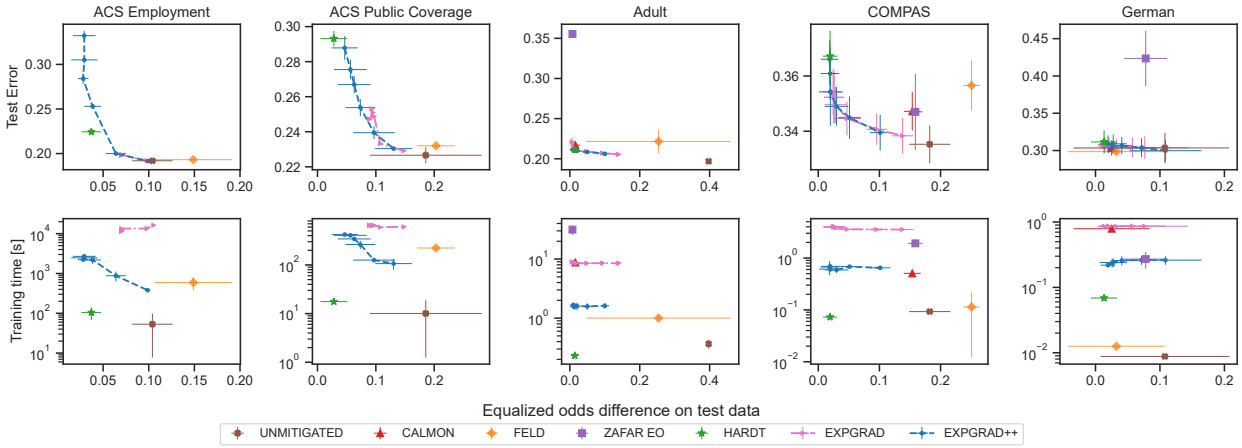
Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez-Rodriguez, and Krishna P. Gummadi. Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. In *WWW*, pp. 1171–1180. ACM, 2017b.

Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. Mitigating unwanted biases with adversarial learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 335–340, 2018.

A Additional experiment results

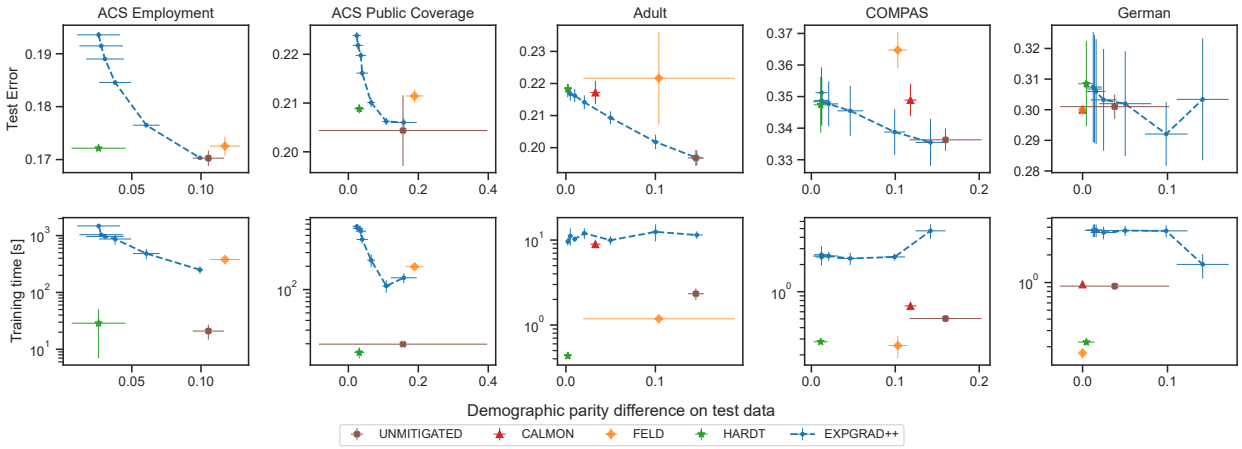


(a) Demographic Parity

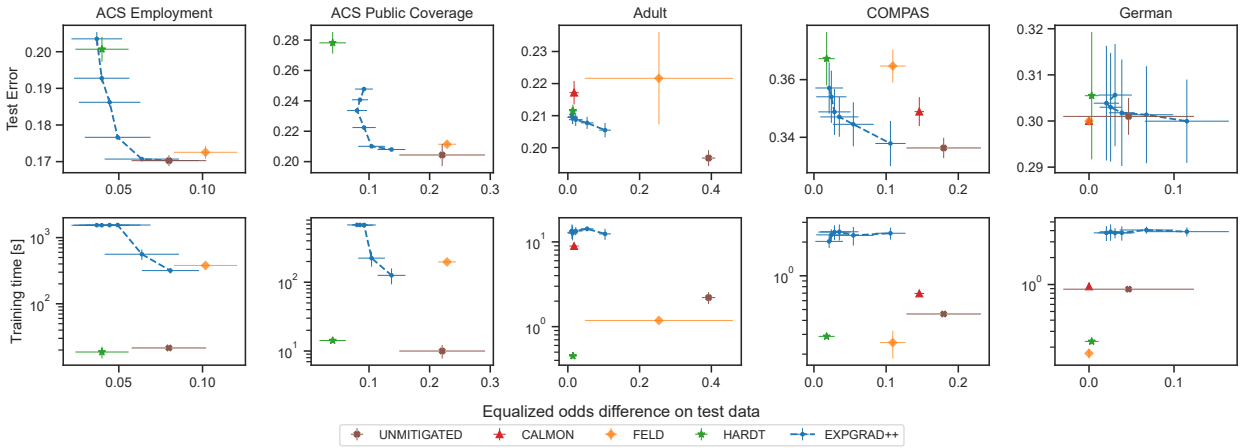


(b) Equalized Odds

Figure 4: *Overall performance comparison (base learner: logistic regression).*

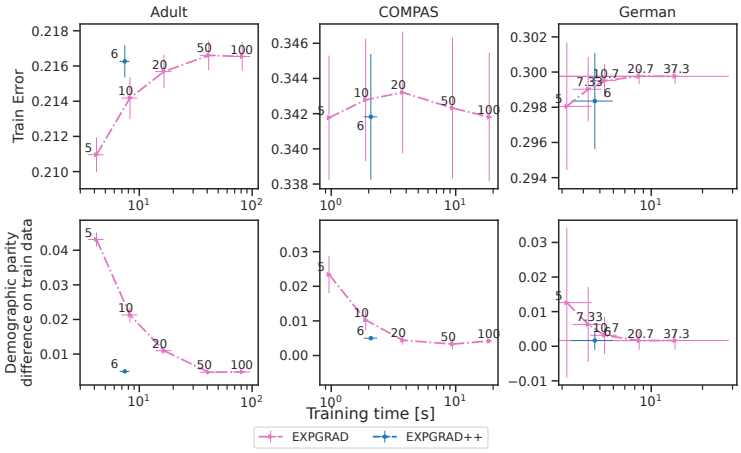


(a) Demographic Parity

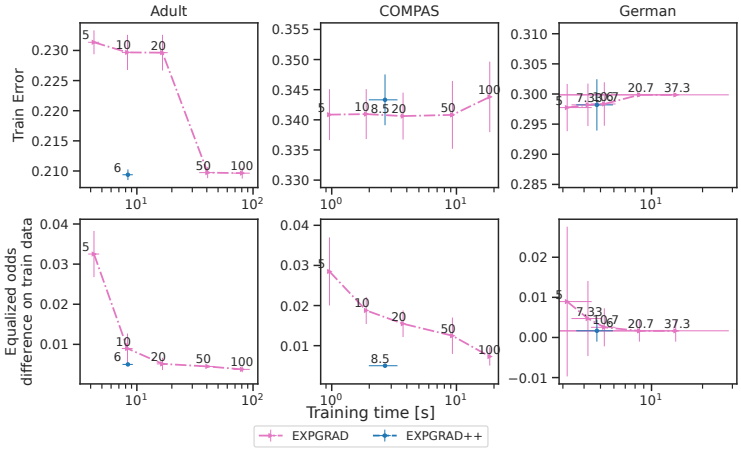


(b) Equalized Odds

Figure 5: *Overall performance comparison (base learner: boosting).*



(a) Demographic Parity.



(b) Equalized odds.

Figure 6: *Efficacy of column generation (base learner: boosting).*

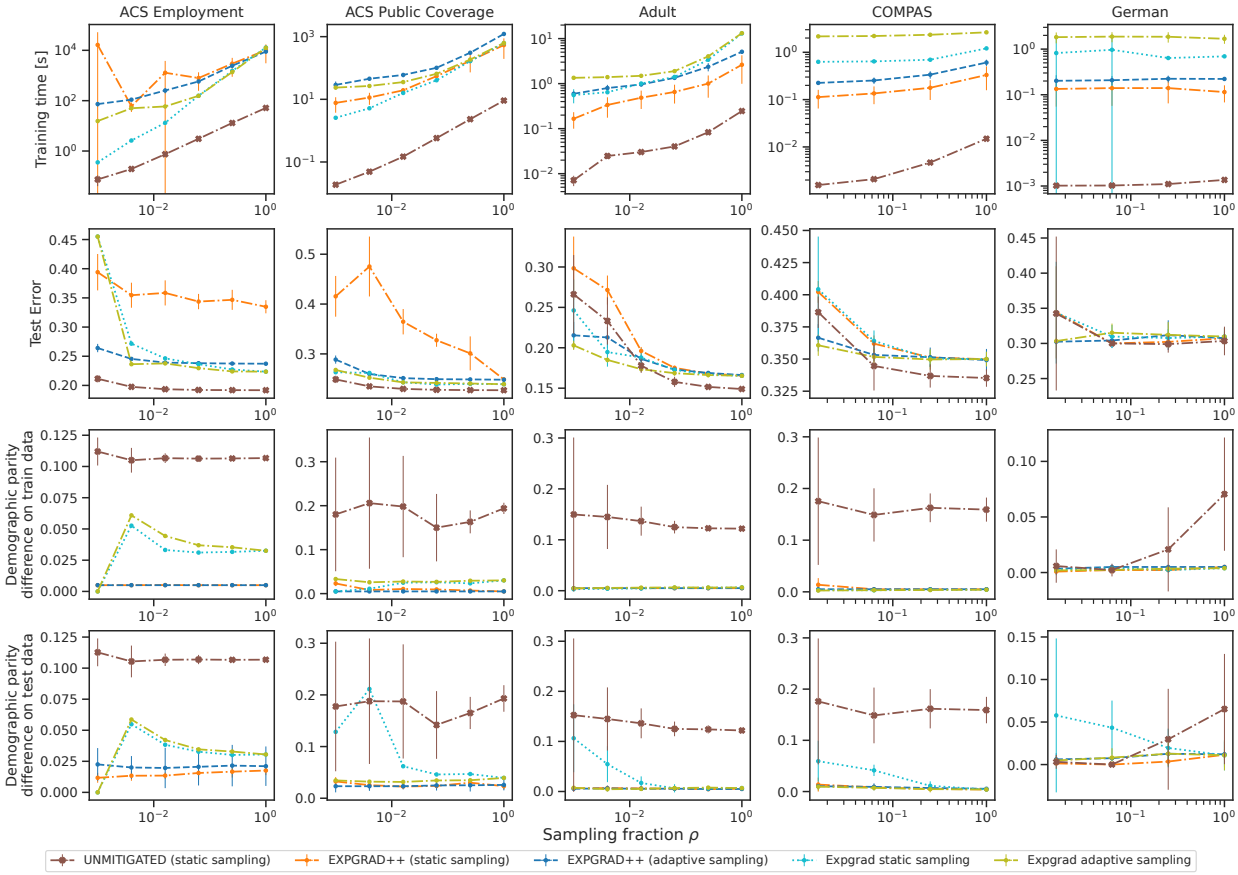


Figure 7: *Efficacy of sampling (base learner: logistic regression; fairness: demographic parity).*

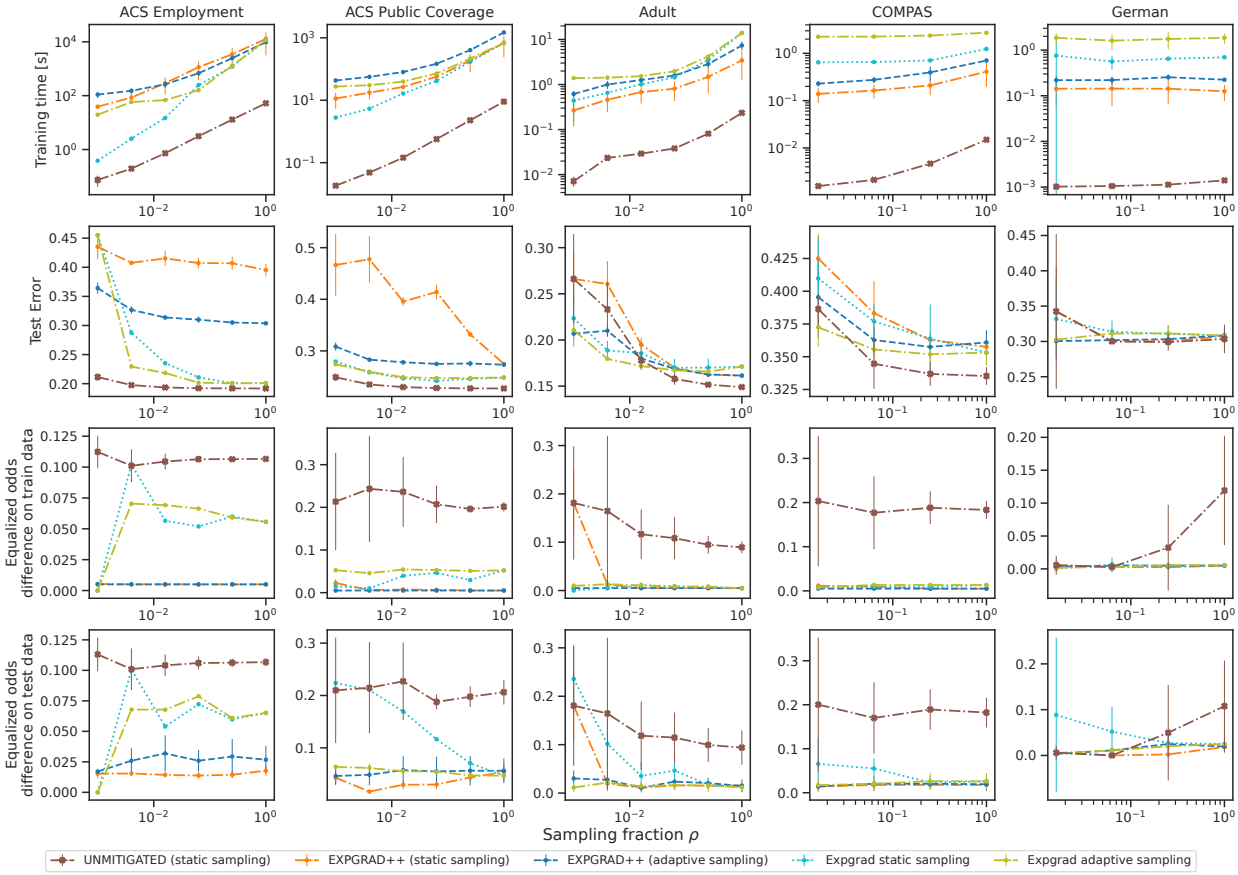


Figure 8: *Efficacy of sampling (base learner: logistic regression; fairness: equalized odds).*

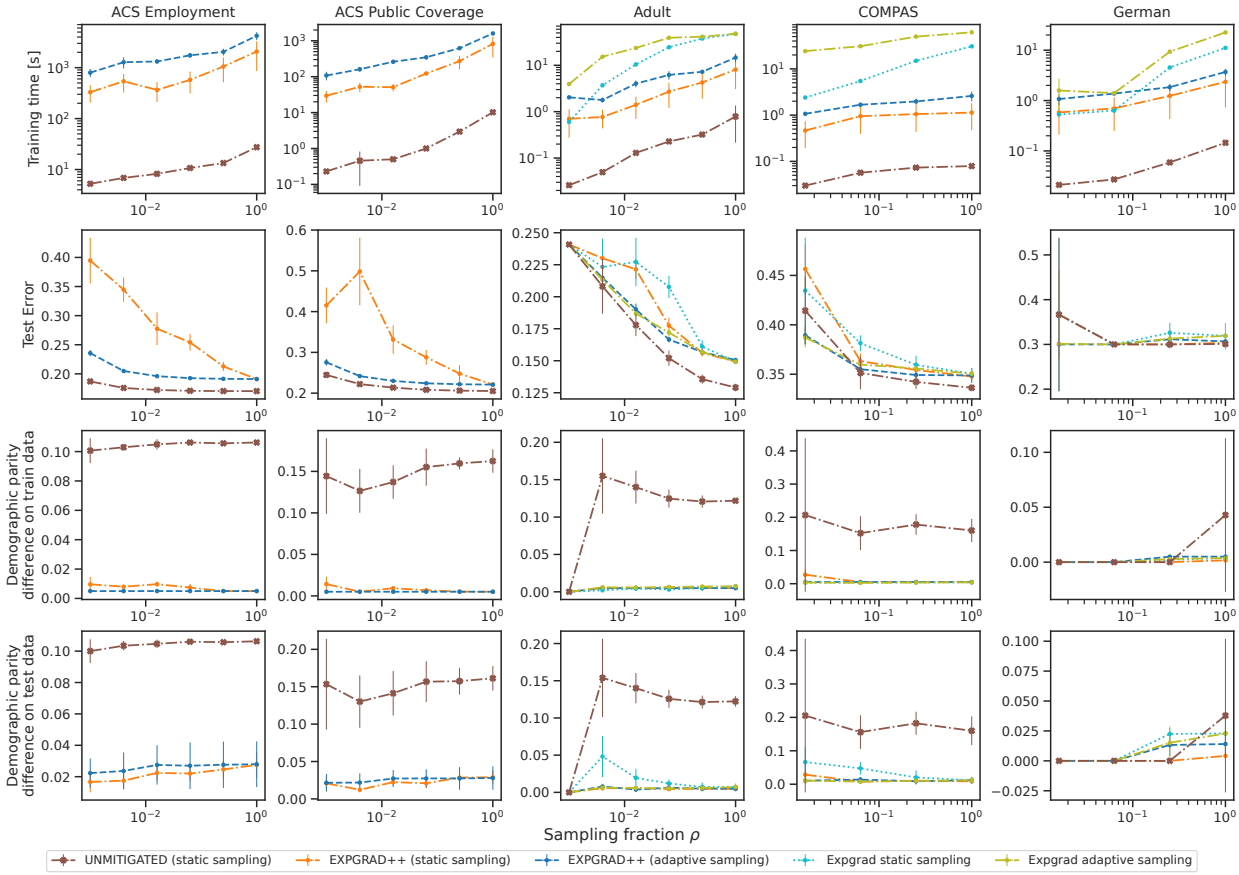


Figure 9: *Efficacy of sampling (base learner: boosting; fairness: demographic parity).*

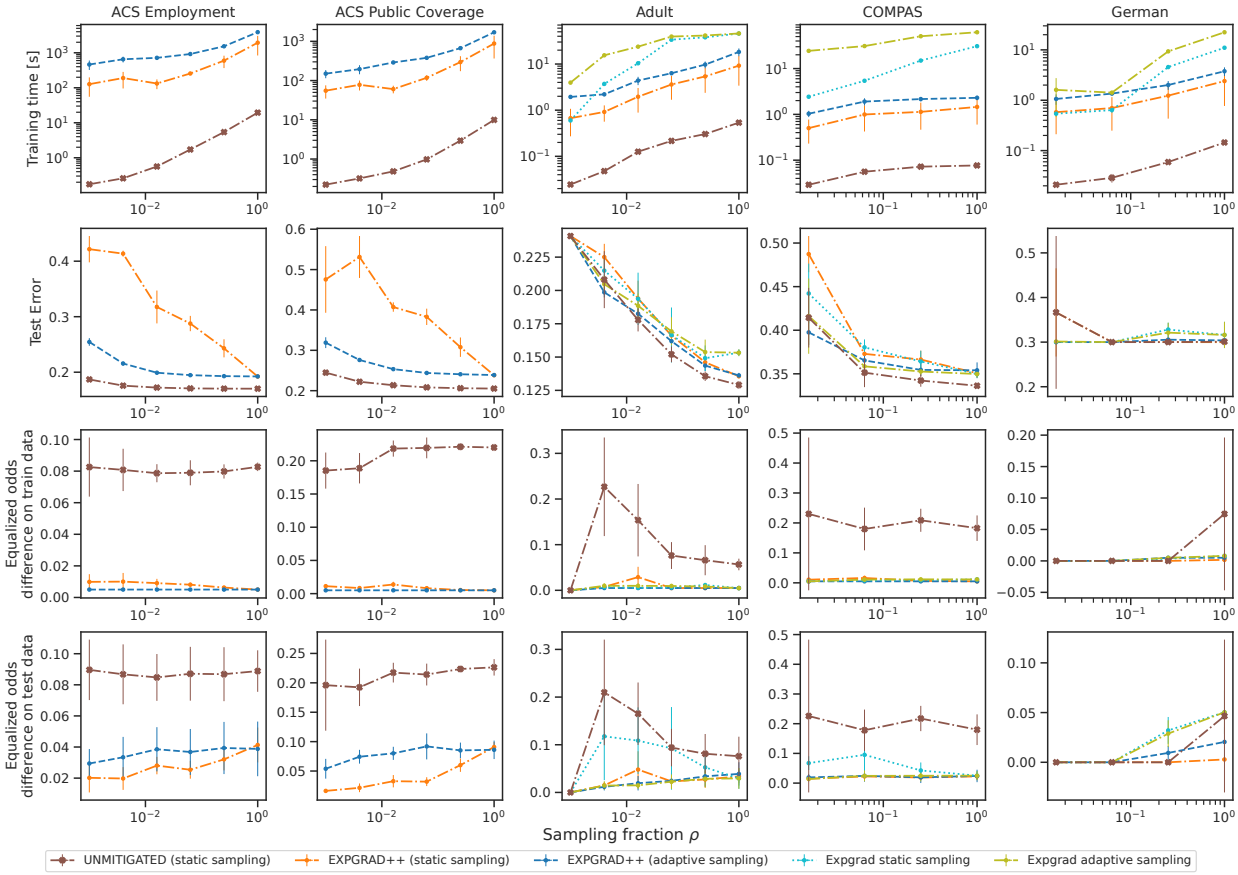


Figure 10: *Efficacy of sampling (base learner: boosting; fairness: equalized odds).*