A Supplementary Material

Fig. 8. Attack success for least vulnerable subgroups on the **COMPAS** dataset.

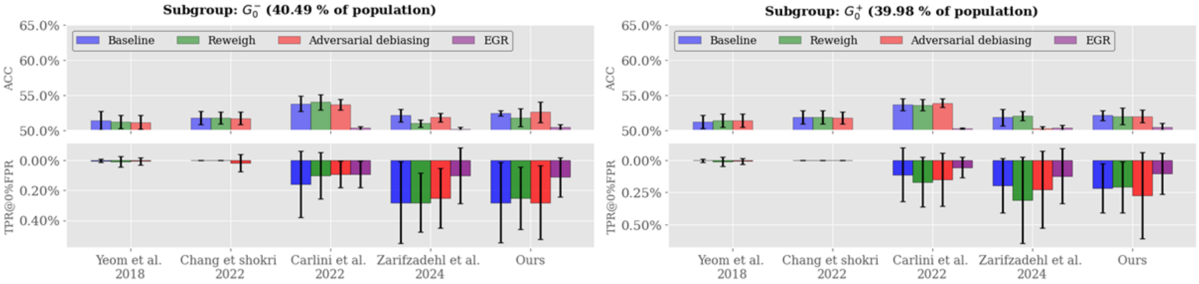

Fig. 9. Attack success for least vulnerable subgroups on the **Law School** dataset.

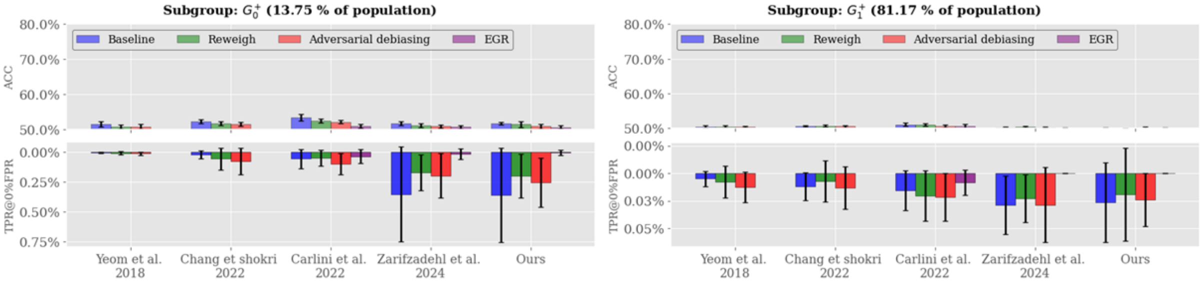

Fig. 10. Attack success for least vulnerable subgroups on the **Bank Marketing** dataset.

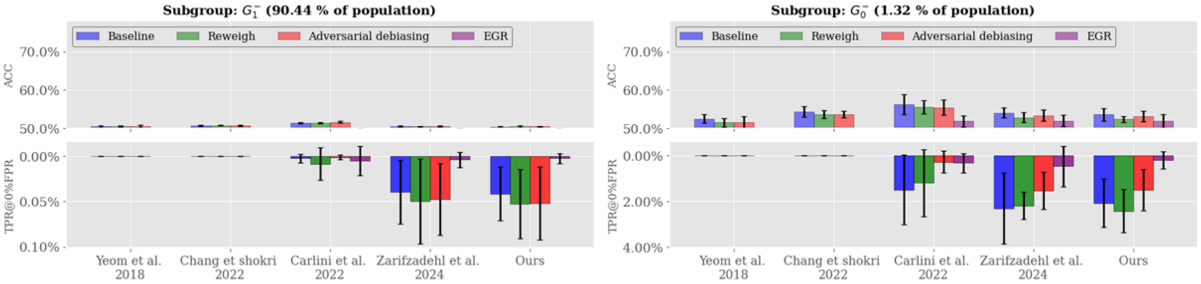

Table 4. Parameter configurations and attack performance for most vulnerable subgroups in the **COMPAS** dataset. TPR is reported at 0% FPR

		$\mathscr{G}_1^-$					$\mathscr{G}_1^+$				
Sampling	**Method**	`z`	a	γ	TPR (%)	ACC (%)	`z_val`	a	γ	TPR (%)	ACC (%)
Population	Unconstrained	75	0.8	1.5	1.1 ± 0.5	55.0 ± 2.0	50	0.9	1.8	1.9 ± 2.9	55.1 ± 2.2
	Reweigh	50	0.7	1.5	1.1 ± 0.9	55.0 ± 1.2	75	1.0	2.0	1.7 ± 1.7	53.8 ± 1.9
	Adversarial Debiasing	50	0.6	1.5	1.1 ± 0.9	54.9 ± 1.6	50	0.9	1.5	2.2 ± 3.2	56.3 ± 2.0
	EGR	50	0.0	1.8	0.2 ± 0.4	50.7 ± 0.7	5	0.2	1.8	0.3 ± 0.5	50.1 ± 0.3
Subgroup	Unconstrained	75	0.8	1.5	1.2 ± 0.4	54.0 ± 1.6	25	1.0	1.5	1.8 ± 2.7	57.7 ± 2.0
	Reweigh	25	0.8	1.5	1.1 ± 0.8	53.6 ± 1.1	75	1.0	1.8	2.0 ± 2.1	55.8 ± 2.4
	Adversarial Debiasing	25	0.7	2.0	1.1 ± 1.0	51.9 ± 1.7	50	1.0	2.0	2.1 ± 2.2	56.7 ± 3.0
	EGR	75	1.0	1.8	0.3 ± 0.5	50.4 ± 0.4	25	0.1	1.8	0.4 ± 1.0	51.1 ± 1.1

Table 5. Parameter configurations and attack performance for least vulnerable subgroups in the **COMPAS** dataset. TPR is reported at 0% FPR

		$\mathscr{G}_0^-$					$\mathscr{G}_0^+$				
Sampling	**Method**	`z`	a	γ	TPR (%)	ACC (%)	`z`	a	γ	TPR (%)	ACC (%)
Population	Unconstrained	50	0.9	1.5	0.3 ± 0.3	52.1 ± 0.9	50	0.9	1.5	0.2 ± 0.2	51.8 ± 1.2
	Reweigh	75	0.9	2.0	0.3 ± 0.2	51.0 ± 0.5	50	0.7	1.2	0.3 ± 0.3	52.1 ± 0.7
	Adversarial Debiasing	75	0.9	1.5	0.3 ± 0.2	51.8 ± 0.6	5	0.5	2.0	0.2 ± 0.3	50.3 ± 0.2
	EGR	50	0.6	1.5	0.1 ± 0.2	50.2 ± 0.2	5	0.3	1.8	0.1 ± 0.2	50.3 ± 0.4
Subgroup	Unconstrained	75	0.8	1.2	0.3 ± 0.3	52.4 ± 0.4	75	0.6	1.5	0.2 ± 0.2	52.1 ± 0.7
	Reweigh	75	0.7	1.5	0.3 ± 0.2	51.8 ± 1.3	75	0.0	1.8	0.2 ± 0.2	52.0 ± 1.2
	Adversarial Debiasing	75	0.8	1.2	0.3 ± 0.2	52.6 ± 1.4	75	0.8	1.5	0.3 ± 0.3	52.0 ± 0.9
	EGR	50	0.4	1.2	0.1 ± 0.1	50.4 ± 0.4	50	0.3	2.0	0.1 ± 0.2	50.5 ± 0.5

Table 6. Parameter configurations and attack performance for least vulnerable subgroups in the **Bank marketing** dataset. TPR is reported at 0% FPR

		$\mathscr{G}_1^-$					$\mathscr{G}_0^-$				
Sampling	**Method**	z	a	γ	TPR (%)	ACC (%)	z	a	γ	TPR (%)	ACC (%)
Population	Unconstrained	50	0.1	1.8	0.0 ± 0.0	50.5 ± 0.1	50	0.5	1.8	2.3 ± 1.6	54.0 ± 1.3
	Reweigh	50	0.3	2.0	0.0 ± 0.0	50.5 ± 0.1	50	0.5	1.8	2.2 ± 0.6	52.8 ± 1.3
	Adversarial Debiasing	75	0.1	1.8	0.0 ± 0.0	50.6 ± 0.2	50	0.0	1.2	1.5 ± 0.8	53.4 ± 1.5
	EGR	75	1.0	1.2	0.0 ± 0.0	50.0 ± 0.1	75	1.0	1.2	0.5 ± 0.9	52.0 ± 1.5
Subgroup	Unconstrained	75	0.0	1.8	0.0 ± 0.0	50.4 ± 0.2	75	0.6	2.0	2.1 ± 1.1	53.6 ± 1.6
	Reweigh	75	0.3	2.0	0.1 ± 0.0	50.5 ± 0.3	50	0.6	1.8	2.4 ± 0.9	52.4 ± 0.8
	Adversarial Debiasing	50	0.1	1.8	0.1 ± 0.0	50.5 ± 0.1	75	0.5	1.2	1.5 ± 0.9	53.2 ± 1.4
	EGR	75	1.0	0.2	0.0 ± 0.0	50.0 ± 0.1	50	1.0	2.0	0.2 ± 0.4	51.9 ± 1.7

Table 7. Parameter configurations and attack performance for most vulnerable subgroups in the **Law School** dataset. TPR is reported at 0% FPR

		$\mathscr{G}_0^-$					$\mathscr{G}_1^-$				
Sampling	**Method**	z	a	γ	TPR (%)	ACC (%)	z	a	γ	TPR (%)	ACC (%)
Population	Unconstrained	75	0.5	2.0	2.7 ± 2.8	59.7 ± 3.1	75	0.3	2.0	2.5 ± 2.0	55.8 ± 2.4
	Reweigh	75	0.3	2.0	1.9 ± 0.8	55.8 ± 2.2	75	0.1	1.5	2.3 ± 2.6	57.9 ± 2.2
	Adversarial Debiasing	75	0.5	2.0	1.6 ± 2.2	54.4 ± 1.9	50	0.9	1.8	1.2 ± 2.0	53.0 ± 1.4
	EGR	25	1.0	1.8	0.4 ± 0.7	50.4 ± 0.6	50	0.2	2.0	0.1 ± 0.3	50.1 ± 0.3
Subgroup	Unconstrained	75	1.0	2.0	4.3 ± 3.3	66.3 ± 4.0	75	1.0	2.0	3.2 ± 3.1	64.9 ± 4.0
	Reweigh	50	0.9	2.0	2.7 ± 2.2	60.1 ± 2.7	75	0.6	2.0	2.6 ± 1.9	62.9 ± 3.6
	Adversarial Debiasing	75	1.0	2.0	2.7 ± 1.9	61.6 ± 2.3	50	0.9	1.5	1.4 ± 1.9	57.4 ± 2.3
	EGR	50	0.0	1.8	0.4 ± 0.8	50.6 ± 0.9	25	0.0	1.5	0.3 ± 0.7	50.2 ± 0.6

Table 8. Parameter configurations and attack performance for least vulnerable subgroups in the **Law School** dataset. TPR is reported at 0% FPR

		$\mathscr{G}_{\mathbf{0}}^{+}$					$\mathscr{G}_{\mathbf{1}}^{+}$				
Sampling	**Method**	$\mathtt{z}$	a	γ	TPR (%)	ACC (%)	$\mathtt{z}$	a	γ	TPR (%)	ACC (%)
Population	Unconstrained	50	0.6	1.8	0.4 ± 0.4	51.6 ± 0.6	75	0.8	2.0	0.0 ± 0.0	50.3 ± 0.1
	Reweigh	75	0.9	2.0	0.2 ± 0.2	51.0 ± 0.6	75	0.4	1.8	0.0 ± 0.0	50.5 ± 0.2
	Adversarial Debiasing	50	0.7	1.8	0.2 ± 0.2	50.8 ± 0.5	75	0.5	1.2	0.0 ± 0.0	50.3 ± 0.2
	EGR	75	0.3	2.0	0.0 ± 0.0	50.6 ± 0.5	5	0.1	0.2	0.0 ± 0.0	50.1 ± 0.1
Subgroup	Unconstrained	75	0.6	1.8	0.4 ± 0.4	51.7 ± 0.4	75	0.9	2.0	0.0 ± 0.0	50.1 ± 0.2
	Reweigh	75	0.0	1.2	0.2 ± 0.2	51.4 ± 0.8	5	0.8	2.0	0.0 ± 0.0	50.1 ± 0.1
	Adversarial Debiasing	50	0.0	1.2	0.3 ± 0.2	50.9 ± 0.7	75	0.3	1.2	0.0 ± 0.0	50.2 ± 0.2
	EGR	75	0.0	1.8	0.0 ± 0.0	50.6 ± 0.5	5	0.2	1.2	0.0 ± 0.0	50.1 ± 0.2

B Reproducibility Statement

B.1 Datasets

We obtained the Law School dataset from Kaggle [57], the COMPAS dataset from the ProPublica GitHub repository [39], and the Bank Marketing dataset from Microsoft's Fairlearn package [50].

Table 9. Representation of subgroups in each dataset.

	G_0^-	G_1^-	G_0^+	G_1^+
Bank Marketing (age)	1.32%	90.44%	0.37%	7.87%
COMPAS (race)	40.49%	12.47%	39.98%	7.07%
Law School (race)	2.34%	2.74%	13.75%	81.17%

B.2 Model Architecture and Training

We used the same neural network architecture across all experiments, tuning hyperparameters for each dataset. The base model has three hidden layers of sizes [200, 100, 50] with ReLU activations, and a sigmoid output layer for binary classification. Training is done in PyTorch using the Adam optimizer, with learning rates and batch sizes adjusted per dataset. Key settings include an adaptive learning rate schedule (ReduceLROnPlateau) and 100 training epochs for each model. These settings were chosen to ensure our results met or exceeded the performance benchmarks set by previous work [18].

Table 10. Dataset-specific hyperparameters.

Dataset	**Learning Rate**	**Batch Size**
Law School	0.002	512
Bank Marketing	0.001	512
COMPAS	0.003	256

B.3 Fairness Algorithms

We used and, where necessary, adapted open-source implementations of the fairness algorithms evaluated in our study. Specifically, we used Microsoft's Fairlearn Python package [50] for the Exponentiated Gradient Reduction algorithm. For the Reweighing algorithm, we used the implementation available in IBM's AIF360 Python package [9]. We implemented the Adversarial Debiasing algorithm in PyTorch, closely following the methodology outlined in [66] as well as the tensorflow implementation provided in IBM AIF360 package [9].

B.3.1 Exponentiated Gradient Reduction (EGR)

EGR [2] is an in-processing algorithm that recasts fair classification as a constrained optimization problem. It reduces the task to a series of cost-sensitive classification problems, yielding a randomized classifier (i.e., a distribution over classifiers) on the Pareto frontier of accuracy and fairness. In practice, EGR uses a Lagrangian framework where a hypothesis selector minimizes prediction error while a fairness regulator updates weights via an *exponentiated gradient* method. The final model achieves near-optimal accuracy under the specified fairness constraints.

B.3.2 Adversarial Debiasing

Adversarial debiasing[66] uses an adversarial framework involving two models:

- A **predictor** that learns to accurately predict the true label Y.
- An **adversary** that attempts to infer the protected attribute A from the predictor's output.

The predictor minimizes a combination of the standard loss (e.g., cross-entropy) and a weighted adversarial loss that penalizes it when the adversary succeeds. To enforce equalized odds, the adversary receives both the prediction $\hat{Y}$ and the true label Y as inputs, ensuring that the predictor's output does not reveal A. Techniques such as gradient reversal facilitate this minimax training process.

B.3.3 Reweighing

Reweighing [36] is a preprocessing method that adjusts the weights of training instances to mitigate bias. For binary sensitive attributes and outcomes, it computes weights so that each group-label combination is proportionally represented. Formally, the weight for an instance with $A = a$ and $Y = y$ is given by:

$$w_{(A=a,Y=y)} = \frac{P(A=a)P(Y=y)}{P(A=a,Y=y)}, \tag{12}$$

which renders A and Y statistically independent in the weighted dataset. Training a standard classifier on this reweighted data helps reduce inherited biases from the original distribution.

B.4 Privacy attacks

We implemented both the Likelihood Ratio Attack (LiRA) and the Pairwise Likelihood Ratio Attack (RMIA) using the original code provided by the respective authors on GitHub. For LiRA, this is available at [48], for RMIA this can be found at [55]. To align with our study's framework, we adapted these implementations to the binary classification setting, ensuring that the core methodologies remained faithful to those described in the original publications.

Table 11. Exploration grid for a, γ, and the number of samples z for the pairwise likelihood ratio attacks

Parameter	Values Explored
a	{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0}
γ	{0.1, 0.25, 0.5, 0.75, 1.0, 1.25, 1.50, 1.75, 2.0}
z	{5, 25, 50, 75}

B.5 Code and data availability

All reported results and the experiments can be reproduced by running our code with the same random seeds and configurations. We are happy to make our code and all intermediate analysis files, including model weights available upon request.