Generalized Disparate Impact for Configurable Fairness Solutions in ML

Luca Giuliani [*1] **Eleonora Misino** [*1] **Michele Lombardi** [1]

Abstract

We make two contributions in the field of AI fairness over continuous protected attributes. First, we show that the Hirschfeld-Gebelein-Renyi (HGR) indicator (the only one currently available for such a case) is valuable but subject to a few crucial limitations regarding semantics, interpretability, and robustness. Second, we introduce a family of indicators that are: 1) complementary to HGR in terms of semantics; 2) fully interpretable and transparent; 3) robust over finite samples; 4) configurable to suit specific applications. Our approach also allows us to define fine-grained constraints to permit certain types of dependence and forbid others selectively. By expanding the available options for continuous protected attributes, our approach represents a significant contribution to the area of fair artificial intelligence.

1. Introduction

In recent years, the social impact of data-driven AI systems and its ethical implications have become widely recognized. For example, models may discriminate over population groups (Julia et al., 2016; Gianfrancesco et al., 2018), spurring extensive research on AI fairness. Typical approaches in this area involve quantitative indicators defined over a "protected attribute", which can be used to measure discrimination or enforce fairness constraints. On the one hand, such metrics are arguably the most viable solution for mitigating fairness issues; on the other hand, the nuances of ethics can hardly be reduced to simple rules. From this point of view, the availability of multiple and diverse metrics is a significant asset since it enables choosing the best indicator depending on the application.

Regarding available solutions, the case of categorical protected attributes is well covered by multiple indicators (see Section 2). Conversely, a single approach works with continuous protected attributes at the moment of writing; this is the *Hirschfeld-Gebelein-Renyi* (HGR) correlation coefficient (Rényi, 1959), which has two viable implementations for Machine Learning (ML) systems (Mary et al., 2019; Grari et al., 2020).

We view the lack of diverse techniques for continuous protected groups as a major issue. We contribute to this area by 1) identifying a few critical limitations in the HGR approach, and 2) introducing a family of indicators that complement HGR semantics and have technical advantages.

In terms of limitations, we highlight how the theoretical HGR formulation is prone to pathological behavior for finite samples, leading to the oversized importance of implementation details and limited interpretability. Moreover, the generality of the HGR formulation makes the indicator unsuitable for exclusively measuring the functional dependency between the protected attribute and the target. Finally, the HGR indicator cannot account for scale effects on fairness since it is based on the scale-invariant Pearson's correlation coefficient.

We introduce the Generalized Disparate Impact (GeDI), a family of indicators inspired by the HGR approach and by the Disparate Impact Discrimination Index (Aghaei et al., 2019). GeDI indicators measure the dependency based on how well a user-specified function of the protected attribute can approximate the target variable. Our indicators support both discrete and continuous protected attributes and 1) complement the HGR semantics, 2) are fully interpretable, 3) are robust for finite samples, and 4) can be extensively configured. Indicators in the family share a core technical structure that allows for a uniform interpretation and unified techniques for stating fairness constraints. Moreover, the GeDI formulation supports fine-grained constraints in order to permit only certain types of dependency while ruling out others. By introducing GeDI, we aim to improve metrics diversity while limiting complexity.

[*]Equal contribution [1]Department of Computer Science and Engineering, University of Bologna, Bologna, Italy. Correspondence to: Luca Giuliani <luca.giuliani13@unibo.it>, Eleonora Misino <eleonora.misino2@unibo.it>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

2. Background and Motivation

State-of-the-art research on algorithmic fairness focuses on measuring discrimination and enforcing fairness constraints. Some techniques (Kamiran & Calders, 2011; Calmon et al., 2017) operate by transforming the data before training the model to mitigate discrimination and are referred to as *pre-processing approaches*. Conversely, *post-processing approaches* calibrate the predictive model once trained (Calders & Verwer, 2010; Hardt et al., 2016). Finally, *in-processing approaches* focus on removing discrimination at learning time. To this end, (Menon & Williamson, 2018) modify the class-probability estimates during training, while (Kamishima et al., 2012; Zafar et al., 2017; Donini et al., 2018; Padala & Gujar, 2020; Komiyama et al., 2018) embed fairness in the learning procedure through constraints in the objective. All the approaches mentioned above rely on metrics restricted to discrete protected attributes. Two recent works (Mary et al., 2019; Grari et al., 2020) extend the method by (Kamishima et al., 2011) to continuous variables by minimizing the Hirschfeld-Gebelein-Renyi (HGR) correlation coefficient (Rényi, 1959). A more extensive review of fairness approaches can be found in (Mehrabi et al., 2021).

Disparate Impact One of the most widely used notions of fairness is based on the legal concept of *disparate impact* (Supreme Court of the United States, 1971), which occurs whenever a neutral practice negatively impacts a protected group. The principle of disparate impact can be extended to ML models by considering their output with respect to protected attributes (Feldman et al., 2015). To quantify the disparate impact in regression and classification, (Aghaei et al., 2019) propose a fairness indicator deemed *Disparate Impact Discrimination Index* (DIDI). The higher the DIDI, the more disproportionate the model output is with respect to protected attributes, and the more it suffers from disparate impact.

The simplicity of the DIDI formulation makes it highly interpretable. Given a sample $\{x_i, y_i\}_{i=1}^n$ including values for a protected attribute x and a continuous target value y, the *Disparate Impact Discrimination Index* is referred to as $\mathrm{DIDI_r}(x, y)$ and defined as:

$$\sum_{v \in \mathcal{X}} \left| \frac{\sum_{i=1}^n y_i I(x_i = v)}{\sum_{i=1}^n I(x_i = v)} - \frac{1}{n} \sum_{i=1}^n y_i \right| \tag{1}$$

where $\mathcal{X}$ is the domain of x and $I(\phi)$ is the indicator function for the logical formula ϕ. The DIDI represents the sum of discrepancies between the average target for each protected group and for the whole dataset. The indicator has a specialized formulation for classification tasks, i.e., when y is discrete. In this case, $\mathrm{DIDI_c}(x, y)$ is defined as:

$$\sum_{u \in \mathcal{Y}} \sum_{v \in \mathcal{X}} \left| \frac{\sum_{i=1}^n I(y_i = u \wedge x_i = v)}{\sum_{i=1}^n I(x_i = v)} - \frac{1}{n} \sum_{i=1}^n I(y_i = u) \right|$$

The HGR Indicator To the best of our knowledge, the HGR indicator is the only fairness metric that applies to continuous protected attributes. It is based on the *Hirschfeld-Gebelein-Renyi* (HGR) correlation coefficient (Rényi, 1959), which is a normalized measure of the relationship between two random variables, $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$. When the coefficient is zero, the two variables are independent, while they are strictly dependent when it is equal to 1. Formally, the HGR correlation coefficient is defined as:

$$\mathrm{HGR}(X, Y) = \sup_{f,g} \rho(f(X), g(Y)) \tag{2}$$

where ρ is Pearson's correlation coefficient and f, g are two measurable functions (referred to as copula transformations) with finite and strictly positive variance.

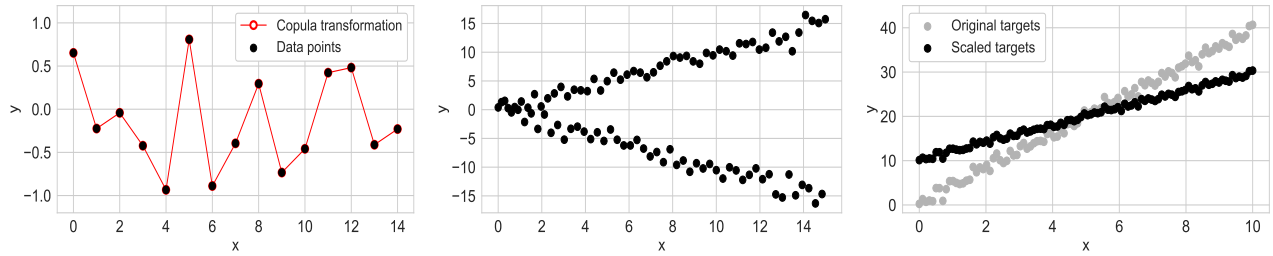
Limitations of the HGR Approach The HGR indicator has several noteworthy properties, including the ability to measure very general forms of dependency. Despite its strengths, however, it is not devoid of limitations. As a first contribution, we identify three of them.

First, when applied to finite samples, the theoretical HGR formulation is prone to pathological behavior. In finite datasets, the Pearson's correlation in Equation (2) is replaced with its sample version, leading to:

$$\mathrm{hgr}(x, y) = \max_{f,g} r(f_x, g_x)) = \max_{f,g} \frac{cov(f_x, g_y)}{\sigma(f_x)\sigma(g_y)} \tag{3}$$

where f_x, g_y, are short notations for $f(x)$ and $g(y)$, respectively; $\sigma(\cdot)$ is the sample standard deviation, and $cov(\cdot, \cdot)$ is the sample covariance. Since the copula transformations are unrestricted, Equation (3) might be ill-behaved when the protected attribute takes many values. As an extreme case, with all-distinct x values, choosing $f(x_i) = y_i$ ensures maximum Pearson's correlation, leading to overly large HGR values (see Figure 1(a)). The existing implementations mitigate this issue by using models with finite variance for f and g – discretized KDE in (Mary et al., 2019), and Neural Networks in (Grari et al., 2020). However, these solutions link the indicator semantics to low-level, sub-symbolic details that are difficult to interpret and to control.

Second, using unrestricted copula transformations allows HGR to measure very general forms of dependency between x and y, including scenarios such as the diverging target values in Figure 1(b). However, there are situations where discrimination arises only when the *expected* value of the



(a) (b) (c)

Figure 1. (a) HGR indicator may overfit data points due to the unrestricted transformations; this leads to overly large correlation estimates in finite datasets, even if y is independent of x, as in the example. (b) Using unrestricted copula transformations allows HGR to capture non-functional dependencies, like the increasing divergence in target values. However, in some scenarios, discrimination is linked only to the function dependency $\mathbb{E}[y \mid x]$, and HGR is not able to measure it exclusively. (c) Since Pearson's correlation is scale-invariant, the HGR indicator is not able to capture scale effects on fairness.

target is affected, i.e., it is linked to the strength of the functional dependency $\mathbb{E}[y \mid x]$. For example, in Figure 1(b), a third confounder attribute, correlated with x, might motivate the divergence. In this case, the confounder might not raise any ethical concern, but the HGR indicator would not able to exclusively measure the functional dependency.

Third, the HGR indicator satisfies all the Renyi properties (Rényi, 1959) by relying on Pearson's correlation coefficient, but this makes the approach unable to account for scale effects on fairness. For example, let us assume the target values y are linearly correlated to the continuous protected attributes. In some practical cases, applying an affine transformation on y may reduce the discrimination (see Figure 1(c)), but the HGR indicator cannot capture this effect since Pearson's correlation is scale-invariant.

3. Generalized Disparate Impact

In this section, we derive the GeDI family of indicators and its semantics. The process is guided by two design goals: 1) complementing the HGR approach to provide more options for continuous protected attributes; 2) improving over the technical limitations we identified in Section 2.

HGR Computation as Optimization We start by observing that the sample Pearson correlation can be restated as the optimal solution of a Least Squares problem. Formally, $r(f_x, g_y)$ from Equation (3) is given by:

$$\arg\min_{r} \frac{1}{n} \left\| r \frac{f_x - \mu(f_x)}{\sigma(f_x)} - \frac{g_y - \mu(g_y)}{\sigma(g_y)} \right\|_2^2 \tag{4}$$

where $\mu(\cdot)$ is the sample average operator; this is a well-known statistical result, whose proof we report in Appendix A.1. Using Equation (4) may seem counter-intuitive since it casts the whole HGR computation process as a bilevel optimization problem. In particular, we have:

$$\max_{f,g} \arg\min_{r} \frac{1}{n} \left\| r \frac{f_x - \mu(f_x)}{\sigma(f_x)} - \frac{g_y - \mu(g_y)}{\sigma(g_y)} \right\|_2^2 \tag{5}$$

However, the two optimization objectives are aligned since larger r values correspond to lower squared residuals; this alignment can be exploited to obtain an alternative single-level formulation for the HGR indicator. The process is covered in detail in Appendix A.2 and leads to:

$$\mathrm{hgr}(x, y) = |r^*| \tag{6}$$

where r^* is obtained by solving:

$$\arg\min_{f,g,r} \frac{1}{n} \left\| r \frac{f_x - \mu(f_x)}{\sigma(f_x)} - \frac{g_y - \mu(g_y)}{\sigma(g_y)} \right\| \tag{7}$$

The equivalence holds if $\sigma(g_y), \sigma(f_x) > 0$, which is also needed for the Pearson correlation to be well-defined.

The GeDI Indicator Family The main insight from Equation (6) and (7) is that the HGR computation can be understood as a Least Square fitting. We derive the GeDI family by building on the same observation, with a few key differences. In particular, we define a GeDI indicator as a measure of how well a user-selected, interpretable function of the protected attribute x can approximate the target variable y. Formally, we have that:

$$\mathrm{GeDI}\,(x, y; F) = |d^*| \tag{8}$$

where d^* is defined via the optimization problem:

$$\begin{aligned} &\arg\min_{d,\alpha} \frac{1}{n} \left\| d(F\alpha - \mu(F\alpha)) - (y - \mu(y)) \right\|_2^2 \\ &\text{s.t. } \|\alpha\|_1 = 1 \end{aligned} \tag{9}$$

where $\alpha \in \mathbb{R}^k$ is a vector of coefficients, $d \in \mathbb{R}$ is a scale factor whose absolute value corresponds to the indicator

Table 1. Type of dependencies measured by indicators for continuous protected attributes. Double check-mark (✓✓) specifies that the indicator *exclusively* measures the corresponding dependency.

Measurable Dep.	GeDI	HGR-kde	HGR-nn
Functional	✓✓	✓	✓
Non-functional	×	✓	✓
Scale-independent	✓	✓✓	✓✓
Scale-dependent	✓	×	×
User-configurable	✓	×	×

value, and $\|\cdot\|_1$ is the L1 norm that we introduce to obtain one of the equivalent optimal solutions of the optimization problem. F is a $n \times k$ *kernel matrix* whose columns Finally, F_j represent the evaluation of the basis functions $f_j(x)$ of the protected attribute x, namely:

$$F\alpha = \sum_{j=1}^{k} F_j \alpha_j = \sum_{j=1}^{k} f_j(x)\alpha_j \tag{10}$$

Any kernel can be chosen, provided that the resulting matrix is full-rank. The kernel and its order k are part of the specification of a GeDI indicator and appear in its notation.

Rationale and Interpretation Our formulation differs from Equation (7) in three regards. First, it lacks the copula transformation on the y variable (the g function). The DIDI inspires this property: it allows our indicators to exclusively measure the strength of the functional dependency $\mathbb{E}[y \mid x]$ but also makes them incapable of measuring other forms of dependency. This semantics complements the HGR one, thus increasing the available options.

Second, the standardization terms have been replaced with the normalization constraint $\|\alpha\|_1 = 1$. This constraint allows us to obtain one of the equivalent optimal solutions, while keeping it viable for linear optimization approaches. The DIDI also inspires this choice as it makes our indicators sensitive to scale changes, complementing the HGR semantics. As expected, this prevents the satisfaction of some Renyi properties that exclusively apply to scale-invariant metrics.

Third, we restrict the copula transformation on x (the f function) to be linear over a possibly non-linear kernel. As a result, our indicator is fully interpretable. In particular, the indicator measures the overall functional dependency, while the coefficients identify which functional dependencies have the most significant effect, as determined by the kernel components.

Choice of Kernel Individual indicators from the GeDI family are obtained via the specification of a kernel, which allows users to define which types of functional dependency are relevant for the considered use case. This is the main criterion for the kernel choice and provides a level of configurability that is absent in existing indicators. Table 1 summarizes the types of dependency that can be measured by indicators with support for continuous protected attributes. A double check-mark specifies that the indicator *exclusively* measures the corresponding dependency.

Specifying a kernel might seem cumbersome, but it should be kept in mind that any HGR implementation also needs a mechanism to avoid ill behavior on finite samples. In (Mary et al., 2019) this is done via a finite discretization and a KDE bandwidth, while in (Grari et al., 2020) via a neural network architecture and stochastic gradient descent. In both cases, the mechanism is strongly linked to the implementation details, thus reducing transparency and control. Integrating the kernel choice into the indicator definition makes the bias-variance trade-off explicit and controllable.

The kernel choice can be simplified by selecting a parametric function family and then adjusting the order k to prevent overfitting and numerical issues. If the chosen function family ensures the F matrix is full-rank, we can asymptotically recover the unrestricted copula transformation f. For example, by choosing a polynomial kernel, we have:

$$f_j(x) = x^j \tag{11}$$

In our notation, $\mathrm{GeDI}\,(\cdot, \cdot, V^k)$ denotes the use of a polynomial kernel of order k (V refers to the Vandermonde matrix). Another suitable option is the Fourier kernel, which satisfies the full-rank property. Both choices have a clear interpretation regarding either shape (for polynomials) or spectra (for Fourier kernels). Figure 2 shows how increasing the order of a polynomial kernel yields indicators sensitive to different types of dependence, thus requiring various adjustments with respect to a reference dataset to achieve an indicator value of 0. For example, $\mathrm{GeDI}\,(x, y; V^2)$ is not sensitive to cubic dependence, which is measured by $\mathrm{GeDI}\,(x, y; V^3)$.

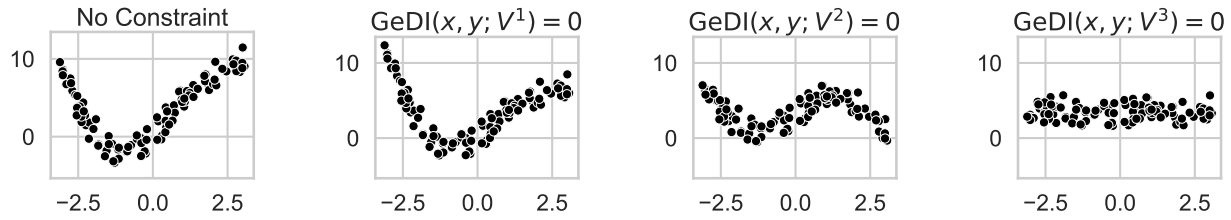
Overall, we recommend the following process for selecting a kernel: 1) choose a family of functions based on relevance to the considered application and ease of interpretation; 2) tune the order to prevent overfitting and numerical issues.

GeDI and DIDI Our approach is designed so that, under specific circumstances, its behavior is analogous to the one of the DIDI (Aghaei et al., 2019). Let us start by differentiating the objective of Equation (9) in d, and stating the optimality condition (i.e., null derivative). This yields:

$$\mathrm{GeDI}\,(x, y; F) = \left| \frac{cov(F\alpha^*, y)}{var(F\alpha^*)} \right| \tag{12}$$

whose full proof is in Appendix A.3. By assuming a polynomial kernel with $k = 1$ we have:

$$\mathrm{GeDI}\,(x, y; V^1) = \left| \frac{cov(x, y)}{var(x)} \right| \tag{13}$$



(a) (b) (c) (d)

Figure 2. Example of how the GeDI semantics depends on the kernel choice. Data in Figure 2(a) are synthetically generated by considering the relation $y(x) = 4sin(x) + x^2 + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$ and $x \sim \mathcal{U}(-\pi, \pi)$. The Figures 2(b)-(c)-(d) showcase the impact of increasing the kernel size in capturing the correlation between x and y. More specifically, in 2(b) we apply a kernel of order 1, which is not able to capture most of the correlations, since the function has no linear terms; in 2(c) we use a kernel of order 2, which captures and cancels out the squared term in $y(x)$, preserving the sinusoidal component only; finally, in 2(d) we choose a kernel of order 3, which cancels out almost all of the sinusoidal component of $y(x)$ given that $sin(x)$ can be approximated as $x - \frac{1}{6}x^3$ as for the Taylor series, hence what is left is just $o(x^5)$ plus noise.

where $\alpha^* = 1$, since there is a single coefficient and $\|\alpha^*\|_1 = 1$. In Appendix A.4, we prove that Equation (13) is equivalent to the DIDI if the protected attribute is binary. As a result, it is possible to consider the $\text{GeDI}(\cdot, \cdot; V^1)$ indicator as a generalization of the (binary) DIDI to continuous protected attributes, thus strengthening the link between our indicators and the already established metrics.

4. GeDI Computation and Constraints

Next, we discuss how GeDI indicators can be computed and used to enforce fairness constraints.

Computation Indicators in the GeDI family are defined via Equation (9), which is a constrained optimization problem. However, it is possible to obtain an unconstrained formulation by applying the following substitutions:

$$\tilde{F}_j = F_j - \mu(F_j) \qquad \tilde{y} = y - \mu(y) \qquad \tilde{\alpha} = |d|\alpha \tag{14}$$

i.e., by centering y and the columns F_j, and by combining d and α in a single vector of variables. The substitutions involve no loss of generality since Equation (14) admits a solution for every $\tilde{\alpha} \in \mathbb{R}^k$. As a result, the problem is reduced to a classical minimal Least Squares formulation:

$$\arg\min_{\tilde{\alpha}} \frac{1}{n}\|\tilde{F}\tilde{\alpha} - \tilde{y}\|_2^2 \tag{15}$$

The solution can be computed via any Linear Regression approach, which typically involves solving the linear system:

$$\tilde{F}^T\tilde{F}\tilde{\alpha}^* = \tilde{F}^T\tilde{y} \tag{16}$$

From here, the absolute value $|d^*|$ can be recovered as:

$$|d^*| = \|\tilde{\alpha}^*\|_1 \tag{17}$$

which is implied by Equation (14) and the constraint $\|\alpha\|_1 = 1$. Overall, the process is simple and well-understood in terms of numerical stability. Additionally, since Equation (15) can be solved to optimality in polynomial time, GeDI indicators are entirely determined by the kernel choice, with benefits for reproducibility. This property is less clearly satisfied by the existing HGR-based indicators: the approach from (Mary et al., 2019) is deterministic but ambiguous unless several implementation details are provided; the method from (Grari et al., 2020) is inherently not deterministic, since it relies on stochastic gradient descent. In Table 2, we summarize the characteristics of the GeDI indicators compared to the HGR approaches.

Fairness Constraints GeDI indicators can be used to formulate fairness constraints. In this setting, the target y can be changed, either because it represents the output of a ML model or because preprocessing/postprocessing techniques are employed. A constraint in the form:

$$\text{GeDI}(x, y; F) \leq q, \quad q \in \mathbb{R}^+ \tag{18}$$

can be translated into a set of piecewise linear relations:

$$\tilde{F}^T\tilde{F}\tilde{\alpha}^* = \tilde{F}^T\tilde{y} \tag{19}$$

$$\|\tilde{\alpha}^*\|_1 \leq q \tag{20}$$

where $\tilde{\alpha}^*$ is not directly computed to avoid matrix inversion. Specific projection-based approaches for constrained ML, such as the one from (Detassis et al., 2021), can rely on mathematical programming to deal with constraints in this form. Indeed, Figure 2 is obtained by adjusting target values to satisfy Equations (19)-(20) and minimizing the Mean Squared Error. As an alternative, it is also possible to obtain a Lagrangian term (i.e., a penalizer) in the form:

$$\lambda \max(0, \|\tilde{\alpha}^*\|_1 - q), \quad \lambda \in \mathbb{R}^+ \tag{21}$$

Table 2. Characteristics of the GeDI indicators when compared to HGR-kde (Mary et al., 2019) and HGR-nn (Grari et al., 2020) approach.

Characteristic	**GeDI**	**HGR-kde**	**HGR-nn**
Interpretability	full	partial	partial
Bias/variance trade-off	transparent	opaque	opaque
Theoretically unrestricted transformations	f	f, g	f, g
Renyi properties	partial	✓	✓
Deterministic	✓	✓	×
Information required for full specification	kernel	discretization, KDE bandwidth	full NN + SGD params
Constraint enforcing	declarative **or** penalizer	penalizer	penalizer
Fine-grained constraints	✓	×	×

where λ is the penalizer weight. In order to avoid matrix inversion in automatic differentiation engines, we can rely on differentiable least-squares operators, such as `tf.linalg.lstsq` or `torch.linalg.lstsq`.

Fine-grained Constraints The choice of the kernel allows a user to specify which types of dependency should be measured. In addition, our formulation allows us to restrict specific terms of the copula transformation by enforcing constraints on individual coefficients. Formally, it is possible to replace the set of constraints from (20) with:

$$|\tilde{\alpha}^*| \leq q, \quad q \in \mathbb{R}^{+k} \tag{22}$$

where the single constraint on $\|\alpha^*\|_1$ is replaced by individual constraints on the absolute value of the coefficients. By algebraic manipulation and taking advantage of the fact that $\tilde{F}^T\tilde{F}$ is positive definite, we obtain a single set of inequalities that avoids any matrix inversion:

$$|\tilde{F}^T\tilde{y}| \leq q \odot \tilde{F}^T\tilde{F} \tag{23}$$

where $\odot$ is the Hadamard (i.e., element-wise) product. Equation (23) can be processed directly by projection approaches or turned into a (vector) penalizer similar to Equation (21).

An interesting setup is to allow a single, easily explainable form of dependency (e.g., linear) while explicitly forbidding others; when applied to a ML model, this setup allows us to obtain a modified sample $\{x_i, y'_i\}$, where y'_i represents the model output after the constraint has been enforced. In this situation, we have:

$$\text{GeDI}\,(x, y'; V^k) = \text{GeDI}\,(x, y'; V^1) \tag{24}$$

The equality holds since all polynomial degree dependencies from 2 to k are explicitly removed, implying that we can measure the discrimination for the constrained model in terms of the DIDI analogy presented in Section 3.

5. Empirical Evaluation

In this section, we discuss an empirical evaluation performed with three objectives: 1) testing how the kernel choice and fine-grained constraints affect the GeDI semantics; 2) studying the relation with other metrics, in particular the DIDI and the HGR indicators; 3) investigating how the use of different ML models and constraint enforcement algorithms impacts effectiveness. Here we report the main details and experiments; additional information is provided in Appendix B. The source code and the datasets are available at `https://github.com/giuluck/GeneralizedDisparateImpact` under MIT license.

Table 3. The four discrimination-aware learning tasks used in our experiments. Type B stands for binary, C for continuous.

	Target		**Protected Att.**	
Dataset	**Name**	**Type**	**Name**	**Type**
Adult	income	B	sex	B
			age	C
Communities & Crimes	violentPerPop	C	race	B
			pctBlack	C

Experimental Setup We rely on two common benchmark datasets in the field of fair AI: *Communities & Crimes*[1] with a continuous target, and *Adult*[2] with a binary target. For both, we define two discrimination-aware learning tasks by considering either a binary or a continuous protected attribute. The resulting tasks are summarized in Table 3.

We restrict our analysis to polynomial kernels due to their ease of interpretation; moreover, their known issues with numerical stability at higher orders make them a computationally interesting benchmark. We consider two setups for our method. In the first, referred to as GeDI-Coarse, we derive an indicator via a V^k kernel and constrain its value as in Equation (20). In the second, referred to as GeDI-Fine, we use the same type of indicator, but the constraint is enforced on individual coefficients; specifically, we allow some degrees of linear dependency while completely forbid-

[1] `https://archive.ics.uci.edu/ml/datasets/Communities+and+Crime`

[2] `https://archive.ics.uci.edu/ml/datasets/Adult`

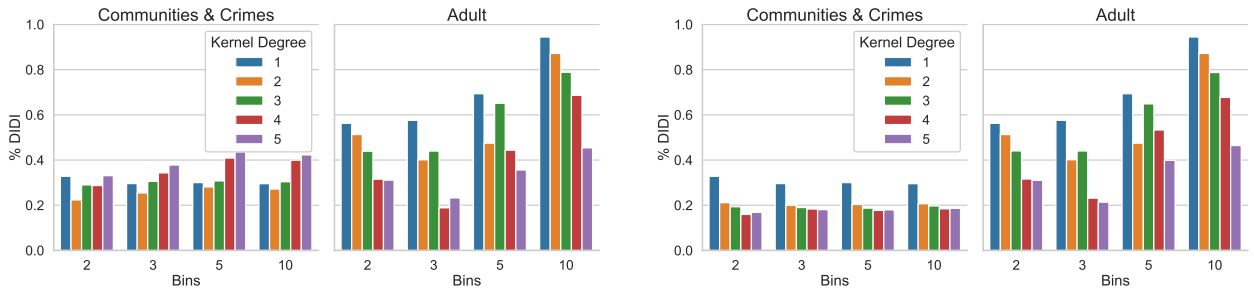


Figure 3. Results from preprocessing tests presented from the perspective of the binned DIDI metrics with different bin numbers both for the coarse-grained (a) and the fine-grained (b) constraint formulation. On the x axis, we plot the number n of bins for the percentage DIDI-n (y axis) value, using five different kernel sizes k (colors) to impose the constraint on $\mathrm{GeDI}(x, y; V^k)$.

ding higher orders (up to k). For binary protected attributes, we only use $k = 1$ as a higher order would not guarantee the full-rank requirement; hence the two setups coincide.

As a representative of the HGR approach, we use the indicator from (Mary et al., 2019) with default parameters. For the DIDI, we use the original formulation from (Aghaei et al., 2019). When adopting DIDI for continuous protected attributes, we apply a quantile-based discretization in n bins before computing the indicator; we refer to the resulting metrics as DIDI-n. This technique does not require specialized indicators but is strongly sensitive to the chosen discretization and prone to overfitting with higher n values. We use *percentage values* when presenting results for experiments with fairness constraints on all metrics, i.e., their values are normalized over those of the original datasets.

Experiments Description We consider two types of tests. The first consist in *preprocessing experiments*, where we directly change the target values to: 1) satisfy coarse- or fine-grained constraints on $\mathrm{GeDI}\,(x, y; V^k)$ and 2) minimize a loss function appropriate to the task – Mean Squared Error for regression and Hamming Distance for classification. No machine learning model is trained in this process. We rely on an exact mathematical programming solver that allows for an efficient and stable formulation for fine-grained constraints (more details in Appendix B.1). We use different kernel orders k, but a fixed threshold for each dataset, corresponding to 20% of the value on the $\mathrm{GeDI}\,(x, y, V^1)$ for the unaltered data. With this design, any difference in behavior is entirely due to changes in the semantics determined by the chosen kernel.

Second, we report results for several *performance tests*, where training problems with fairness constraints are solved using multiple machine learning models and optimization algorithms. These experiments aim to test how these factors affect constraint satisfaction, model accuracy, and generalization. In particular, we train an unconstrained version of a Random Forest (RF), a Gradient Boosting (GB), and a Neural Network (NN) model. Then, the same models are trained using the Moving Targets (MT) algorithm from (Detassis et al., 2021), which allows us to deal with constraints in a declarative fashion. Finally, to investigate the impact of different constraint enforcement methods, we train a version of the neural network model where a penalizer (or Semantics-Based Regularizer, SBR) is used during gradient descent. The penalizer weight is dynamically controlled at training time using the approach from (Fioretto et al., 2021). All these experiments are performed using a 5-fold cross-validation procedure on the entire dataset. See Appendix B.2 for a complete list of the technical settings, and Appendix B.3 for a broader description of the constrained models. Similarly to the preprocessing tests, we use a fixed threshold for all constraints, corresponding to 20% the value of $\mathrm{GeDI}\,(x, y, V^1)$ for the original datasets.

Preprocessing experiments Figure 3 shows the outcome of our preprocessing tests for the $\mathrm{GeDI}\,(x, y; V^k)$, both for coarse-grained (a) and fine-grained (b) approaches. In particular, we report the results for multiple kernel sizes (colors) and different bin numbers (x axis) from the perspective of the binned DIDI metrics (y axis). All GeDI constraints are satisfied by construction due to the use of an exact solver. Since increasing the bin number in DIDI-n enables measuring more complex dependencies, and both indicators focus on functional dependencies, the figures allow us to observe how the GeDI semantics depends on the kernel order.

We notice that increasing the kernel order with the GeDI-Fine approach tends to improve the discretized DIDI across all bin numbers (Figure 3(b)); in the few cases where the DIDI-3 and DIDI-5 worsen by adding GeDI constraints, we expect that the discrepancy is due to how the bin boundaries "cut" the shapes induced by the polynomial kernels, i.e., it is

Table 4. Performance tests results for tasks with *binary* protected feature.

	Communities & Crimes					*Adult*				
	R2		GeDI-V1 %		Time	Accuracy		GeDI-V1 %		Time
	train	val	train	val		train	val	train	val	
RF	*0.95 ± 0.00*	*0.63 ± 0.02*	*1.00 ± 0.00*	*1.02 ± 0.12*	*5.19*	*1.00 ± 0.00*	*0.85 ± 0.00*	*1.00 ± 0.00*	*0.94 ± 0.05*	*1.54*
+MT	0.66 ± 0.01	0.52 ± 0.04	0.24 ± 0.00	0.28 ± 0.10	54.00	**0.95** ± 0.00	0.84 ± 0.00	0.20 ± 0.00	0.72 ± 0.03	**128.28**
GB	*0.87 ± 0.01*	*0.63 ± 0.02*	*1.03 ± 0.00*	*1.05 ± 0.11*	*2.43*	0.87 ± 0.00	*0.86 ± 0.00*	*0.88 ± 0.01*	*0.88 ± 0.04*	*2.25*
+MT	0.64 ± 0.01	**0.54** ± 0.02	0.21 ± 0.00	0.27 ± 0.09	29.25	0.85 ± 0.00	**0.85** ± 0.00	0.35 ± 0.02	0.34 ± 0.04	129.65
NN	*0.99 ± 0.01*	*0.58 ± 0.01*	*0.99 ± 0.02*	*0.97 ± 0.12*	*7.64*	*0.89 ± 0.00*	*0.83 ± 0.00*	*1.03 ± 0.05*	*1.03 ± 0.08*	*55.31*
+MT	**0.89** ± 0.02	0.49 ± 0.04	0.20 ± 0.02	0.32 ± 0.11	85.81	0.86 ± 0.00	0.83 ± 0.00	0.19 ± 0.01	0.17 ± 0.06	2593.11
+SBR	0.88 ± 0.01	0.45 ± 0.05	**0.10** ± 0.03	**0.24** ± 0.07	**6.36**	0.84 ± 0.00	0.84 ± 0.00	**0.17** ± 0.02	**0.17** ± 0.03	563.58

Table 5. Performance tests results for tasks with *continuous* protected feature.

	Communities & Crimes					*Adult*				
	R2		GeDI-V5 %		Time	Accuracy		GeDI-V5 %		Time
	train	val	train	val		train	val	train	val	
RF	*0.95 ± 0.00*	*0.63 ± 0.02*	*1.70 ± 0.08*	*2.03 ± 0.59*	*5.23*	*1.00 ± 0.00*	*0.85 ± 0.00*	*3.40 ± 0.06*	*3.76 ± 0.18*	*2.74*
+MT-Fine	0.48 ± 0.01	0.36 ± 0.02	0.24 ± 0.01	0.64 ± 0.20	61.97	**0.92** ± 0.00	0.78 ± 0.00	**0.20** ± 0.00	1.13 ± 0.25	256.02
+MT-Coarse	0.60 ± 0.01	0.46 ± 0.03	0.24 ± 0.01	0.69 ± 0.27	64.91	**0.92** ± 0.00	0.79 ± 0.00	**0.20** ± 0.00	0.57 ± 0.19	**60.89**
GB	0.87 ± 0.01	*0.63 ± 0.02*	*1.67 ± 0.10*	*2.09 ± 0.50*	*2.52*	*0.87 ± 0.00*	*0.86 ± 0.00*	*3.59 ± 0.06*	*3.59 ± 0.16*	*4.14*
+MT-Fine	0.47 ± 0.01	0.37 ± 0.02	0.23 ± 0.00	**0.55** ± 0.24	30.58	0.81 ± 0.00	0.80 ± 0.01	0.65 ± 0.06	0.76 ± 0.26	658.27
+MT-Coarse	0.60 ± 0.01	**0.49** ± 0.04	0.22 ± 0.00	0.92 ± 0.44	31.93	0.83 ± 0.00	**0.83** ± 0.01	0.86 ± 0.03	0.90 ± 0.21	66.18
NN	*0.99 ± 0.01*	*0.58 ± 0.01*	*1.79 ± 0.14*	*1.78 ± 0.35*	*7.58*	*0.89 ± 0.00*	*0.83 ± 0.00*	*4.09 ± 0.16*	*4.12 ± 0.21*	*306.42*
+MT-Fine	0.70 ± 0.03	0.32 ± 0.05	0.27 ± 0.03	1.09 ± 0.67	94.14	0.80 ± 0.01	0.79 ± 0.01	0.32 ± 0.08	**0.51** ± 0.29	2096.00
+MT-Coarse	**0.78** ± 0.06	0.40 ± 0.04	0.21 ± 0.02	0.90 ± 0.54	99.33	0.80 ± 0.01	0.78 ± 0.01	0.29 ± 0.05	0.63 ± 0.48	1378.40
+SBR-Fine	0.64 ± 0.03	0.43 ± 0.04	**0.18** ± 0.03	0.92 ± 0.47	**15.87**	0.84 ± 0.00	**0.83** ± 0.00	0.89 ± 0.04	0.91 ± 0.10	793.28
+SBR-Coarse	0.67 ± 0.03	0.45 ± 0.02	0.21 ± 0.04	0.98 ± 0.52	16.20	0.84 ± 0.00	**0.83** ± 0.00	0.91 ± 0.04	0.93 ± 0.06	925.15

due to discretization noise. The same behavior is observed for GeDI-Coarse (Figure 3(a)), although with a lower degree of consistency. Such a trend confirms that increasing the kernel order makes our indicators capable of measuring more complex dependencies. In the GeDI-Fine case, all higher-order dependencies are canceled, while the GeDI-Coarse approach is more flexible regarding which types of dependency are allowed; this explains why enforcing a 20% threshold with the GeDI-Fine approach also leads to similar DIDI-n values since the semantics of the two approaches are similar. Despite this, we underline that the equivalence from Section 4 does not strictly hold in the analyzed datasets since the protected attribute is not natively categorical.

Performance experiments Table 4 shows the results for the performance experiments in tasks with binary protected attributes. We report the average values on the *train* and *validation* splits for: 1) an accuracy metric, i.e., *R2* for regression tasks and *Accuracy* for classification ones, and 2) our fairness indicator $\mathrm{GeDI}(\cdot,\cdot;V^1)$, which in this case is equivalent to the DIDI. Additionally, we report a single column with training times. Rows related to unconstrained models are italicized, and the best results for the constrained models are highlighted in bold font. Table 5 reports similar data for tasks with continuous protected attributes. We rely on $\mathrm{GeDI}\left(x,y,V^5\right)$ as a fairness indicator, and we enforce both GeDI-Fine and GeDI-Coarse since they differ in formulation for higher-order kernels.

The results obtained using different models and algorithms in the four tasks analyzed are rather diverse regarding accuracy, degree of constraint satisfaction, and training time. This stresses the advantage of having access to multiple learning models and constraint enforcing methods, which our approach provides. The behavior is reasonably consistent in terms of both accuracy and constraint satisfaction, with low standard deviations. The only exception is the classification task (*Adult*) with continuous protected attribute: in this scenario, and in general, when dealing with continuous protected attributes, the unconstrained models tend to introduce additional discrimination on top of the existing one. As a consequence, our thresholds become particularly demanding since they are based on GeDI values for the original dataset. Stringent thresholds require significant alterations to the input/output relation that the models would naturally learn, thus exacerbating generalization issues.

Finally, in Figure 4, we analyze the link between GeDI and the HGR indicator by reporting their values for all of our performance tests. The displayed results come from unconstrained and constrained models and different tasks. According to this figure, we highlight that: (1) GeDI is a valid option for fairness metrics, since it correlates with an already established metrics (namely HGR) but (2) it is not redundant, as its semantics differs from the one of HGR. Indeed, as expected, we can see that the two metrics are correlated (suggesting they capture related concepts of fairness), but with a high variance (highlighting the difference

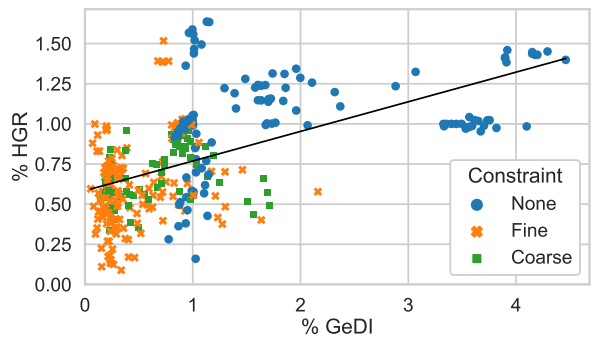


Figure 4. Percentage values of GeDI and HGR indicator from (Mary et al., 2019) on the performance tests.

in their semantics). For this reason, GeDI offers a valid alternative to the set of existing metrics, enriching the range of options among which the practitioners can select the indicator that best fits their application.

6. Conclusions

In this work, we introduce the GeDI family of indicators to extend the available options for measuring and enforcing fairness with continuous protected attributes. Our indicators complement the existing HGR-based solutions regarding semantics and provide a more interpretable, transparent, and controllable fairness metric. GeDI allows the user to specify which types of dependency are relevant and how they should be restricted.

While some of the design choices in our approach are incompatible with the HGR formulation, others could be applied in principle. The resulting configurable HGR-based solution would have technical properties similar to the GeDI indicators. Preliminary work in this direction corroborates this conjecture, making us regard this topic as a promising area for future research.

Acknowledgements

This work has been supported by the project TAILOR (funded by European Union's Horizon 2020 research and innovation programme, GA No. 952215), and by the project AEQUITAS (funded by European Union's Horizon Europe research and innovation programme, GA No. 101070363)[3].

Funded by the European Union

[3]Discaimer: This paper reflects only the authors' views. The European Commission is not responsible for any use that may be made of the information it contains.

References

Aghaei, S., Azizi, M. J., and Vayanos, P. Learning optimal and fair decision trees for non-discriminative decision-making. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pp. 1418–1426. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33011418. URL https://doi.org/10.1609/aaai.v33i01.33011418.

Calders, T. and Verwer, S. Three naive bayes approaches for discrimination-free classification. *Data Min. Knowl. Discov.*, 21:277–292, 09 2010. doi: 10.1007/s10618-010-0190-x.

Calmon, F., Wei, D., Vinzamuri, B., Natesan Ramamurthy, K., and Varshney, K. R. Optimized pre-processing for discrimination prevention. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/9a49a25d845a483fae4be7e341368e36-Paper.pdf.

Detassis, F., Lombardi, M., and Milano, M. Teaching the old dog new tricks: Supervised learning with constraints. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5):3742–3749, may 2021. doi: 10.1609/aaai.v35i5.16491.

Donini, M., Oneto, L., Ben-David, S., Shawe-Taylor, J., and Pontil, M. Empirical risk minimization under fairness constraints. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 2796–2806, Red Hook, NY, USA, 2018. Curran Associates Inc.

Feldman, M., Friedler, S. A., Moeller, J., Scheidegger, C., and Venkatasubramanian, S. Certifying and removing disparate impact. In Cao, L., Zhang, C., Joachims, T., Webb, G. I., Margineantu, D. D., and Williams, G. (eds.), *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pp. 259–268. ACM, 2015. doi: 10.1145/2783258.2783311. URL https://doi.org/10.1145/2783258.2783311.

Fioretto, F., Hentenryck, P. V., Mak, T. W. K., Tran, C., Baldo, F., and Lombardi, M. Lagrangian duality for constrained deep learning. In *Machine Learning and Knowledge Discovery in Databases. Applied Data Science and*

Demo Track, pp. 118–135. Springer International Publishing, 2021. doi: 10.1007/978-3-030-67670-4_8. URL https://doi.org/10.1007/978-3-030-67670-4_8.

Gianfrancesco, M. A., Tamang, S., Yazdany, J., and Schmajuk, G. Potential Biases in Machine Learning Algorithms Using Electronic Health Record Data. *JAMA Internal Medicine*, 178(11):1544–1547, 11 2018. ISSN 2168-6106. doi: 10.1001/jamainternmed.2018.3763. URL https://doi.org/10.1001/jamainternmed.2018.3763.

Grari, V., Lamprier, S., and Detyniecki, M. Fairness-aware neural rényi minimization for continuous features. In Bessiere, C. (ed.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 2262–2268. International Joint Conferences on Artificial Intelligence Organization, 7 2020. doi: 10.24963/ijcai.2020/313. URL https://doi.org/10.24963/ijcai.2020/313. Main track.

Hardt, M., Price, E., Price, E., and Srebro, N. Equality of opportunity in supervised learning. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/9d2682367c3935defcb1f9e247a97c0d-Paper.pdf.

Julia, A., Jeff, L., Surya, M., and Lauren, K. Machine bias: There's software used across the country to predict future criminals and it's biased against blacks. *ProPublica*, 2016. URL https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing.

Kamiran, F. and Calders, T. Data preprocessing techniques for classification without discrimination. *Knowl. Inf. Syst.*, 33(1):1–33, 2011. doi: 10.1007/s10115-011-0463-8. URL https://doi.org/10.1007/s10115-011-0463-8.

Kamishima, T., Akaho, S., and Sakuma, J. Fairness-aware learning through regularization approach. In *2011 IEEE 11th International Conference on Data Mining Workshops*, pp. 643–650, 2011. doi: 10.1109/ICDMW.2011.83.

Kamishima, T., Akaho, S., Asoh, H., and Sakuma, J. Fairness-aware classifier with prejudice remover regularizer. In Flach, P. A., De Bie, T., and Cristianini, N. (eds.), *Machine Learning and Knowledge Discovery in Databases*, pp. 35–50, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

Komiyama, J., Takeda, A., Honda, J., and Shimao, H. Nonconvex optimization for regression with fairness constraints. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2737–2746. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/komiyama18a.html.

Mary, J., Calauzènes, C., and Karoui, N. E. Fairness-aware learning for continuous attributes and treatments. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 4382–4391. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/mary19a.html.

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., and Galstyan, A. A survey on bias and fairness in machine learning. *ACM Comput. Surv.*, 54(6), jul 2021. ISSN 0360-0300. doi: 10.1145/3457607. URL https://doi.org/10.1145/3457607.

Menon, A. K. and Williamson, R. C. The cost of fairness in binary classification. In Friedler, S. A. and Wilson, C. (eds.), *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, volume 81 of *Proceedings of Machine Learning Research*, pp. 107–118. PMLR, 23–24 Feb 2018. URL https://proceedings.mlr.press/v81/menon18a.html.

Padala, M. and Gujar, S. Fnnc: Achieving fairness through neural networks. In Bessiere, C. (ed.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 2277–2283. International Joint Conferences on Artificial Intelligence Organization, 7 2020. doi: 10.24963/ijcai.2020/315. URL https://doi.org/10.24963/ijcai.2020/315. Main track.

Rényi, A. On measures of dependence. *Acta Mathematica Academiae Scientiarum Hungarica*, 10:441–451, 1959.

Supreme Court of the United States. Griggs v. duke power co., 1971.

Zafar, M. B., Valera, I., Rodriguez, M. G., and Gummadi, K. P. Fairness beyond disparate treatment & disparate impact. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, apr 2017. doi: 10.1145/3038912.3052660. URL https://doi.org/10.1145%2F3038912.3052660.

A. Proofs of Section 3

A.1. Least Squares Problem and Pearson's Correlation

Consider two random samples $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$ and the following least square problem:

$$\arg\min_r \frac{1}{n} \left\| r\frac{x-\mu_x}{\sigma_x} - \frac{y-\mu_y}{\sigma_y} \right\|_2^2 \tag{25}$$

where r is the sample Pearson's correlation coefficient. This is a basic Linear Regression problem over standardized variables. Since the problem is convex, an optimal solution can be found by requiring the loss function gradient to be null. By differentiating over r we get:

$$\frac{1}{n}\left(r\frac{x-\mu_x}{\sigma_x} - \frac{y-\mu_y}{\sigma_y}\right)^T \frac{x-\mu_x}{\sigma_x} = 0 \tag{26}$$

By algebraic manipulations we get:

$$r\frac{1}{n}\frac{(x-\mu_x)^T(x-\mu_x)}{\sigma_x^2} = \frac{1}{n}\frac{(x-\mu_x)^T(y-\mu_y)}{\sigma_x\sigma_y} \tag{27}$$

By applying the definition of variance and standard deviation, we have that $\frac{1}{n}(x-\mu_x)^T(x-\mu_x) = \sigma_x^2$, thus leading us to:

$$r = \frac{1}{n}\frac{(x-\mu_x)^T(y-\mu_y)}{\sigma_x\sigma_y} \tag{28}$$

This is the value of the sample Pearson correlation coefficient, which is therefore equivalent to the optimal parameter for a properly defined Linear Regression problem.

A.2. Bilevel Optimization Problem Simplification

We prove that the objective of the outer and inner optimization problems are in this case aligned, thus making it possible to simplify the formulation.

Specifically, let us consider two pairs of copula transformations f'_x, g'_y and f''_x, g''_y, and let r' and r'' be the corresponding values for the sample Pearson correlation coefficient. Let us assume that one pair of transformation leads to a smaller MSE, i.e.:

$$\frac{1}{n}\left\|r' f'_x - g'_y\right\|_2^2 < \frac{1}{n}\left\|r'' f''_x - g''_y\right\|_2^2 \tag{29}$$

The two terms can be expanded as:

$$r'^2\frac{f_x'^T f'_x}{n} - 2r'\frac{f_x'^T g'_y}{n} + \frac{g_y'^T g'_y}{n} < r''^2\frac{f_x''^T f''_x}{n} - 2r''\frac{f_x''^T g''_y}{n} + \frac{g_y''^T g''_y}{n} \tag{30}$$

Due to our zero-mean assumption, all quadratic terms in the form $f_x'^T f'_x/n$, etc., correspond to sample variances. Again due to our assumptions, variances have unit value. Therefore, we have:

$$r'^2 - 2r'\frac{f_x'^T g'_y}{n} + 1 < r''^2 - 2r''\frac{f_x''^T g''_y}{n} + 1 \tag{31}$$

Now, as established in Equation (28), we have that $r' = f_x'^T g'_y/n$ and $r'' = f_x''^T g''_y/n$. Therefore, we have:

$$r'^2 - 2r'^2 + 1 < r''^2 - 2r''^2 + 1 \tag{32}$$

Since all steps leading from Equation (29) to Equation (32) are invertible, we have that:

$$r'^2 > r''^2 \quad \Leftrightarrow \quad \frac{1}{n}\left\|r' f'_x - g'_y\right\|_2^2 < \frac{1}{n}\left\|r'' f''_x - g''_y\right\|_2^2 \tag{33}$$

In other words, maximizing the square of the sample HGR is equivalent to minimizing the Mean Squared Error. Now, maximizing r^2 corresponds to maximizing either r or minimizing $-r$. Since the copula transformations are generic, we can

always change the sign of r by changing the sign of either f or g. This means that maximizing r is in fact equivalent to maximizing r^2 in our context.

Overall, this suggests an approach for computing the sample HGR that does not rely on bilevel optimization. Namely, first we determine f and g by solving the Least Squares Problem:

$$\arg\min_{f,g,r} \frac{1}{n} \|rf_x - g_y\|_2^2 \tag{34}$$

Then we compute the sample HGR as the absolute value of the sample Pearson correlation, i.e.:

$$\text{hgr}(x, y) = |r(f_x, g_x)| \tag{35}$$

Computing the absolute value is equivalent to performing the sign swap on one of the two transformations, as previously described.

A.3. Closed-form Computation of Generalized Disparate Impact

We start from the definition of the GeDI family of indicators as for Equation (9), i.e.:

$$\arg\min_{d,\alpha} \frac{1}{n} \|d(F\alpha - \mu(F\alpha)) - (y - \mu(y))\|_2^2 \qquad \text{s.t. } \|\alpha\|_1 = 1 \tag{36}$$

We can embed the constraint into the objective function $C(d, \alpha, \lambda)$ using a Lagrangian multiplier λ, from which we obtain:

$$\arg\min_{d,\alpha,\lambda} C(d, \alpha, \lambda) \qquad \text{s.t. } C(d, \alpha, \lambda) = \frac{1}{n} \|d(F\alpha - \mu(F\alpha)) - (y - \mu(y))\|_2^2 + \lambda(\|\alpha\|_1 - 1) \tag{37}$$

The optimal solution of the objective function can be found by requiring its gradient to be null. This implies that having a null derivative in d is a necessary condition for optimality, which we can exploit to retrieve the value of d^* as follows. First, we compute the derivative of $C(d, \alpha, \lambda)$ with respect to d:

$$\frac{\partial C(d, \alpha, \lambda)}{\partial d} = \frac{2}{n}(F\alpha - \mu(F\alpha))^T((F\alpha - \mu(F\alpha))d - (y - \mu(y))) \tag{38}$$

Then, by requiring Equation (38) to be null, we get the following equivalence:

$$\frac{1}{n}(F\alpha^* - \mu(F\alpha^*))^T(F\alpha^* - \mu(F\alpha^*))d^* = \frac{1}{n}(F\alpha^* - \mu(F\alpha^*))^T(y - \mu(y)) \tag{39}$$

The scalar values $\frac{1}{n}(F\alpha^* - \mu(F\alpha^*))^T(F\alpha^* - \mu(F\alpha^*))$ and $\frac{1}{n}(F\alpha^* - \mu(F\alpha^*))^T(y - \mu(y))$ represent the variance of the vector $F\alpha^*$ and the covariance between $F\alpha^*$ and y, respectively. Hence, we get the closed-form value of d^* as:

$$d^* = \frac{\text{cov}(F\alpha^*, y)}{\text{var}(F\alpha^*)} \tag{40}$$

Finally, since Equation (8) ties the value of the GeDI indicator to the absolute value of d^*, we get:

$$\text{GeDI}\,(x, y; F) = \left|\frac{\text{cov}(F\alpha^*, y)}{\text{var}(F\alpha^*)}\right| \tag{41}$$

A.4. Equivalence Between GeDI and DIDI

We consider the case of continuous target y and binary protected attribute x. The DIDI can be computed as for Equation (1):

$$\text{DIDI}_r(x,y) = \sum_{v \in \mathcal{X}} \left| \frac{\sum_{i=1}^n y_i I(x_i = v)}{\sum_{i=1}^n I(x_i = v)} - \frac{1}{n}\sum_{i=1}^n y_i \right| \tag{42}$$

Since x is a binary attribute ($\mathcal{X} = \{0,1\}$), we can replace the indicator function $I(x_i = v)$ with either $1 - x_i$ or x_i depending on the value v, obtaining:

$$\text{DIDI}_r(x,y) = \left| \frac{\sum_{i=1}^n (1-x_i) y_i}{\sum_{i=1}^n (1-x_i)} - \frac{1}{n}\sum_{i=1}^n y_i \right| + \left| \frac{\sum_{i=1}^n x_i y_i}{\sum_{i=1}^n x_i} - \frac{1}{n}\sum_{i=1}^n y_i \right| \tag{43}$$

By algebraic manipulations within the summations, and by dividing for the constant factor n both the numerator and the denominator of every fractional term, we can rewrite Equation (43) as:

$$\text{DIDI}_r(x,y) = \left| \frac{\mu_y - \mu_{xy}}{1 - \mu_x} - \mu_y \right| + \left| \frac{\mu_{xy}}{\mu_x} - \mu_y \right| \tag{44}$$

where μ_x and μ_y represent the average value of vectors x and y respectively, and μ_{xy} is the average value of vector $x \odot y$.

We can further manipulate this equation to obtain:

$$\text{DIDI}_r(x,y) = \left| \frac{\mu_x \mu_y - \mu_{xy}}{1 - \mu_x} \right| + \left| \frac{\mu_{xy} - \mu_x \mu_y}{\mu_x} \right| \tag{45}$$

The numerators of both terms are equal except for the sign, hence we can join the two absolute values by simply swapping the sign of one of them. Moreover, we can notice that $\mu_{xy} - \mu_x \mu_y$ represents the covariance between x and y. Therefore:

$$\text{DIDI}_r(x,y) = \left| \frac{\text{cov}(x,y)}{1 - \mu_x} + \frac{\text{cov}(x,y)}{\mu_x} \right| = \left| \frac{\text{cov}(x,y)}{\mu_x - \mu_x^2} \right| \tag{46}$$

Since x is a binary vector, it is invariant to the power operator. Thus, $\mu_x = \mu_{x^2}$ and, subsequently, the denominator of Equation (46) reduces to the variance of x. Hence:

$$\text{DIDI}_r(x,y) = \left| \frac{\text{cov}(x,y)}{\text{var(x)}} \right| = \text{GeDI}\,(x,y;V^1) \tag{47}$$

The same reasoning can be applied when both x and y are binary vectors. Again, the indicator function $I(x_i = v)$ can be replaced with either $1 - x_i$ or x_i depending on the value v. Similarly, since in this case we are computing the DIDI_c, we replace $I(y_i = u)$ with $1 - y_i$ and y_i, and $I(y_i = u \wedge x_i = v)$ with the corresponding product between the previous terms. Eventually, we obtain:

$$\begin{aligned}\text{DIDI}_c(x,y) = &\left| \frac{\sum_{i=1}^n (1-x_i)(1-y_i)}{\sum_{i=1}^n (1-x_i)} - \frac{1}{n}\sum_{i=1}^n (1-y_i) \right| + \left| \frac{\sum_{i=1}^n x_i (1-y_i)}{\sum_{i=1}^n x_i} - \frac{1}{n}\sum_{i=1}^n (1-y_i) \right| + \\ &\left| \frac{\sum_{i=1}^n (1-x_i) y_i}{\sum_{i=1}^n (1-x_i)} - \frac{1}{n}\sum_{i=1}^n y_i \right| + \left| \frac{\sum_{i=1}^n x_i y_i}{\sum_{i=1}^n x_i} - \frac{1}{n}\sum_{i=1}^n y_i \right|\end{aligned} \tag{48}$$

By using the same notation as in Equation (44) and applying analogous mathematical manipulations, we get to:

$$\text{DIDI}_c(x,y) = \left| \frac{\mu_x \mu_y - \mu_{xy}}{1 - \mu_x} \right| + \left| \frac{\mu_{xy} - \mu_x \mu_y}{\mu_x} \right| + \left| \frac{\mu_x \mu_y - \mu_{xy}}{1 - \mu_x} \right| + \left| \frac{\mu_{xy} - \mu_x \mu_y}{\mu_x} \right| \tag{49}$$

This value is twice as much that in Equation (45), meaning that the DIDI_c is twice our indicator $\text{GeDI}\,(x,y;V^1)$. This is not a problem since we can get rid of the constant scaling factor. Moreover, when we constrain the DIDI indicator up to a relative threshold that depends on the original level of discrimination, the scaling factor cancels out naturally.

B. Appendix of Section 5

B.1. Optimized Fine-grained Formulation

We want to solve the following optimization model:

$$\begin{aligned} &\arg\min_z \|z-y\|_2^2 \\ \text{s.t. } &\text{GeDI}\left(x,z;V^k\right) = \text{GeDI}\left(x,z;V^1\right), \\ &\text{GeDI}\left(x,z;V^1\right) \leq q \end{aligned} \tag{50}$$

where the GeDI indicator is defined as in Equation (8) and computed according to Equations (16)-(17).

Since we impose $\text{GeDI}\left(x,z;V^k\right) = \text{GeDI}\left(x,z;V^1\right)$, the optimal vector $\tilde{\alpha}^*$ which solves Equation (16) for $F = V^k$ is such that all the elements are null apart from the first one. It follows that:

$$\tilde{F_1}^T \tilde{F} \tilde{\alpha}_1^* = \tilde{F}^T \tilde{y} \tag{51}$$

where $\tilde{F}_1 = x - \mu(x) = \tilde{x}$ is the first column of the kernel matrix, namely the only one paired to a non-null $\tilde{\alpha}$ coefficient.

Equation (51) is a system of k equations with a single variable. From the first equation we can retrieve the value of $\tilde{\alpha}_1^*$ in the following way:

$$\tilde{x}^T \tilde{x} \tilde{\alpha}_1^* = \tilde{x}^T \tilde{y} \tag{52}$$

whose solution is $\tilde{\alpha}_1^* = \frac{\text{cov}(x,y)}{\text{var}(x)}$. According to Equation (17), the GeDI indicator can be retrieved as the absolute value of $\tilde{\alpha}_1^*$ since all the remaining items of $\tilde{\alpha}^*$ are null. This result is in fact equivalent to $\text{GeDI}\left(x,y;V^1\right)$.

In addition to that, the remaining $k-1$ equalities from Equation (51) must be satisfied. This can be achieved by operating on the projections. Indeed, the optimization problem defined in Equation (50) has n free variables – i.e., the vector z –, with $n \gg k$ in almost all the practical cases. These equalities are in the form:

$$\tilde{F_1}^T \tilde{F}_j \tilde{\alpha}_1^* = \tilde{F_j}^T \tilde{y} \qquad \forall j \in \{2,\dots,k\} \tag{53}$$

where $\tilde{F}_j = x^j - \mu(x^j) = \tilde{x}^j$.

Since $\tilde{\alpha}_1^*$ can be computed in closed-form, we can plug it into Equation (53), eventually obtaining the following set of constraints:

$$\text{cov}(x^j,x)\,\text{cov}(x,y) = \text{cov}(x^j,y)\,\text{var}(x) \qquad \forall j \in \{2,\dots,k\} \tag{54}$$

which can be used to solve Equation (50) without the need to set up the respective Least Squares Problems.

B.2. Experimental Setup for Reproducibility

All the models are trained on a machine with an Intel Core I9 10920X 3.5G and 64GB of RAM.

The Random Forest and Gradient Boosting models are based on their available implementations in `scikit-learn 1.0.2` with default parameters, while the Neural Network and the semantics-based Regularization models are implemented using `torch 1.13.1`. Specifically, the hyper-parameters of neural-based models are obtained via a grid search analysis with train-test splitting on the two unconstrained tasks aimed at maximizing test accuracy. In particular, the neural models are trained for 200 epochs with batch size 128 and two layers of 256 units for the *Communities & Crimes* tasks and three layers of 32 units for the *Adult* tasks. The only exception is the semantics-based Regularization model which runs for 500 epochs to compensate the fact that it is trained full-batch in order to better deal with group constraints. Additionally, all the neurons have ReLU activation function except for the output one, which has either linear or sigmoid activation depending on whether the task is regression or classification, respectively. Accordingly, the loss function is either mean square error or binary crossentropy, but in both cases the training is performed using the Adam optimizer with default hyperparameters.

As regards Moving Target's optimization routine, we leverage the Python APIs offered by `gurobipy 10.0` to solve it within our `Python 3.7` environment. The backend solver is `Gurobi 10.0`, for which we use the default parameters except for WorkLimit $= 60$.

B.3. Constrained Approaches Descriptions

Here we will briefly present the two approaches that we use to enforce our constraint in several ML models. Both approaches are employed to solve the following constrained optimization problem:

$$\arg\min_{\theta} \mathcal{L}(f(x;\theta), y) \qquad \text{s.t. } f(x;\theta) \in \mathcal{C} \tag{55}$$

where $f(x;\theta)$ represent the predictions of the ML model f with learned parameters θ, $\mathcal{C}$ is the feasible region, and $\mathcal{L}$ is a task-specific loss function. For both the approaches, the original papers are provided as reference for a more details.

Moving Targets Moving Targets (MT) (Detassis et al., 2021) is a framework for constrained ML based on bilevel decomposition. It works by iteratively alternating between a *learner step*, which is in charge of training the ML model, and a *master step*, which projects the solution onto the feasible space while minimizing the distance between both model's predictions and original targets. In practice, MT solves the problem described in Equation (55) by alternating between the two following sub-problems:

$$z_{(i)} = \arg\min_{z} \mathcal{L}(z, f(x;\theta_{(i-1)})) + \alpha_{(i)} \cdot \mathcal{L}(z, y) \qquad \text{s.t. } z \in \mathcal{C} \tag{56}$$

$$\theta_{(i)} = \arg\min_{\theta} \mathcal{L}(f(x;\theta), z_{(i)}) \tag{57}$$

where the subscript (i) indicates values obtained at the ith iteration, $\alpha_{(i)}$ is a factor used to balance the distance between the original targets and the predictions during the *master step*, and the first value $\theta_{(0)}$ is obtained by pre-training the ML model.

The algorithm is perfectly suited for our purpose for three main reasons: 1) it is model-agnostic, thus allowing us to test the behaviour of our constraint for different models, each of which has its own specific bias and limitation; 2) it is conceived to deal with declarative group-constraints like the GeDI one, since it allows to train the ML model using mini-batches if needed; and 3) it can naturally deal with classification tasks without the need of relaxing the class targets to class probabilities.

As regards our experiments, the *learner step* is performed as a plain ML task leveraging either `scikit-learn` or `torch` depending on the chosen model. Instead, the *master step* is formulated as a Mixed-integer Program (MIP) and delegated to the `Gurobi` solver. More specifically, for the coarse-grained constraint formulation we define k free variables for the coefficients $\tilde{\alpha}$ and retrieve their values by imposing an equality constraints according to Equation (19); the overall constraint on the GeDI value is then imposed according to Equation (20). As for the fine-grained formulation, we force the constraint $-q \leq \text{cov}(x, z) \leq q$ in order not to exceed the defined threshold and, additionally, we impose the satisfaction of the $k-1$ equality constraints defined in Equation (53), as motivated by the optimized formulation showed in Appendix B.1.

Finally, in our setup we define each value $\alpha_{(i)}$ as the ith item of the harmonic series, namely $\alpha_{(i)} = i^{-1}$, and the loss function $\mathcal{L}$ is either MSE or Hamming Distance depending on whether the task is regression or classification. Respectively to the Semantic-based Regularization approach, a key advantage of MT lies in the fact that MIP models can naturally deal with discrete variables, thus requiring no need to relax the problem to the continuous domain.

Semantic-based Regularization The Lagrangian Dual framework for Semantic-based Regularization (Fioretto et al., 2021) extends the concept of loss penalizers by allowing for an automated calibration of the lagrangian multipliers. Specifically, let us consider the case in which we have a penalty vector $\mathcal{P}(y, f(x;\theta)) \in \mathbb{R}^{+k}$ which represents the violations for k different constraints. We can embed these violations in the loss function $\mathcal{L}(y, f(x;\theta)) \in \mathbb{R}^{+}$ of our neural model by multiplying each violation with its respective multiplier λ_i. The overall loss will be

$$\mathcal{L}(y, f(x;\theta)) + \lambda^T \mathcal{P}(y, f(x;\theta)).$$

The main pitfall of this approach is that it requires to fine-tune the multipliers vector λ according to the task. The Lagrangian Dual framework solves this problem by proposing a bilevel optimization schema where: 1) the loss function is minimized via gradient-descent with fixed multipliers, and 2) the loss function is maximized via gradient-ascent with fixed network structure. In practice, this is equivalent to perform the following steps in sequence:

$$\theta_{(i)} = \arg\min_{\theta} \left\{ \mathcal{L}(y, f(x;\theta)) + \lambda_{(i-1)}^T \mathcal{P}(y, p) \right\} \tag{58}$$

$$\lambda_{(i)} = \arg\min_{\lambda} \left\{ \mathcal{L}(y, f(x;\theta_{(i)})) + \lambda^T \mathcal{P}(y, p) \right\} \tag{59}$$

where the subscript (i) indicates the value of the ith iteration – performed once per batch –, and $\lambda_{(0)}$ is a null vector.

As regards our experiments, the $\tilde{\alpha}$ coefficients are computed via the `torch.linalg.lstsq` differentiable operator. In the coarse-grained formulation, the penalizers vector $\mathcal{P}(y, f(x;\theta))$ consists of a single which is computed according to Equation (21). Instead, in the fine-grained formulation the vector has k different terms – one for each $\tilde{\alpha}_i$ –, of which all but the first term exhibit a violation proportional to their absolute value.

A major pitfall of this approach is due to the incompatibility of the *round* operator with gradient-based learning algorithms, being its gradient null for each x. This makes it necessary to relax the formulation of the GeDI indicator (only) in classification tasks, by adopting predicted probabilities rather than predicted class targets.